目次

第 1 章 緒言 3

1.1 研究の背景 3

1.2 研究の目的 4

1.3 論文の構成 4

第 2 章 つくばチャレンジについて 5

2.1 つくばチャレンジの概要 5

2.2 今年の課題 6

第 3 章 システムの全体像 9

3.1 ロボット構成 9

3.2 機械設計 10

3.3 ソフト 20

第 4 章 単眼カメラによる三次元測量 23

4.1 カメラの幾何学 23

4.2 三角測量の原理 23

4.3 単眼カメラによる三角測量 24

4.4 射影変換行列の算出 25

4.5 移動量の取得 28

第 5 章 マップ生成 30

5.1 マップ生成環境 30

5.2 マップ生成 31

5.3 移動量のばらつき 31

第 6 章 結言 33

6.1 本研究のまとめ 33

6.2 今後の課題 33

参考文献 35

謝辞 36

付録 37

.1 OpenCV のインストール 37

第 1 章

緒言

1.1 研究の背景

1.1.1 安定性の改善

前年度, つくばチャレンジで使用したロボットの重心が高かったため, 凸凹道やコンクリートなどでロボットが傾きそのまま転倒する恐れがあった. そのため, 前年度の問題点を元に転倒しても走行可能なロボットの開発を試みた. だが, 今年度のつくばチャレンジに参加するにあたり, ロボットの高さ制限を満たし, 転倒しても走行可能な機構の開発ができなかった. よって今回はロボットの走破性を高めることで, 転倒しにくいロボットの開発を行った.

1.1.2 自己位置の推定

ロボットが自律走行を行うためには, 自身の位置を推定する必要がある. その手段としてローカルマップをグローバルマップに変換する方法が主流である. これはロボットが何らかの方法で周辺の情報を得る必要があり, 一般的に測域センサや地磁気センサが用いられることが多い. また, 近年はカメラを用いた信号認識など視覚情報を用いた研究が活発である. そこで今回は低コストで省スペース化が期待できる単眼カメラを用いることにした.

1.1.3 三角測量とカメラの移動量

ローカルマップを生成するために周囲との位置関係を求める必要がある.カメラでは対象物を写した2枚の画像とそれらの画像間距離から三角測量の原理を用いることで対象物からカメラまでの距離を計測することができる.この原理を用いるためには,カメラの姿勢と移動量が既知の値の必要があり,それらを求めなければならない.

1.2 研究の目的

今年度は走破性の向上を考え,前年度の改善案として低重心化とそれを補佐する懸架装置に注目し,”超低重心6輪独立懸架ローバー”を開発する.またカメラの移動量をモーションセンサから得られる位置情報によって取得し,三角測量を行う.これを複数回行い,得られたデータをマップとする.

1.3 論文の構成

1章では研究背景を中心に本論文で述べる内容を簡略化して示す.2章では我々が参加したつくばチャレンジについて概要を述べる.3章ではつくばチャレンジに参加したロボットの概要について示す.4章では3次元測量を行う実際の手順について述べる.5章では4章で述べた手順でマップを作成した結果について述べる.最後に6章で本論文のまとめを述べる.

第 2 章

つくばチャレンジについて

2.1 つくばチャレンジの概要

今年度参加したつくばチャレンジについて説明する. つくばチャレンジとは茨城県南部に位置するつくば市の屋外で行こなわれるロボット実験である. これは人間とロボットの共存する社会の実現のための先端的技術への挑戦が狙いである. 人間とロボットの共存を目的とするため, ロボットには以下のことが求められる.

1) そこにいる人間が安全であること
2) そこにいる人間に危害を加えないこと. またそのように取られる不安感を与えないこと
3) 環境内に存在するものに手を加えないこと. また環境景観を損なわない外観であること
4) 環境内に存在するものの邪魔にならないこと. また, 環境内の運動を妨害しないこと

これらの項目が満たされていなければつくばチャレンジでは走行を行うことができない. また, つくばチャレンジではその走行に対して課題が与えられる. 以下につくばチャレンジのコースを図 2.1 として示す (つくばチャレンジ 2016 の課題より参照).

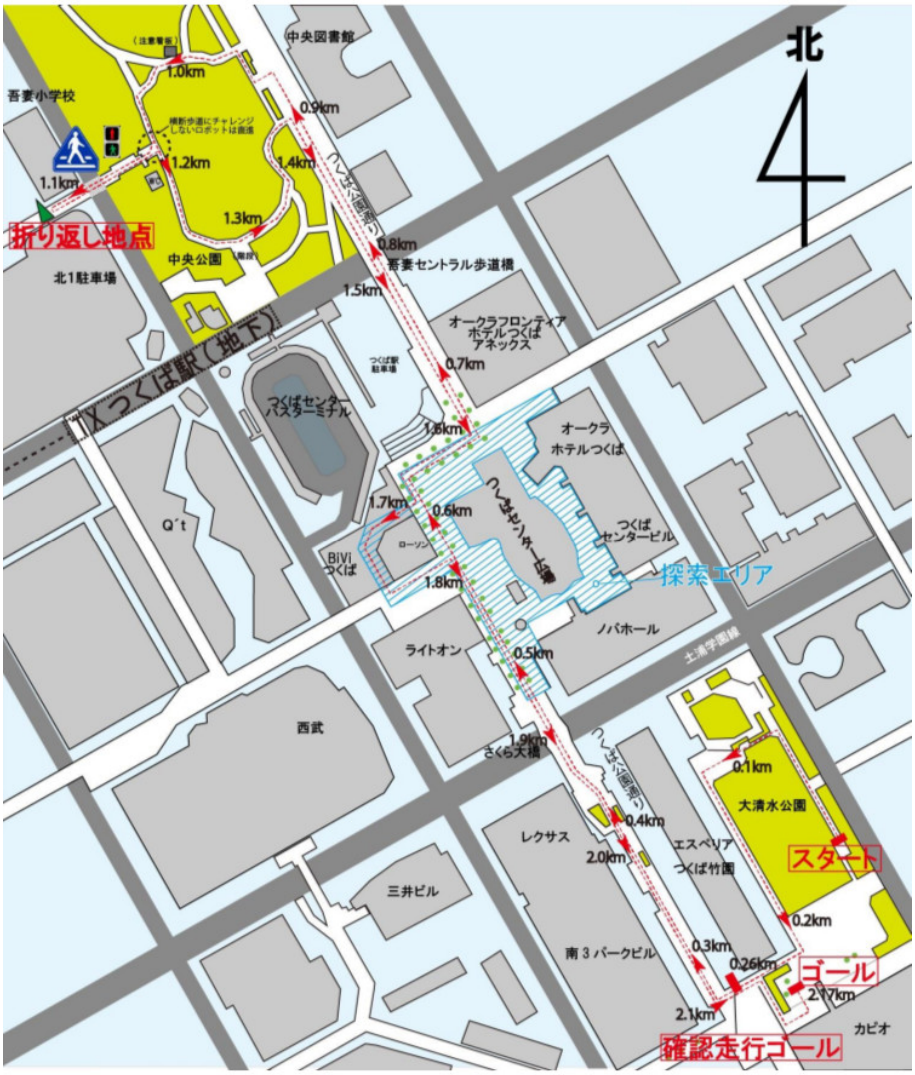

図 2.1　つくばチャレンジコース図

2.2　今年の課題

今年度のつくばチャレンジの課題について説明する. 全体のコース, コース環境, そしてロボットに必要な機能をそれぞれ述べる. またロボットは歩道橋等を走行するが, その場合原則として端より 1m 以上離れての走行とされる.

2.2.1 全体のコース

本コースのスタート地点は図 2.1 の右下に位置するつくば市大清水公園東側の歩道をスタート点とし, 公園内を反時計回りに周回する. コースは大清水公園から遊歩道へと進み, 更に北上してつくばセンター広場へと出る. この広場の東側を探索エリアと呼ぶ. センター広場から北上し, 途中左折して中央公園に入り, 公園内を反時計回りに周回する. 中央公園内の周回途中で, 一旦右折して信号機付の横断歩道を渡る. そして北 1 駐車場脇で折り返し, 再度横断歩道を渡り中央公園に戻る. 中央公園を周回したあと, 再度つくば公園通りの遊歩道に戻り, つくばセンター広場に向かって南下する. センター広場では, 西側にある BiVi つくば (つくばバスターミナルビル) の 2 階の自動扉を通り, その中を通り抜ける. そのまま遊歩道を南下大清水公園に戻る. 大清水公園入口のつくばカピオ前がゴールである.

