

## 第11-3章 ロボット実験3: 加速度・地磁気センサ

### 11-3.1 はじめに

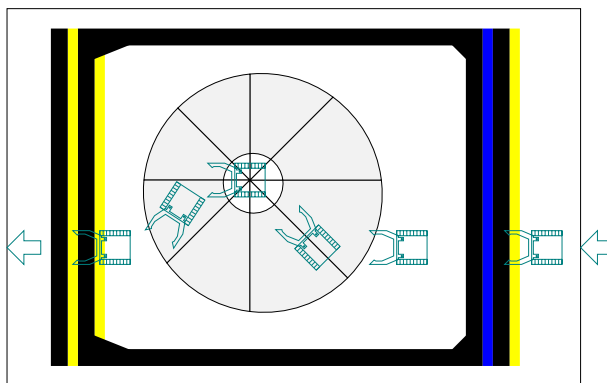
本実験では Zumo に搭載されている加速度・地磁気センサの使い方について学ぶ。加速度・地磁気センサの情報をもとに、前期で学んだ PI 制御を応用して、山登り・山下りや東西南北方向へ向くプログラムを作成する。また、ロボットの姿勢を推定し、推定した姿勢を Processing 上で 3D 描画する。

実験目的:

- 加速度・地磁気センサの値を読み取ることができる。
- 地磁気センサのキャリブレーションができる。
- PI 制御を用いて山登り・山下りなどできる。
- 加速度センサ・地磁気センサの値からロボットの姿勢を推定できる。
- Processing によりロボットの姿勢を 3D 描画できる。

簡易ゾーン課題:

- ゾーンに進入後、(1) 山に登り、(2) 頂上でその場回転 (その際、東西南北で一旦停止)、(3) 下山して、(4) ゾーン出口から退場する。
- 各行程におけるロボットの姿勢などを Processing により 3D 表示する。



ZumoMotors, Pushbutton, LSM303 のライブラリをまだインストールしていない人は、付録 F 節などを参考にインストールしておくこと

本章の構成はつぎのとおりである。11-3.2 節、11-3.3 節に加速度センサ、地磁気センサの概要を述べる。実験手順は 11-3.4 節に、レポート課題は 11-3.5 節に、それぞれ示す。付録 A~付録 E にはロボットの 3D 描画のためのヒント、剛体の回転を表す Euler 角の計算方法などを載せておいた。

## 11-3.2 加速度センサの概要

### 11-3.2.1 加速度とは

加速度とは移動する物体の座標の時間に関する 2 階微分である。物体の座標を  $\vec{x}$  とすると、 $\vec{x}$  の時間に関する 1 階微分  $\vec{v} = \dot{\vec{x}} = \frac{d\vec{x}}{dt}$  は速度と呼ばれ、2 階微分  $\vec{a} = \ddot{\vec{x}} = \frac{d^2\vec{x}}{dt^2}$  は加速度と呼ばれる。ニュートンの第 2 法則 [1] は、“加速度は物体に作用する力に比例する” という法則である:

$$\vec{f} = m\vec{a}$$

ここで、 $\vec{f}$  は物体に作用する力であり、比例係数の  $m$  は質量と呼ばれる。加速している飛行機や乗用車の中に座っていると、後ろから背中を押されているように感じるがあると思う。特に離陸に向けて動き出した飛行機では、速度はゆっくりでも加速度は大きく、速度と加速度は違うということがよくわかる。

加速度センサは、移動する物体に作用する力を計測することによって、加速度を検出している。先ほど、“後ろから背中を押されているように感じる” と述べたが、この力の大きさを計測する。