2.2.2 コース環境

本大会のコースはつくば市民が日頃生活をしている生活環境である. したがって日常のように人や自転車が通行している. つくばチャレンジでは危険防止のため安全責任者を各チーム内から定め, ロボットの走行時には通行人や見物人に注意を呼びかける.

ロボットの走行エリアと市民の歩く領域は区別されない. 天候は当日の天候状況によるが, 小雨程度であれば滞り無く開催される. また路面や環境の状況 (落ち葉, 水たまりなど) も前日やそれ以前の天候, または社会的条件 (お祭りなど) に大きく影響を受ける.

ロボットは環境内にもともと存在する街路樹や縁石柵あるいは建物を走行のガイドとして用いるのは自由であるが, ロボットの走行の為に新しくガイドとなるものを設置することや, 環境を改変することは許されない.

2.2.3 ロボットに必要な機能

つくばチャレンジは人間が暮らす環境で行われる. 人間は多少の凸凹や路面のひび割れ, フレーチングの溝などの障害は問題なく歩行できる. だが, ロボットは人間のようにフレキシブルではない, 段差で転んだり溝にハマることが大いに考えられる. よって走行の際に必要なロボットの機能として悪路が走行できる, 溝に嵌っても自力で這い上がることができる等路面走行に対してフレキシブルに対応することが求められる.

第 3 章

システムの全体像

3.1 ロボット構成

3.1.1 ロボットの構成

今年度のロボットに搭載したセンサを以下の表 3.5 に示す.

表 3.1　ハードウェアの構成

構成要素	メーカ・型番・スペックなど
Mian PC	NVIDIA JETSON TK1
OS	Ubuntu 14.04 LTS
camera	Panasonic HX-A1H
Buttery	KyPOM KT5100 4S 35C
Motor	MABUCHI MOTOR RS-555VC-5524
Motion Sensor	東京航空計器株式会社　CSM-MG100

3.1.2 ロボットの外観

以下の図 3.1 にロボットの外観を示す. また, 車体の仕様を表 3.2 に示す

3.1.3 システムの構成

ロボットのシステム構成を図 3.2 に示す.NVIDIA JETSON TK1 を USB ハブに接続し, そこから各センサ類に接続する. 得られたデータは USB ハブを介して NVIDIA JETSON TK1 に送信され, モータ類に指令を与える.

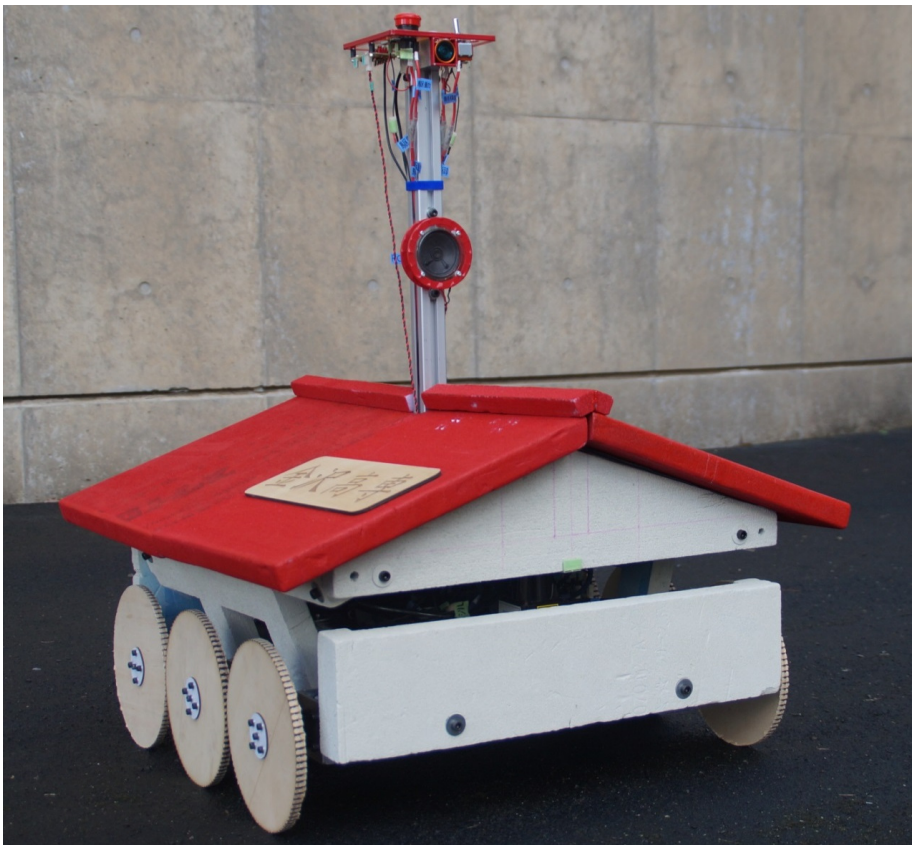

図 3.1　ロボットの外観図

表 3.2　車体の仕様

全高	72[cm]
全長	58[cm]
車幅	61[cm]
重量	20.6[kg]
駆動形式	6 車輪 2 輪駆動
タイヤ径	直径 19[cm]
使用モータ	MABUCHI MOTOR RS-555VC-5524
最大速度	4[km/h]

3.2　機械設計

3.2.1　ハードウェア開発の狙い

サスペンションを内蔵するハードウェアを制作する. つくばチャレンジ 2016 では走破性向上を目的として低重心を目標としているため, ハー

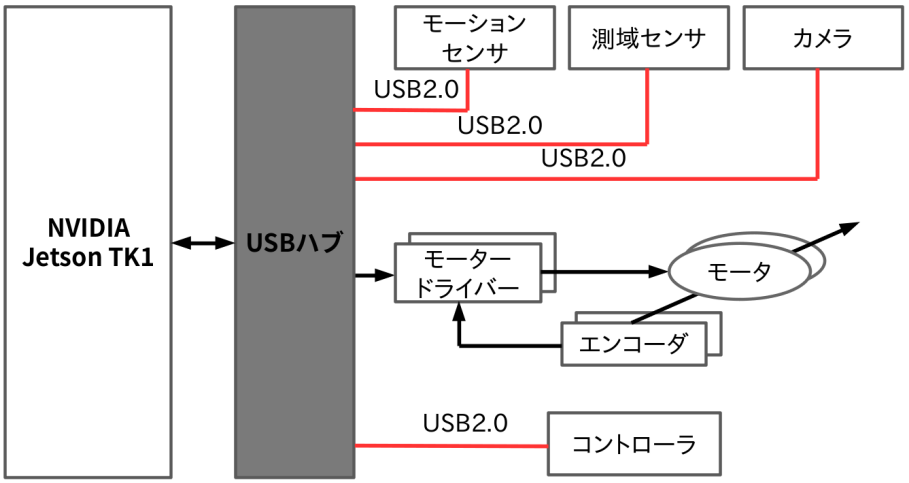

図 3.2　システム構成

ドウェアの全高を小さく設計することでその目標に尽くす. また今回のロボットの超低重心を実現するための数値を表 3.3 に示す

表 3.3　目標数値

横幅	600mm 以内
長さ	1000mm 以内
高さ	150mm 以内
乗り越える段差	およそ 30mm

3.2.2　サスペンションの概要

製作するロボットはある程度の段差を超えるための上下できる機構が必要である. よって近年最も身近な上下機構を持つ機械として, 車に利用されるサスペンションを参考にした. 図 3.3 に示すダブルウィッシュボーン式サスペンションは上下のアームとショックアブソーバ, ばねからなるもので, レーシングカーに多く使用されている. 図 3.4 に示すストラット式サスペンションと呼ばれるものはショックアブソーバとサスペンションを一体化したものをアッパーアームにしたもので, 近年の車やミニ四駆などに使用されている. 両方とも独立懸架で安定

性や利便性に置いてはストラット式が大きく優るが, 今回はコンパクトに構成することも必要なため, もともと上下アームで囲われているスペースを活かせることができそうなダブルウィッシュボーン式サスペンションを基礎として開発する.