物体に働く力の代表例として重力がある。重力は質量に比例した大きさで、鉛直下向き（地球の重心方向）に作用する力である。ロボットのように地面の上にある物体は、重力が作用しているからといって下に“落ちて”いくことはなく、地面の上に載ったままである。これは、地面から反作用という重力と同じ大きさの上向きの力が作用しているからである（ニュートンの第 3 法則 [1]）。地面の上に立ったとき、“地面から足の裏を押されているように感じ”たら、それは重力の反作用を知覚しているのである。加速度センサは、この重力の反作用とその他の力を区別することはできないので、あたかも、上向きに加速しているような結果を返す。これを利用すると、重力の反作用を計測することによって、ロボットの傾きを検知することができる（11-3.2.2 項参照）。

加速度センサによって検出するもう一つの例として、衝突検知がある。ロボットが物体に衝突すると、急に停車するので、後ろ向きにパルス状の加速度が発生する。坂道の角度が徐々に急勾配になる場合などとは違って、衝突の場合は加速度が急激に変化するので、高域通過フィルタなどを使うことによって、衝突を検知することができる。高域通過フィルタ (High Pass Filter, HPF) とは、信号の低周波数成分を差し引いて、高周波成分を抽出するフィルタである [2]。信号の低周波数成分を抽出するフィルタは、低域通過フィルタ (Low Pass Filter, LPF) と呼ばれる。また、特定の周波数帯域のみを取り出すフィルタを帯域通過フィルタ (Band Pass Filter, BPF) という。

この他、エンジンによる振動を加速度センサによって検出し、その逆位相の振動をモータによって発生させることによって、車内の振動を抑制し、高級感を醸出するアクティブ防振なども考えられている。

### 11-3.2.2 加速度センサの原理

加速度センサは、基本的には力センサである [5]。バネのたわみを静電容量の変化として取り出すことにより、バネに働く力を計測する。MEMS (Micro Electro-Mechanical Systems) 技術の発達により、半導体の上に、質量・バネ・ダンパ系と質量の変位に応じて容量が変化するコンデンサを集積化した、微小で精度の良い加速度センサの実現が可能となった。地磁気センサなども MEMS デバイスとして実装されている。

LSM303 では  $\vec{X}$ -軸、 $\vec{Y}$ -軸、 $\vec{Z}$ -軸の 3 軸方向の加速度を検出することができる。 $\vec{X}$ 、 $\vec{Y}$ 、 $\vec{Z}$  の各軸はロボットに固定した軸であり、進行方向を  $\vec{X}$ -軸、上方向を  $\vec{Z}$ -軸とする。座標系は右手系なので、 $\vec{Y}$ -軸は左方向である。加速度  $\vec{a}$  (ベクトル) を 3 軸方向に分解し、各値を計測する。

$$\vec{a} = a_X \vec{e}_X + a_Y \vec{e}_Y + a_Z \vec{e}_Z \quad (11-3.1)$$

ここで、 $\vec{e}_X$ 、 $\vec{e}_Y$ 、 $\vec{e}_Z$  は、それぞれ、 $\vec{X}$ 、 $\vec{Y}$ 、 $\vec{Z}$  方向の単位ベクトル、 $a_X$ 、 $a_Y$ 、 $a_Z$  は、それぞれ、加速度の  $\vec{X}$ 、 $\vec{Y}$ 、 $\vec{Z}$  方向成分 (スカラー) である (図 11-3.2.1)。

重力は常に働いているので、ロボットが静止状態、あるいは、等速直線運動している状態では、重力の反作用の  $\vec{X}$ ,  $\vec{Y}$ ,  $\vec{Z}$  方向成分を知ることによって、ロボットの姿勢のうち, roll と pitch がわかる. ここで, **roll 角**  $\phi$  は  $\vec{X}$ -軸まわりの回転角, **pitch 角**  $\theta$  は  $\vec{Y}$ -軸まわりの回転角である. また,  $\vec{Z}$ -軸まわりの回転角を **yaw 角** ( $\psi$ ) という (図 11-3.2.2). それぞれ原点から各座標軸の + 方向に向かって右ねじが進むとき, 右ねじが回る方向が**正の方向**である.  $\vec{X}$ -軸方向の重力の反作用  $a_X$  は,  $-\sin \theta$  に比例した値として検出される.  $\vec{Y}$ -軸方向の重力の反作用  $a_Y$  は,  $\sin \phi$  に比例した値として検出される.  $\pm 90^\circ$  を超える角度は,  $a_Z$  の正負から判断できる. (yaw 角は加速度センサからはわからない. なぜか? cf. yaw は地磁気センサの他, ジャイロセンサや角加速度センサを用いて推定することができる.)

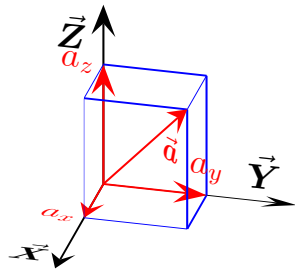


図 11-3.2.1: 3 軸方向への加速度の分解

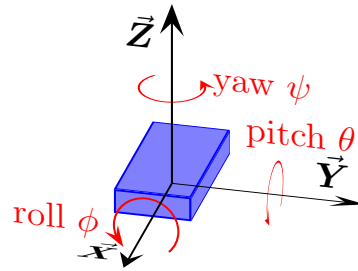


図 11-3.2.2: pitch, roll, yaw

## 11-3.3 地磁気センサの概要

### 11-3.3.1 地磁気センサの原理と方位計測

地磁気とは、地球の北極を S 極、南極を N 極とする磁石によって生じる磁場を意味し、その磁界の強度（磁束密度） $\vec{F}$  は、大きさと方向を持つベクトル量で表される。磁束密度の大きさの MKS 単位はテスラ [T]、CGS 単位はガウス [G] である。（ $1[\text{T}] = 10000[\text{G}]$ ）方向は一般的に使用される角度の単位である度  $[\circ]$  または分  $[']$  が使用される。地磁気の大きさは、およそ  $2 \sim 3 \times 10^{-1}[\text{G}]$  である。