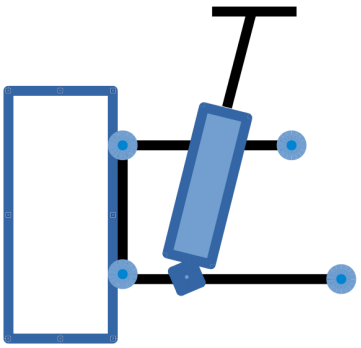

図 3.3　ダブルウィッシュボーン式サスペンション

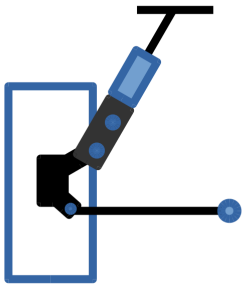

図 3.4　ストラット式サスペンション

3.2.3　メカの概要

製作する”超低重心6輪独立懸架ローバー”(以下6輪ローバ)は図3.5に示すとおり低重心であるがゆえにロボットそのものが平たく大きい物になる. そのままではメンテナンス性にかける他, 運搬時に非常に不便なため, モータを持つドライブモジュール(以下Dモジュール)と指示塔であるコントロールモジュール(以下Cモジュール)に分解できるようにハードウェアを設計し, 分解から組み立てを容易とするようにした. つまり2つのDモジュールと1つのCモジュールからなる1つの

ハードウェアである.

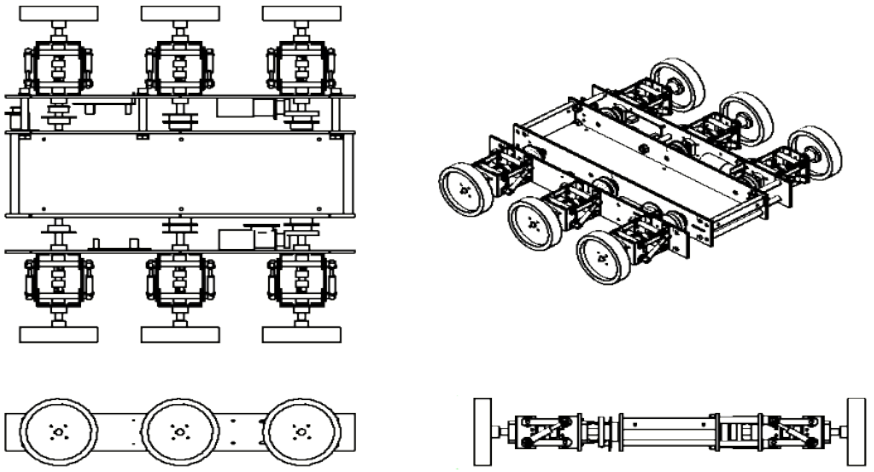

図 3.5　6輪ローバ

3.2.4　サスペンションの配置

ハードウェアの全高は小さくなるため,サスペンションの配置は小スペースに収めることが必要になる.そこで図3.6のように軸棒,ユニバーサルジョイント,車輪を上下のリンクと板を用いて1つの平行リンクをつくり,その平行リンクの両斜辺に2つ直接斜めに振動吸収となるばねを設置すれば車輪の径に左右されるものの従来のものよりスペースが小さく上下機構が得られるのではないかと考えた.

3.2.5　ばねのストロークの算出

図3.7に平行リンクの簡易図を示す.また平行リンクが移動した際の図を図3.8示す.これらの数値よりサスペンションのストロークの長さを求める.サスペンションの全体長さは斜辺に設置するため余弦定理より

$$AC = b^2 = a_1^2 + c^2 - 2a_1 \cdot c \cdot cos\angle B \tag{3.1}$$

$$b = \sqrt{38^2 + 60^2 - 2 \cdot 38 \cdot 60 \cdot cos90} \tag{3.2}$$

$$= 71.022 \qquad 71.0[mm] \tag{3.3}$$

となる.また今回は30[mm]程度の障害物を乗り越える想定しているため,この平行リンクが点A,Bを回転軸として点C,Dが30[mm]上昇した際の平

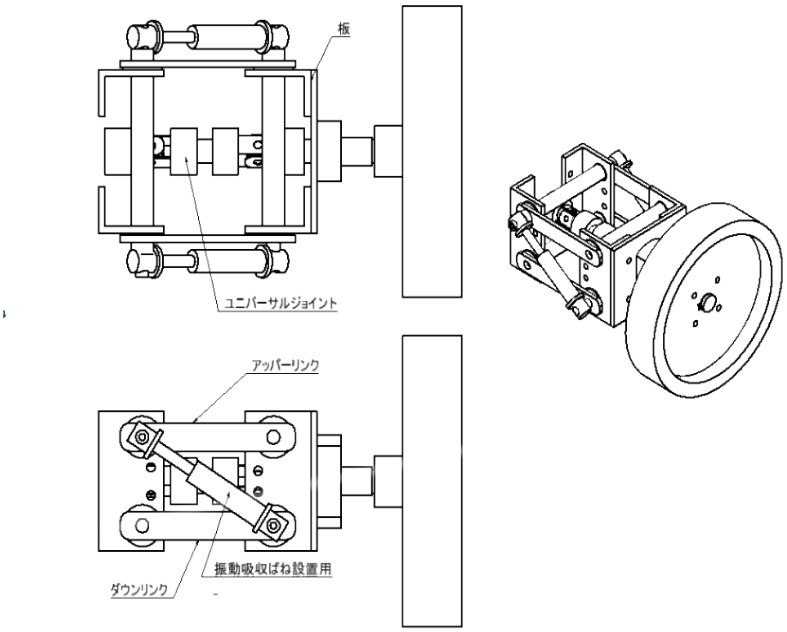

図 3.6　平行リンク

行四辺形の斜辺 AC’ を求める必要がある. 平行リンクが移動してできる Δ AC’E の斜辺 AC’ は Δ AD’E の辺 d’ に等しい. よって AC’ の式は余弦定理と代入法より

$$AE = d' = \sqrt{a_2^2 + e_1^2 - 2a_2 \cdot e \cdot cos\angle D} \tag{3.4}$$

$$d' = c' \tag{3.5}$$

$$AC' = e_2 = \sqrt{c'^2 + a_3^2 - 2 \cdot c' \cdot a_3 \cdot cos\angle E} \tag{3.6}$$

となる. また Δ AD’E の

$$\angle A$$

は 30[mm] 上がった際に 30[deg] になることから, 斜辺 AC’ は

$$d' = \sqrt{30^2 + 60^2 - 2 \cdot 30 \cdot 60 \cdot cos\angle 60} \tag{3.7}$$

$$d' = 30\sqrt{3} = c' \tag{3.8}$$

$$AC' = e_2 = \sqrt{c'^2 + a_3^2 - 2 \cdot c' \cdot a_3 \cdot cos\angle E} \tag{3.9}$$

$$AC' = \sqrt{(30\sqrt{3})^2 + 8^2 - 2 \cdot 30\sqrt{3} \cdot 8 \cdot cos90} \tag{3.10}$$

$$= 52.574 \quad 52.6[mm] \tag{3.11}$$

となる.

サスペンションは伸縮するものであり, それはこれらの辺 AC と辺 AC’ の

差分だけ変化することになる. 今回の値から変化値 x　18[mm] 程度の変化があることが確認された. 以上の AC,AC',x を用いてサスペンションの細かい設計を行う.

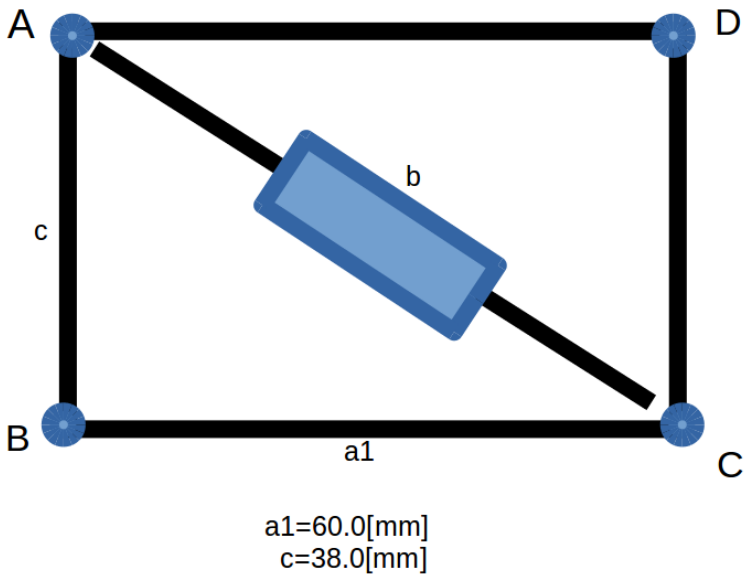

図 3.7　平行リンク簡易図

3.2.6　ばね定数の算出

サスペンションには路面の凹凸を車体に伝えない緩衝装置としての機能としてばねが必要であるため, フックの法則よりばねを選定する. 平行リンクの点 A,B をそれぞれ固定支点と考えて計算を行う. 図 3.9 に Free Body Diagram(FBD) を示す. 図 3.9 よりスラスト方向荷重 x, ラジアル方向荷重 y, モーメント M の式はそれぞれ

$$x = R_A X + R_B X = 0 \tag{3.12}$$
$$y = R_A Y + R_B Y + F = 0 \tag{3.13}$$
$$M = BC \cdot F - AB \cdot R_A X = 0 \tag{3.14}$$

となる. 更に点 A,B,C それぞれの力は以下のようになる.