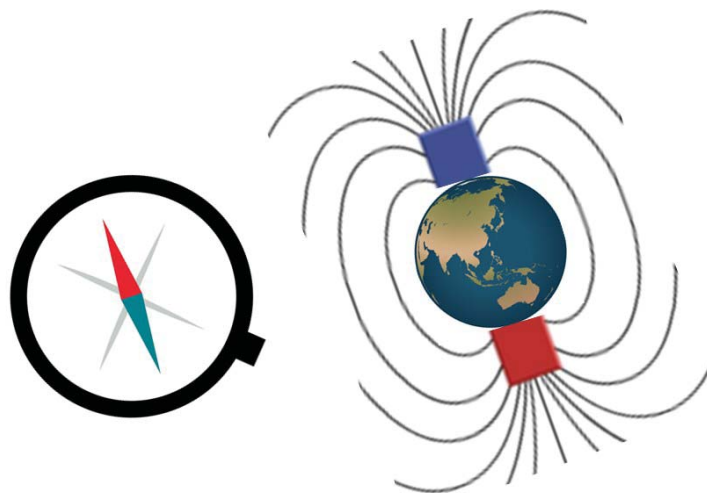


図 11-3.3.3: 地磁気

地磁気センサには、**磁界の強度（磁束密度）の変化に応じて磁気インピーダンスが変わる仕組み**など利用して、磁束量を電気信号に変換するセンサ（磁気センサ）が使用される。LSM303 では  $\vec{X}$  軸、 $\vec{Y}$  軸、 $\vec{Z}$  軸の 3 軸に対する各磁界強度を検出する磁気センサが搭載されており、各軸の磁束密度を算出することができる [4]。各軸の磁束密度は地磁気の  $\vec{X}$  軸成分、 $\vec{Y}$  軸成分、 $\vec{Z}$  軸成分となる。地表の 1 点に固定し、北を  $\vec{x}$  軸、鉛直上方向を  $\vec{z}$  軸とする**グローバル座標系** ( $\vec{x}, \vec{y}, \vec{z}$ ) を考える。（右手系なので  $\vec{y}$  軸は西方向となる。）これに対して、ロボットに固定した座標系 ( $\vec{X}, \vec{Y}, \vec{Z}$ ) を**ローカル座標系**と呼ぶ。ロボットの  $\vec{Z}$  軸がグローバル座標系の  $\vec{z}$  軸と一致しているとき、 $\vec{X}$  軸と  $\vec{Y}$  軸とで生成される面は水平面を成す。地磁気的全磁力を  $\vec{F}$  とすると、 $\vec{F}$  は水平面分力  $\vec{H}$  と  $\vec{Z}$  軸方向成分  $F_z$  に分解され、さらに、水平面分力  $\vec{H}$  は  $\vec{X}$ 、 $\vec{Y}$  軸方向成分  $H_y$ 、 $H_x$  に分解される。ロボットに取り付けられた地磁気センサより各値を計測することができる。このとき、 $\vec{H}$  の方向は磁北を指す。この特性を利用することで、地磁気センサは電子コンパスとして使用され、ロボット搭載時には、ロボットの進行方向の計測に使用されることが少なくない。

簡易版方位計測（ロボットが水平面内にいるとき）

LSM303 では  $\vec{X}$  軸方向を進行方向とするように磁気センサが搭載されている。ロボットが水平面上で操作されるとすると、進行方向の方位は、図 11-3.3.4 に示す水平面 ( $\vec{X}$ - $\vec{Y}$  平面) で検出することができる。水平面分力  $\vec{H}$  の方向は磁北であり、水平面分力と  $\vec{X}$  軸のなす角（偏角） $\theta$  を計算することで進行方向の位が算出される。

$$\theta = \arctan \frac{H_y}{H_x} \quad (11-3.2)$$

ただし、 $H_y$  は  $\vec{F}$  の  $\vec{Y}$  軸成分、 $H_x$  は  $\vec{F}$  の  $\vec{X}$  軸成分である。

地磁気は真北を向いているわけではなく、徳島近辺では約  $7^\circ$  西を向いている（**偏角**  $7^\circ$ ）。また、必ずしも水平面上を通っているわけではなく、徳島近辺では N 極が約  $43^\circ$  下を向いている（**伏角**  $43^\circ$ ）。以下では、北

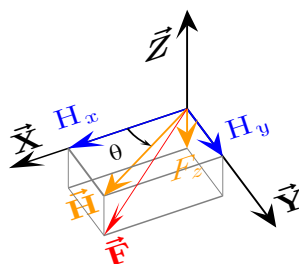


図 11-3.3.4: 地磁気の要素

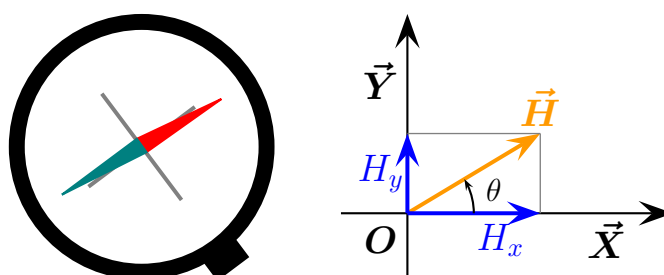


図 11-3.3.5: 方位の計算

と言ったときは真北ではなく、**磁北** (方位磁石が指す北) を表すものとする。ロボットが3次元的な動きをする場合、伏角を考慮しないとロボットの姿勢を正しく認識できない。その場合は、全磁力  $\vec{F}$  の  $\vec{X}$  方向成分 ( $H_x$ )、 $\vec{Y}$  方向成分 ( $H_y$ ) だけでなく、 $\vec{Z}$  方向成分 ( $F_z$ ) も計測しなければならない。詳しくは、付録 C, 付録 D, 付録 E, を参照のこと。

### 11-3.3.2 地磁気センサのキャリブレーション


地磁気センサの出力は磁場の強さに比例する。そのため、センサ感度およびゼロ点 (原点) が分かれば、正しく磁束密度の値に換算することができる。しかながら、地磁気センサは、温度や周囲部品の状況などに依存し、その特性が変わることがある。感度の変動は比較的少ないが、原点は比較的ずれやすい。もし、原点がずれてしまうと正しい方位を算出することができなくなる。そこで、感度および原点を算出し、補正する必要がある。補正することを地磁気センサの**キャリブレーション**という。磁気センサには個体差があり、それぞれのずれが異なるので、各センサに対してキャリブレーションを行う必要がある。


本実験では、ロボットは3次元的に動くので、 $\vec{X}\vec{Y}\vec{Z}$ -軸のキャリブレーションを行う。例えば、初期化 (setup()) のところで、1回地磁気センサの値を読み込み、各軸方向の最大値・最小値の値を現在の地磁気センサの値とする。その後、loop() の中で、地磁気センサの各軸方向成分を読み取ったら、その都度、各方向の最大値と最小値を更新するようしておく。また、Processing を用いて  $\vec{X}\vec{Y}\vec{Z}$  方向成分を逐次表示するようしておく。Zumo ロボット起動後、ロボットを手にとって、ロボットの**前方向・後方向・左方向・右方向・上方向・下方向**をそれぞれ磁北の方角 (偏角約  $7^\circ$ 、伏角約  $43^\circ$ ) へ順次向ける。 $\vec{X}$  軸方向成分 (前後方向) をキャリブレーションするとき、 $\vec{Y}$ 、 $\vec{Z}$  軸方向成分は0となるので、Processing の画面を見ながら  $\vec{Y}$ 、 $\vec{Z}$  軸方向成分が0となるようにロボットを回転させればよい。 $\vec{Y}$ 、 $\vec{Z}$  軸方向成分のキャリブレーションのときも同様。

## 11-3.4 加速度・地磁気センサの演習

### 〈演習 11-3.4.1〉 Processing を用いたロボットの姿勢と方角の表示

1. 加速度センサ・地磁気センサの値を読み取り、加速度センサの値とキャリブレーションされた地磁気センサの値をバイナリで Processing に送る Arduino スケッチ (リスト 11-3.4.1) を作成し、実行せよ。また、Processing で加速度センサ・地磁気センサの値を受け取り、ロボットの法線方向、地磁気の方角、および、ロボットの進行方向に対する北の方角を図示する Processing プログラム (リスト 11-3.4.2) を作成し、実行せよ。シリアルケーブルの断線を防ぐため、通信には XBee を用いよ。プログラムはテキストからカット & ペーストせず manaba からダウンロードせよ。(文字化けのため、非常に見つけにくいエラーが発生する。)
2. 11-3.3.2 項にしたがってキャリブレーションせよ。
3. ロボットを左右に傾ける (roll 角が  $\pm$  に変化する) と  $a_Y$  の値が  $\pm$  に変化することを確認せよ。また、ロボットを前後に傾ける (pitch 角が  $\pm$  に変化する) と  $a_X$  の値が  $\pm$  に変化することを確認せよ。その他、ロボットをいろいろと傾けてみよ。符号の正負に気をつけよ。
4. 地磁気センサの向きが正しく北を指していることを確認せよ。

 Arduino プログラム (リスト 11-3.4.1) 中の `compass.a.x` などは `int(2byte: -32768 ~ 32767)` である。ところが、processing 上では `int` は 4byte なので、`0 ~ 65535` の範囲の整数と認識されてしまう。32768 以上の場合 65536 を引くと正しい値を得ることができる。(リスト 11-3.4.2参照)