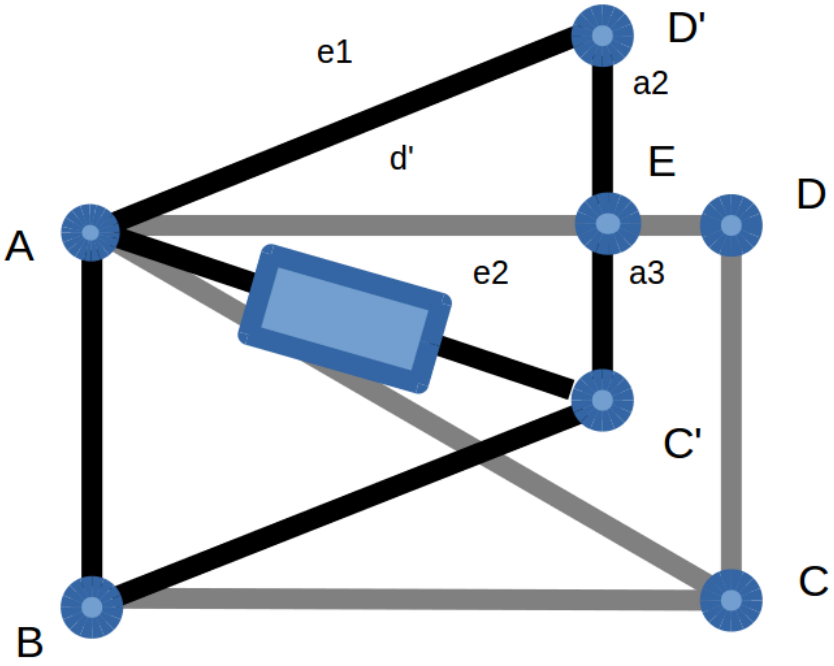

図 3.8　平行リンクの移動量変化

点 A での力学計算

点 A では図 3.10 より以下のように計算できる.

$$R_A X + FAC \cdot cos\ \theta = 0 \tag{3.15}$$

$$R_A Y = FAC \cdot sin\ \theta \tag{3.16}$$

点 B での力学計算

点 B は図 3.10 より以下のように計算できる.

$$R_B X + FBC = 0 \tag{3.17}$$

$$R_B Y = 0 \tag{3.18}$$

点 C での力学計算

点 C は図 3.10 より以下のように計算できる.

$$FAC \cdot cos\ \theta + FBC = 0 \tag{3.19}$$

$$F + FAC \cdot sin\ \theta = 0 \tag{3.20}$$

上記で求めた式より,AC にかかる力 FAC の式を計算する. FAC は式 (14) と式 (18) より

$$R_A Y + F = 0 \tag{3.21}$$
$$F = -R_A Y \tag{3.22}$$

$$R_A Y = FAC \cdot sin\ \theta \tag{3.23}$$
$$FAC = \frac{F}{sin\ \theta}[N] \tag{3.24}$$

$$F + FAD \cdot sin\ \theta = 0 \tag{3.25}$$
$$FAC = \frac{F}{sin\ \theta}[N] \tag{3.26}$$

となることが確認できる.

力 FAC[N] はばね全体に掛る力なので, これをばね 1 個辺りに加わる力に除算する必要がある. 今回作成する 6 輪ローバは 1 輪に 2 つのばねを持つため, ばねを合計で 12 個保有する. よってばね 1 個にかかる力 FAC[N] は

$$FAC = \frac{F}{sin\ \theta} \cdot \frac{1}{12}[N] \tag{3.27}$$

となる. 今回は重量が 15kg になると想定して, ばね定数を求めると

$$FAC = \frac{F}{sin\ \theta} \cdot \frac{1}{12} \tag{3.28}$$
$$FAC = \frac{15 \cdot 9.81}{sin30} \cdot \frac{1}{12}[N] \tag{3.29}$$
$$= 24.525[N] \tag{3.30}$$

これより

$$k = \frac{F}{x} \tag{3.31}$$
$$k = \frac{24.525}{18} \tag{3.32}$$
$$= 1.3625[N/mm] \tag{3.33}$$

となる. よってばね定数は 1.0[N/mm] の物を採用する.

3.2.7　使用したばね定数

上記でばね定数を求めたが, 実際に使用した際に車体が沈むのが確認された. 想定していた 15kg を超えたせいだと考えられる. 車体を沈ませないこと, また段差を超えるため車体を浮かせるために 2,3N/mm の物を用意し, 使用した. 結果的に 2N/mm は沈まなかったがばねが反発しなかったため,3N/mm のばねを使用することとなった.

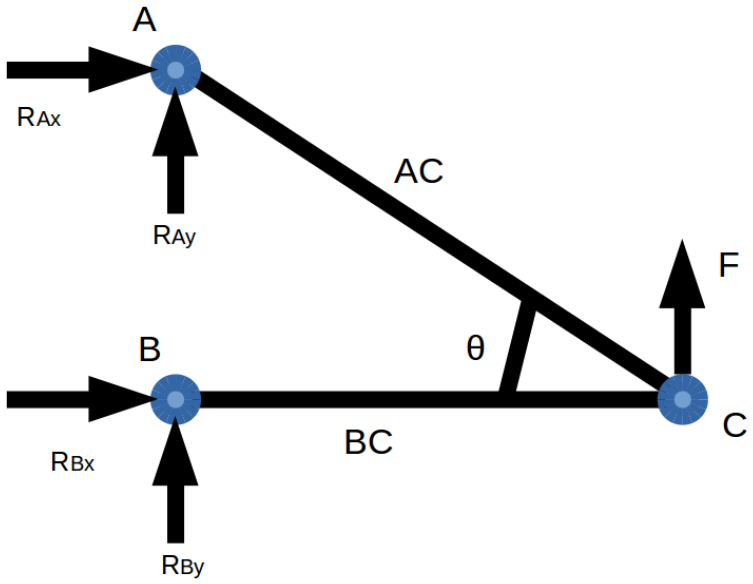

図 3.9　FBD

3.2.8　低重心化の実現

設計した走行モジュールの CAD を図 3.11 に示す. 図から確認できるように全高がタイヤ直径である 190mm 内に収まっているため低重心化が達成された. また 6 輪に付属している懸架装置を図 3.12 に示す. ばねを 2 本設置することでサスペンションとし, またそれらの取り付け位置を斜めにすることでさらに低重心が可能となった.