 シリアルケーブル (USB ケーブル) を接続したまま、ロボットを動かすと、ケーブルの断線につながる。ロボットを動かすときは、XBee による通信に切り換えよ。

リスト 11-3.4.1: Arduino スケッチ: enshu11-3-4-1

```
#include <Wire.h>
#include <LSM303.h>

LSM303 compass;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  compass.init();
  compass.enableDefault();
  compass.read();
  compass.m_min.x = compass.m.x; compass.m_max.x = compass.m_min.x+1;
  compass.m_min.y = compass.m.y; compass.m_max.y = compass.m_min.y+1;
  compass.m_min.z = compass.m.z; compass.m_max.z = compass.m_min.z+1;
}

void write2byte(int x) {
  Serial.write(x>>8);
  Serial.write(x&255);
}

void loop() {
  float mx,my,mz;

  compass.read();

  Serial.write('H');
  write2byte(compass.a.x); write2byte(compass.a.y); write2byte(compass.a.z);

  mx = compass.m.x; my = compass.m.y; mz = compass.m.z;

  /* Calibration */
  compass.m_min.x=min(compass.m_min.x,mx); compass.m_max.x=max(compass.m_max.x,mx);
  compass.m_min.y=min(compass.m_min.y,my); compass.m_max.y=max(compass.m_max.y,my);
  compass.m_min.z=min(compass.m_min.z,mz); compass.m_max.z=max(compass.m_max.z,mz);
  /* end of Calibration */

  mx = map(mx,compass.m_min.x,compass.m_max.x,-128,127);
  my = map(my,compass.m_min.y,compass.m_max.y,-128,127);
  mz = map(mz,compass.m_min.z,compass.m_max.z,-128,127);

  write2byte((int)mx); write2byte((int)my); write2byte((int)mz);

  delay(100);
}
```

リスト 11-3.4.2: Processing スケッチ: accelerometer

```

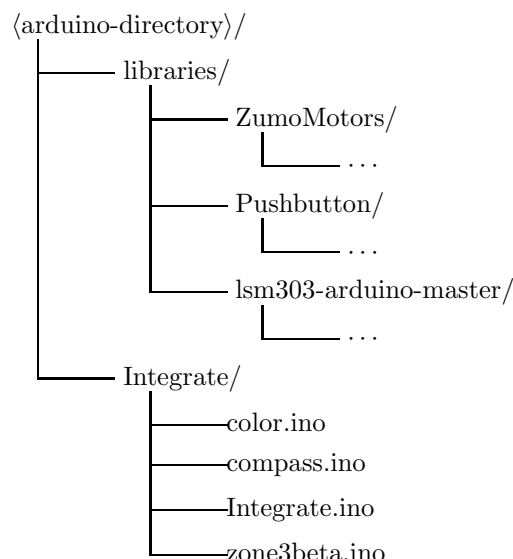
import processing.serial.*;
Serial port;
int ax = 0, ay = 0, az = 0, mx = 0, my = 0, mz = 0;
int CX=250, CY=250;
void setup() {
    size(1000,500);
    port = new Serial(this, "COM3", 9600); // COM3 は各自の環境に合わせて変更せよ
    port.clear();
}
void line3D(float x0, float y0, float z0, float x1, float y1, float z1) {
    float X0 = CX+y0-0.5*x0, Y0 = CY + 1.7320508*x0/2-z0;
    float X1 = CX+y1-0.5*x1, Y1 = CY + 1.7320508*x1/2-z1;
    line(X0,Y0,X1,Y1);
}
void drawVec(float x, float y, float z) {
    stroke(128);
    line3D(0,0,0,250,0,0); line3D(0,0,0,0,250,0); line3D(0,0,0,0,0,250);
    stroke(0);
    line3D(0,0,0,x,y,0); line3D(x,y,0,x,y,z); line3D(0,0,0,x,0,0);
    line3D(x,0,0,x,y,0); line3D(x,y,0,0,y,0); line3D(0,y,0,0,0,0);
    stroke(255,0,0); line3D(0,0,0,x,y,z);
    fill(0); text(x,CX-80,490); text(y,CX,490); text(z,CX+80,490);
}
void draw() {
    noStroke(); fill(255); rect(0,0,1000,500);
    CX = 250;
    drawVec(ax/100,ay/100,az/100);
    CX = 750;
    drawVec(mx,my,mz);
    CX = 600;
    float sc = 0.5;
    line(CX-sc*my,100+sc*mx,CX+sc*my,100-sc*mx);
    line(CX+sc*my,100-sc*mx,CX+0.6*sc*my+0.2*sc*mx,100-0.6*sc*mx+0.2*sc*my);
    line(CX+sc*my,100-sc*mx,CX+0.6*sc*my-0.2*sc*mx,100-0.6*sc*mx-0.2*sc*my);
}
int read2byte(Serial p) {
    int x = p.read(); x <= 8; x |=p.read();
    if (x>32757) x -= 65536;
    return x;
}
void serialEvent(Serial p) {
    if (p.available() >=13 ) {
        if (p.read() == 'H') {
            ax = read2byte(p);    ay = read2byte(p);    az = read2byte(p);
            mx = read2byte(p);    my = read2byte(p);    mz = read2byte(p);
        }
    }
}