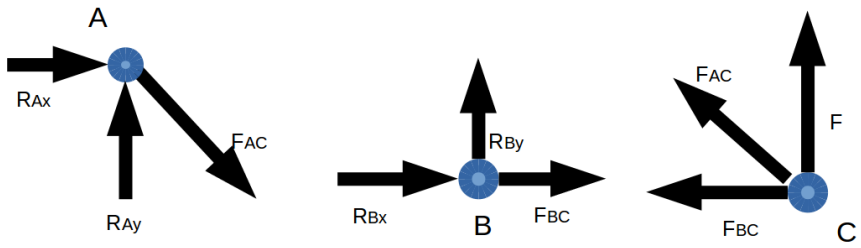

図 3.10　各点にかかる力

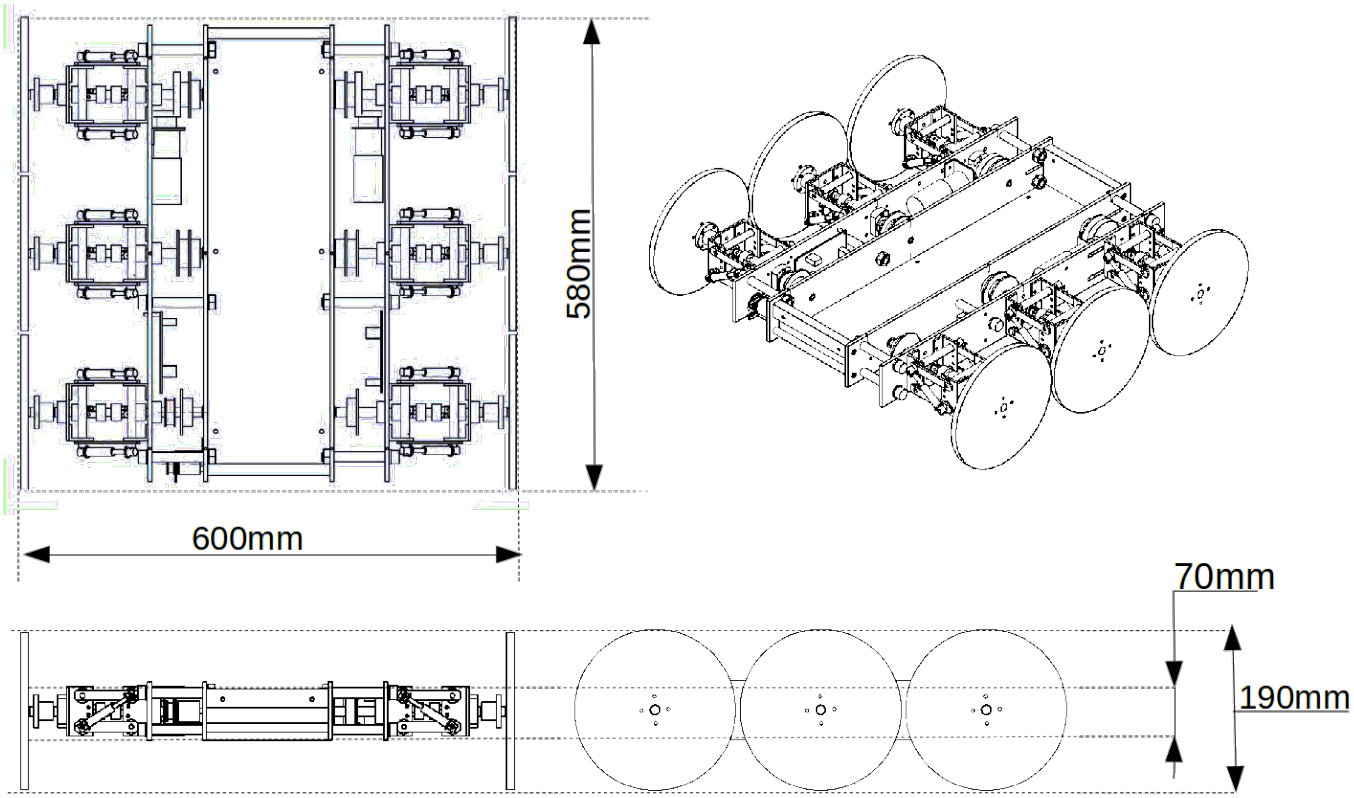

図 3.11　走行モジュール

3.2.9　超堤・超壕能力

今回作成したロボットの性能を調査するため, 超堤・超壕実験を行った結果, 超堤:110[mm], 超壕 220[mm] が得られた.

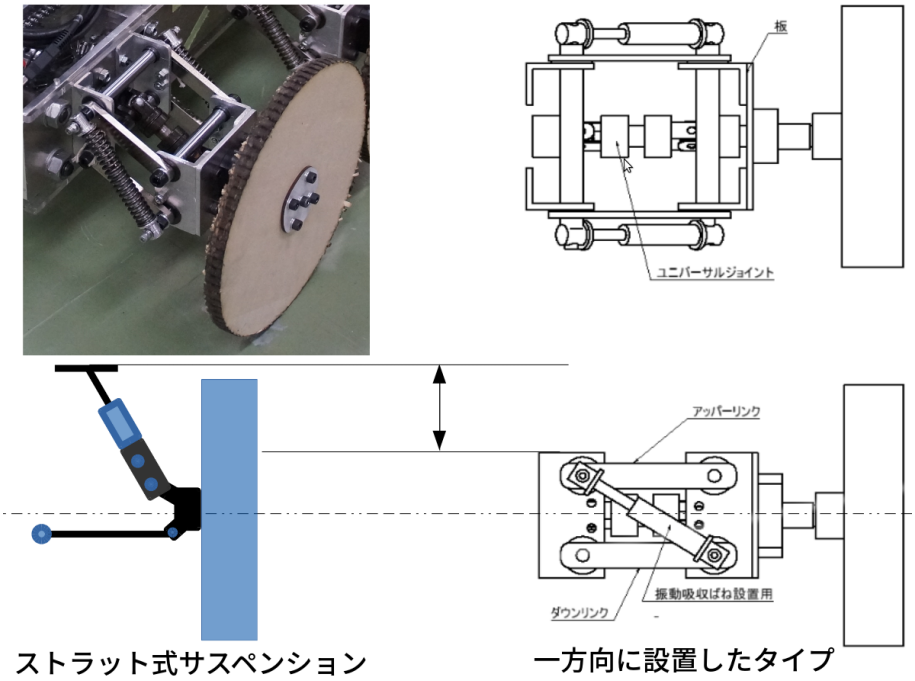

図 3.12　懸架装置

3.2.10　電気設計

ロボットに搭載したモータドライバの回路図を図 3.13 に示し, 使用した部品を表 3.4 に示す. モータドライバは各 D モジュールに 1 つづつの設置とした.

3.3　ソフト

3.3.1　使用ソフトウェア

ロボットのソフトウェアには Ubuntu 14.04 と GNU Compiler を用いる. また使用言語は C++ 言語である.

3.3.2　使用したライブラリ

表 3.5 に使用したライブラリを示す. OpenCV は画像処理・画像解析および機械学習等の機能を持つ C/C++,Java,Python,MATLAB 用ライブラリであり, 後述するマップ作成の主に画像処理に用いる. Point Cloud Libray とは 2,3 次

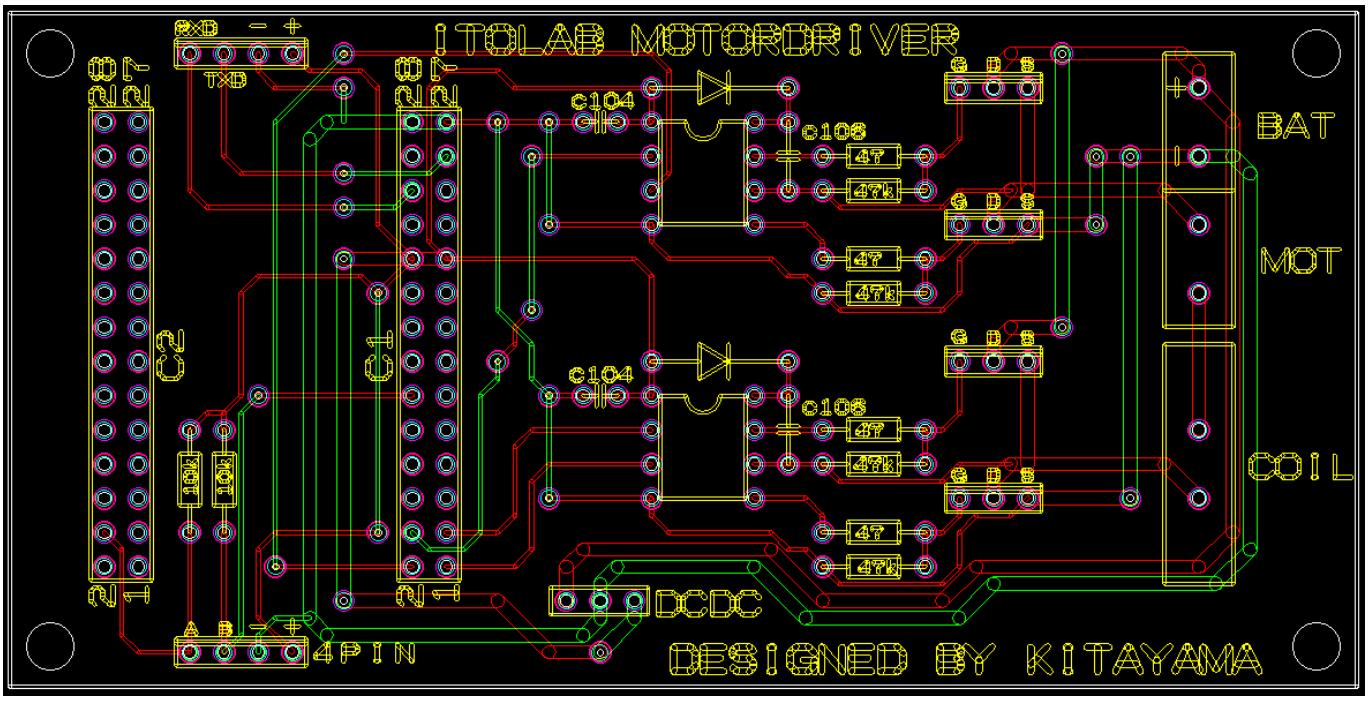

図 3.13　モータドライバ

表 3.4　使用した電気部品

品名	型番	メーカ	単価
自作 MD 板			
トロイダルコイル 330 μ H9A	秋月電子		
EI コネクタ 4 ピンオス			
EI コネクタ 2 ピンオス			
FET			
カーボン抵抗 1/4 W 47k Ω （100 本入）	R-25473		
カーボン抵抗 1/4 W 47 Ω （100 本入）	R-25470		
カーボン抵抗 1/4 W 10k Ω （100 本入）	R-25103		
セラミックコンデンサ 103			
セラミックコンデンサ 105			
発光ダイオード			
RX220 マイコンボード	K-08769		

元の点群からフィルタリング, 特徴推定, サーフェス再構成, レジストレーション（位置合わせ）, モデルフィッティング, セグメンテーション

などのアルゴリズムが使用できる. これは OpenCV から得られた点情報を画面上に表示するために使用する. Urg library とはレーザ距離センサ URG シリーズ用のオープンソースライブラリである. 今回は障害物回避のため用いる.

表 3.5　使用したライブラリ

ライブラリ	使用用途
OpenCV 3.00	画像処理
Point Cloud Library 1.7	点群管理
urg library 1.2.0	障害物回避

第 4 章

単眼カメラによる三次元測量

4.1 カメラの幾何学

カメラの姿勢を並進・回転の同次変換行列で表したものを外部パラメータ行列Eという. また, カメラの焦点距離と画像中心座標をピクセル単位で4×4行列の形に表したものを内部パラメータ行列Kと呼ぶ. さらに, 三角測量を行う際に用いる情報をまとめた行列を射影変換行列Hと呼びこれらの関係を以下に示す. ただし, この時の焦点距離と中心座標は (fx,fy),(cx,cy) とする.

$$K = \begin{pmatrix} fx & 0 & cx & 0 \\ 0 & fy & cy & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

$$E = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

$$H = KE \tag{4.3}$$

これらの関係から射影変換行列Hを求める.

4.2 三角測量の原理

三角測量とは座標が既知の2点とそれらの点間距離, それぞれの点と対象物との角度から正弦定理を用いて対象物の座標を得るものであ

る.カメラでは対象物を写した2枚の画像と射影変換行列から三角測量を行うことができる.以下にその原理を図4.2として示す.図より三角測量の原理を用いて対象物の座標は以下の式で求めることができる.

$$X = \frac{L(X_l + X_r)}{2(X_l - X_r)} \tag{4.4}$$

$$Y = \frac{L(Y_l + Y_r)}{2(X_l - X_r)} \tag{4.5}$$

$$Z = \frac{fL}{X_l - X_r} \tag{4.6}$$

よって射影変換行列とカメラ間の距離を求めることで対象物の座標がわかる.

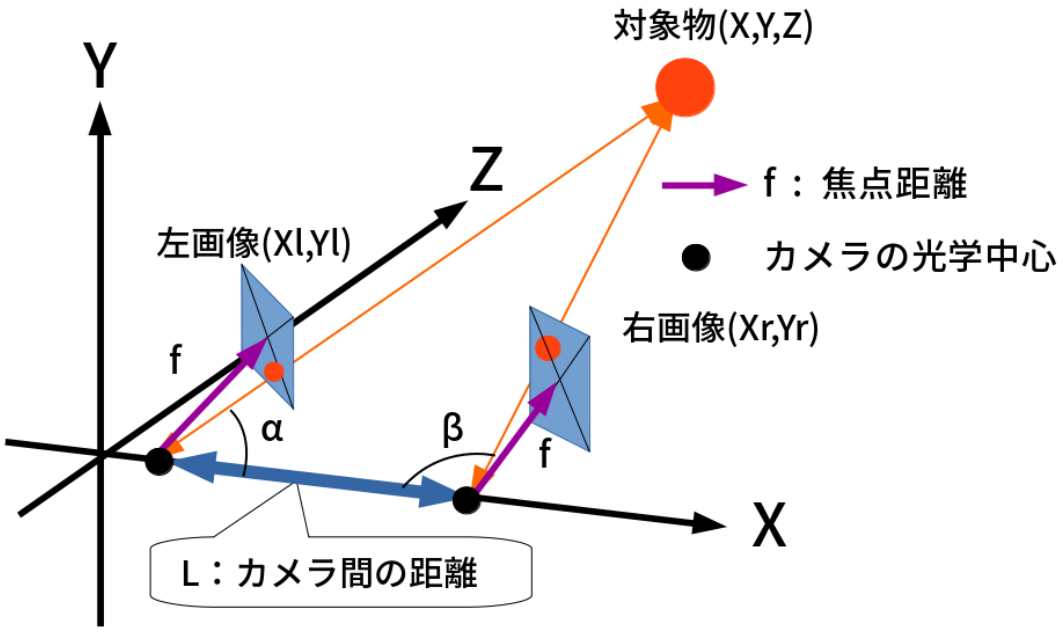

図 4.1　三角測量の原理

4.3　単眼カメラによる三角測量

図4.2のように三角測量を行う際には2つのカメラを用いるのが一般的だ.だが今回はカメラを移動させ過去の位置で取得した画像を利用することで単眼カメラで三角測量を行えるようにした.この原理を図4.1に示す.図4.1より,カメラ間の距離はカメラの移動量に等しい.また,カメラが定点でなく移動するため外部パラメータ行列が変化する.この

ことから単眼カメラで三角測量を行うためには射影変換行列とカメラの移動量をリアルタイムで得る必要がある.

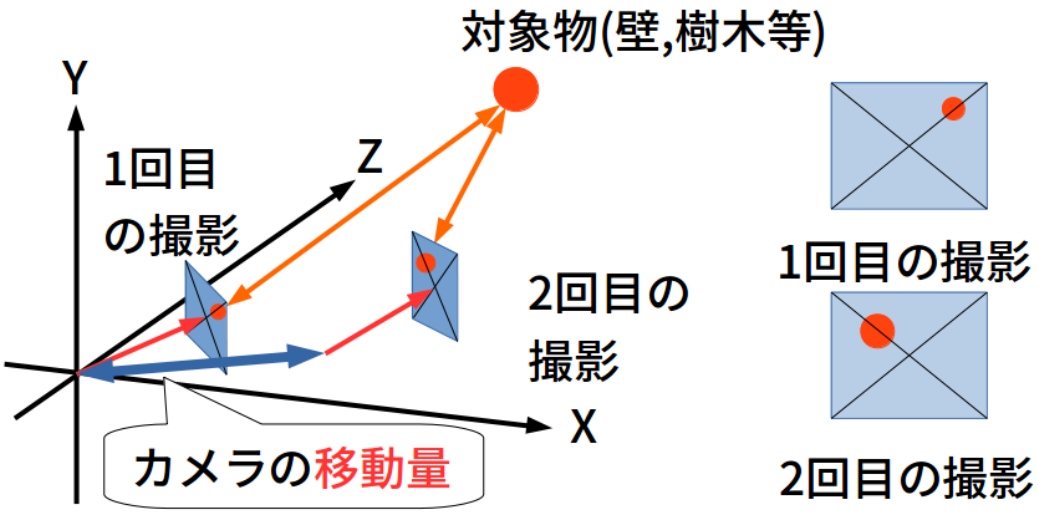

図 4.2　単眼カメラによる三角測量

4.4　射影変換行列の算出

4.4.1　内部パラメータ行列の算出

OpenCV で内部パラメータ行列を求めることができる関数が用意されている. この関数は 8 点法 [3] を用いており, 同じカメラで撮影した 2 枚の画像の対応付けられた点の座標を与えることによって内部パラメータ行列を算出することができる. 今回はチェッカーボード画像を用いて, 2 画像上のチェッカーボードのコーナー座標を検出しそれらを対応付けられた点座標とした.

4.4.2　外部パラメータ行列の算出

OpenCV では外部パラメータ行列を求めることができる関数が用意されている. この関数は 5 点法 [3] を用いており, 内部パラメータ行列,2 枚の画像の対応付けられた点の座標を与えることによって外部パラメータ行列を算出することができる.