```



## 〈課題 11-3.4.1〉簡易ゾーン課題

1. プログラムの雛形を manaba にアップロードしてあるので、それを加筆・修正して今回の簡易ゾーン課題のプログラムを作成する。統合開発環境の arduino スケッチの作業ディレクトリ (フォルダ) は `<arduino-directory>/Integrate/` とし、その下に、`Integrate.ino`, `color.ino`, `compass.ino`, `zone3beta.ino` を配置せよ。ZumoMotors, Pushbutton, LSM303 の各ライブラリはすでにインストールされているものとする。(スケッチ名 “Integrate” は適宜変更してよい。)



注. `Integrate.ino` はメインとなるプログラムで、`setup()`, `loop()` 関数の他、Processing との通信のための `sendData()` 関数が用意されている。 `color.ino` はカラーセンサを使うための関数が、 `compass.ino` は加速度・地磁気センサを使うための関数が、それぞれ、定義されている。 Zumo の電源を入れると **カラーセンサのキャリブレーション** が始まる。 Zumo を白い面の上に置いてユーザープッシュボタンを押すと、10cm ほど前に進みながら “白色” を認識する。 つぎに黒い面の上に置いてユーザープッシュボタンを押すと、10cm ほど前に進みながら “黒色” を認識する。 (**カラーセンサを使わない場合は、この部分はコメントアウトしてよい。**) 地磁気センサの最大値・最小値の初期化の後、`setup()` を抜け `loop()` に入る。 引き続き **地磁気センサのキャリブレーション** を 11-3.3.2 項にしたがって実施せよ。 もう一度ユーザープッシュボタンを押すと山登りの動作が開始される。

2. 本実験では比較的多くの変数を Processing に送信する必要がある。 通信にかかる時間をなるべく短くするため、`int` 型 (2byte) の変数を `unsigned char` 型 (1byte) に変換してからバイナリ形式で送信する。 送信するデータはつぎの 13 byte である。

- データの先頭を表す記号 ‘H’ (1byte)
- プログラムの制御に使う `[0, 255]` の値をとる大域変数 `mode_G` (1byte)
- カラーセンサの RGB 値 (各 1byte, 計 3byte)
- 加速度センサ値 ( $\vec{X}$ ,  $\vec{Y}$ ,  $\vec{Z}$  方向成分)  `$[-2^{15}, 2^{15} - 1]$`  から  `$[0, 255]$`  に変換 (各 1byte, 計 3byte)
- 地磁気センサ値 ( $\vec{X}$ ,  $\vec{Y}$ ,  $\vec{Z}$  方向成分)  `$[\text{min}, \text{max}]$`  から  `$[0, 255]$`  に変換 (各 1byte, 計 3byte)
- モータスピード指示値 (左右)  `$[-256, 255]$`  を  `$[0, 255]$`  に変換 (各 1byte, 計 2byte)