4.4.3 特徴点抽出

座標が既知の2点を算出するためにそれぞれ2枚の画像から特徴的な点を抽出する必要がある. 本論文ではOpenCVで使用可能な既存のアルゴリズムを用いて特徴点を抽出する.OpenCVで使用可能なアルゴリズムには様々な種類がありそれぞれ異なった特徴を持っている. 以下に代表的なアルゴリズムとその特徴をまとめ表4.1として示す. これらのアルゴリズムは基本的に画像内の明暗から角(エッジ)を検出している. そのため色情報を用いないことが多く, 計算量削減のためモノクロ画像が用いられることが多い.

表 4.1　代表的な特徴点抽出アルゴリズム

アルゴリズム	備考
SIFT,SURF	照明変化に頑健で回転, 拡大縮小に不変であるが計算量が多い.SURF は SIFT の改良版
ORB	局所性鋭敏型ハッシュという手法を用いており, 計算量が少ない
FAST	回転に不変である.
KAZE,AKAZE	SIFT に比べ, ロバスト性の向上と計算量削減を重視 AKAZE は KAZE の改良版

今回は外で使用するため照明変化に頑健なアルゴリズムのうち計算量を削減したSURFを使用する.

4.4.4 特徴点の対応付け

1枚目の画像から抽出した特徴点が2枚目の画像のどこに対応しているのかを知るためにk近傍法(k nearest neighbor)を使用する. 1枚目の画像のある特徴点を

$$\vec{a} = (a_1, a_2 \cdots, a_n) \tag{4.7}$$

2枚目の画像のある特徴点を

$$\vec{b} = (b_1, b_2 \cdots, b_n) \tag{4.8}$$

図 4.3　特徴点の抽出

とすると以下のように各特徴点の成分を差分し三平方の定理を用いることで対応している点の候補を k 個 (k は任意) 算出する.

$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 \cdots + (a_n - b_n)^2} \tag{4.9}$$

この手法を用いて 2 枚の蝶が写っている画像を対応付けし, 描画した画像を図 4.4 に示す.

図 4.4　特徴点の対応付け

図 4.4 のように, 座標が既知の 2 点を多数算出することができた.

4.4.5 射影変換行列の算出

前述のように2枚の画像の対応付けられた点座標と内部パラメータ行列を得ることができた. これより前述のOpenCVの関数を用いることで外部パラメータ行列を算出することができる. よって, 式4.3を用いて射影変換行列を求めることができる

4.5 移動量の取得

カメラの移動量を取得するため, 東京航空計器社製のモーションセンサから得られる位置情報を使用する. モーションセンサーを図4.5に示す.1回目に取得した位置と2回目に取得した位置の差から移動量を算出する.

図 4.5 モーションセンサー

4.5.1 座標変換

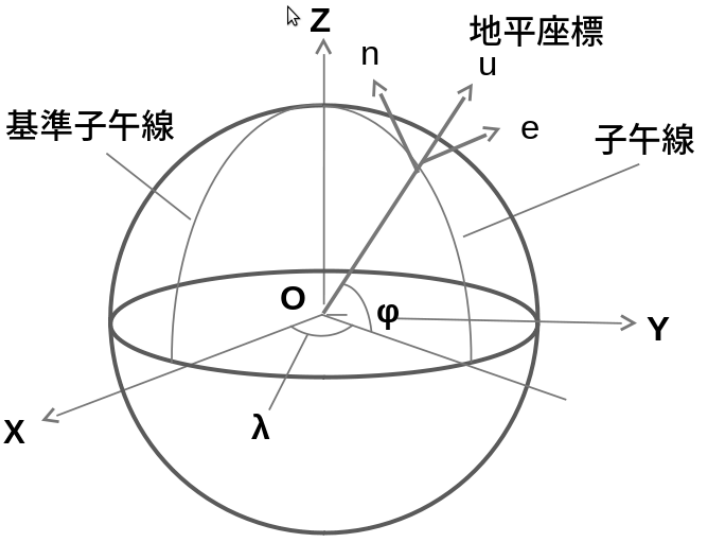

図 4.6　地平座標への変換

第 5 章

マップ生成

5.1 マップ生成環境

マップの生成環境を図 5.1 に示す. 天候はカメラ画像に影響の少ないくもりの日, 画像間距離を 1.4[m] として金沢工業高等専門学校の駐車場付近でマップ生成を行った. ロボットは Start 地点から矢印の方向に進み,2 回の旋回を経て Goal 地点まで走行する.

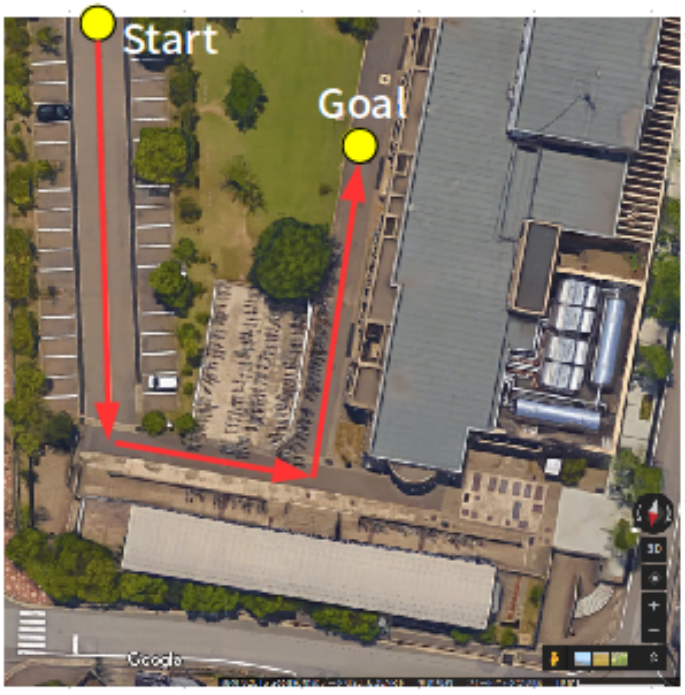

図 5.1 マップ生成環境

5.2 マップ生成

生成されたマップを図 5.2 に示す. 赤い点群はカメラ画像から三角測量によって得られた周囲の位置関係, 青い線はカメラ画像から得られたロボットの走行軌跡, 黄色い線を実際にロボットが走行した軌跡である. 図より, カメラ画像からマップを生成することができた. しかしカメラ画像から得られたロボットの走行軌跡は 1 回目の旋回時から, 実際にロボットが走行した軌跡と比べて大きく歪んでいることが確認できる.

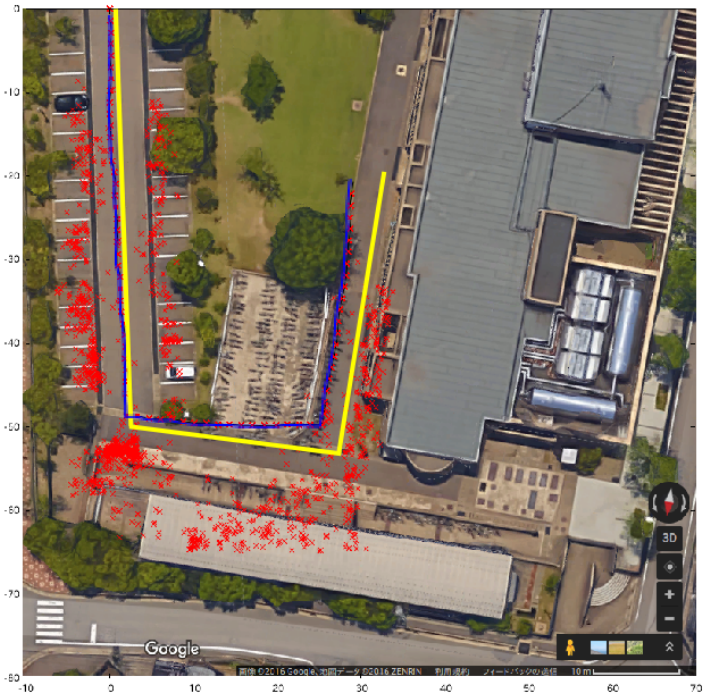

図 5.2　生成したマップ

5.3 移動量のばらつき

マップ生成を行った際にモーションセンサーから得られた移動量のばらつきを図 5.3 に示す.

図を見て分かるとおり, 駐車場側では取得した移動量が真値から増減しているためマップ生成をした際のズレが小さい. しかし建物側に近づくに連れて取得した移動量が真値よりも大きく増加してしまってい

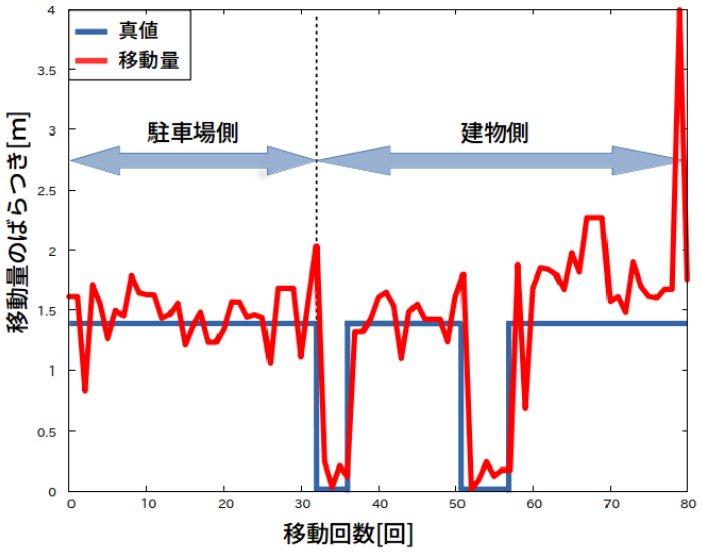

図 5.3　移動量のばらつき

る. また旋回時は移動を行っていないため移動量は 0[m] になっている必要があるのに対し, 取得した移動量は 0[m] になっておらず, これらの原因によってマップ生成の際にズレが発生してしまったと考えられる.

第 6 章

結言

6.1 本研究のまとめ

走破性の高いロボットの開発の実現にあたり, つくば市内での走行が問題なく行えたため, 走破性の高いロボットの開発は達成したと考える. また移動量の取得とマップ生成の実現にあたり, 生成したマップは旋回時に大きく歪むことが確認されたが直線時には問題がなかったため 3 次元復元技術の基礎は確立できたと考える.

6.2 今後の課題

6.2.1 自己位置推定

現在作成できたマップには旋回時に大きく歪むという欠点がある. そのため旋回時には他のセンサーからの情報で外部パラメータ行列に補正をかける必要がある. そこで, 今回はタイヤの滑りが原因で使用できなかったがエンコーダなどを使用する方法が考えられる. また, マップ作成時に使用した画像から一定間隔でテンプレート画像を抽出しそれと現在の画像をテンプレートマッチングできれば, 対応したテンプレート画像の位置から自己位置を推定できる. この場合, 現在の画像とテンプレート画像は鳥瞰図のように同じ視点からの画像になるように透視変換する必要があるが, 前後のテンプレート画像に比較対象を絞ることができるため精度が高くなると考えられる. ただし, スタート時

には参照できる前後のテンプレート画像を自力で用意できない問題がある他, テンプレートマッチング自体の計算量が多く, 処理時間がかかってしまうことが考えられる.

6.2.2 タイヤ素材検討

ロボットのタイヤは木材で作成している. その理由は, ロボットに合うタイヤが見つからず, 金沢工業高等専門学校に新しく導入されたレーザー加工機の試験的運用を兼ねて木材で作成した. しかし木材のタイヤでは路上を走行した際に削れる, 滑るなどの問題が発生し, タイヤの交換が必要となってしまう. よって削れる, 滑ることのないタイヤ素材の検討が必要である.

6.2.3 屋内対応

今回我々は移動量を取得するためにGPSから得られる位置情報を使用しているため,GPSを取得することができない屋内などでは移動量を取得することができない. そこで我々が使用したモーションセンサーは位置情報以外にも加速度や姿勢なども取得することができるため, これらのセンサー情報から屋内でも移動量を取得し屋内マップを生成する.

6.2.4 全天候性の付与

現状では, カメラで周囲の環境を撮影した際に天候によっては周囲の環境を上手く写しだすことができない可能性がある. 写り方によってはマップを生成できないため, 写り方が天候によって作用されない手段を考える必要がある.

参考文献

[1] Bradski,Kaehler:詳 解 OpenCV,433, 株 式 会 社 オ ラ イ リ ー・ジ ャ パ ン (2012)

[2] 理 解 す る た め の GPS 測 位 計 算 プ ロ グ ラ ム 入 門,http://www.enri.go.jp/ fks442/K_MUSEN

[3] 初 期 値 推 定 付 非 線 形 最 小 二 乗 法 に よ る 5 点 法 の 解 法

謝辞

本論文作成にあたりテーマの決定, 研究の考え方, 方法のまとめ方など全てにおいて長期にわたって厳しくも熱意のあるご指導, ご鞭撻していただいた, 伊藤恒平教授に厚く御礼申し上げます.

特に分析においても論文の書き方においても論文を何度も読んでいただき, 指導していただいた伊藤恒平教授に大変ご苦労をかけてしまいましたことにも心よりお詫び申し上げたいです.

同級生のメンバーには論文の作成, 修正にご協力いただき心より感謝しております.その他, 助けていただいた多くの皆様に心から感謝しております. ありがとうございました.

付録

付 録

.1 OpenCV のインストール

今 回 は Ubuntu で OpenCV を イ ン ス ト ー ル す る 手 順 を 示 す. 基 本 的 に 以 下 の コ マ ン ド を copy and paste す れ ば イ ン ス ト ー ル が 完 了 す る.

.1.1 必要なファイルをインストールする

- 開 発 ツ ー ル

  sudo apt-get -yV install cmake libeigen3-dev

  ubuntu 16.04 以 降 sudo apt -yV install cmake libeigen3-dev
- GUI フ レ ー ム ワ ー ク 関 連

  sudo apt-get -yV install libgtk2.0-dev libgtkglext1-dev libqt4-dev freeglut3-dev

  ubuntu 16.04 以 降 sudo apt -yV install libgtk2.0-dev libgtkglext1-dev libqt4-dev freeglut3-dev
- 並 列 処 理 関 連

  sudo apt-get -yV install opencl-headers libtbb-dev

  ubuntu 16.04 以 降 sudo apt -yV install opencl-headers libtbb-dev
- 画 像 フ ォ ー マ ッ ト 関 連

  sudo apt-get -yV install libjpeg-dev libjasper-dev libpng++-dev libtiff-dev libopenexr-dev libwebp-dev ubuntu 16.04 以 降 sudo apt -yV install libjpeg-dev libjasper-dev libpng++-dev libtiff-dev libopenexr-dev libwebp-dev
- Python 関 連

  sudo apt-get -yV install libpython3.4-dev python-numpy python-scipy python-matplotlib python3-

numpy python3-scipy python3-matplotlib

ubuntu 16.04 以 降 sudo apt -yV install libpython3.4-dev python-numpy python-scipy python-matplotlib python3-numpy python3-scipy python3-matplotlib

- Qt 関 連

  sudo apt-get -yV install qtbase5-dev ubuntu 16.04 以 降

  sudo apt -yV install qtbase5-dev

- cmake-gui イ ン ス ト ー ル

  sudo apt-get -yV install cmake-gui ubuntu 16.04 以 降

  sudo apt -yV install cmake-gui

.1.2 GitHub から OpenCV をクローンする

git clone https://github.com/Itseez/opencv.git

git clone https://github.com/Itseez/opencv_contrib.git

.1.3 クローンした OpenCV に移動しビルド用のフォルダを作成する

cd opencv

mkdir build

.1.4 CMame-gui を実行する

cmake-gui

.1.5 CMake-gui を以下の手順で操作する

1) Where is the source code:に opencv デ ィ レ ク ト リ ま で の パ ス を 指 定 す る

   例) /home/kouhei/work/opencv

2) Where to build the binarise:に ビ ル ド し た も の を 置 く デ ィ レ ク ト リ を 指 定 す る. 先 に 作 っ た build デ ィ レ ク ト リ を 指 定 す る.

   例) /home/kouhei/work/opencv/build

3) configure ボ タ ン を 押 す

4) OPENCV_EXTRA_MODULES_PATH に opencv_contrib/modules までのパスを指定する.

例) /home/kouhei/work/opencv_contrib/modules

5) Configure ボタンを押す

6) Generate ボタンを押す

7) cmake-gui 終了

.1.6 OpenCV をコンパイル, インストールする

sudo make install

.1.7 共有ライブラリの依存情報を更新して終了

sudo ldconfig