Processing スケッチ `accelerometer` を修正して、これらを受け取り、適当な範囲に変換後、加速度・地磁気の方角や見えている色などを描画・表示せよ。 Arduino から受け取った加速度センサの値 `ax`, `ay`, `az` は演習 11-3.4.1 からスケールが変わっているので注意すること。 (`drawVec` に渡すとき、100 で割らなくてもよい。)

3. 地磁気センサの 3 次元的なキャリブレーション終了後、簡易山登りゾーンのフィールドでロボットを山の方向へ向けもう一度ユーザーブッシュボタンを押してプログラムを実行せよ。山頂まで登ることを確認せよ。
4. zone3beta() の switch() 内 case 4:以降を修正して、山頂で 1 周その場回転 (その際、東西南北で一旦停止) した後、出口方向を向くプログラムを作成し、その動作を確認せよ。
5. 続いて、**最急降下方向** (下の注参照) へ降りるプログラムを作成し、その動作を確認せよ。その際、zone3beta(): switch( mode\_G ) 中の case 1:, case 2: を参考に P-制御を使用すること。
6. 最急降下方向へ降りると、光センサが引っかって Zumo がスタックしてしまう場合がある。これを避けるため、最急降下方向が進行方向に向かって右 45°(図 11-3.4.6 参照) となるように、プログラムを変更し、動作を確認せよ。(ヒント: 加速度の  $\vec{X}$  方向成分と  $\vec{Y}$  方向成分の絶対値が等しくなる。)
7. 下山後、出口方向へ進み、ゾーンを出たら停止するプログラムを作成せよ。また、山頂から下山する方向を工夫して、なるべく短時間でゾーンを出るようにせよ。

**注.** 斜面の高さ  $z$  を、座標  $(x, y)$  の関数  $z(x, y)$  と表すと、各点  $(x, y)$  における**勾配ベクトル** (gradient vector)  $\vec{g}(x, y)$  は

$$\vec{g}(x, y) = \nabla z(x, y) = \begin{bmatrix} \frac{\partial z(x, y)}{\partial x} \\ \frac{\partial z(x, y)}{\partial y} \end{bmatrix}$$

と定義される。微分は傾きを表すことを思い出せば、 $\vec{g}(x, y)$  は点  $(x, y)$  における  $z(x, y)$  の  $x$ -方向、および、 $y$ -方向の傾きを要素としてもつベクトルとなり、点  $(x, y)$  においてもっとも (上りの) 勾配が急な (最急勾配) 方向となる。**最急降下方向**は勾配ベクトルの反対方向  $-\vec{g}(x, y)$  である (図 11-3.4.6)。最急勾配方向も最急降下方向も斜面の各点で定義される**局所的な概念**で、大域的に勾配が最も急なところという概念ではない。

**ヒント.** サンプルプログラム Integrate/zone3beta.ino で山を登るとき (mode\_G=1~2), 左右車輪速の決定は P-制御によって行われている。その部分だけを抜き出すとつぎのようになる。

```
speed0 = 150;
diff = -0.02*compass.a.y;
motorL_G = speed0 + diff;
motorR_G = speed0 - diff;
```

山の頂上が進行方向 (ローカル座標系  $\vec{X}$ -軸の + 方向) の左手 (ローカル座標系  $\vec{Y}$ -軸の + 側) にあるとき、重力の反作用 (グローバル座標系  $\vec{z}$ -軸の + 方向 (鉛直上向き)) はローカル座標系  $\vec{Y}$ -軸の + 側にある。すなわち、compass.a.y は正である。よって、上のプログラムは、左車輪速を小さく、右車輪速を大きくし、進行方向を山の頂上方向 (最急勾配方向) へ向けるように働く。山の頂上が進行方向の右手にあるときも、最急勾配方向へロボットを向けるように働くことを確認せよ。勾配ベクトル  $\vec{g}$  と進行方向  $\vec{X}$  をロボットの位置における**接平面**に射影したとき、それらのなす角  $\Theta$  は、

$$\sin \Theta = \frac{a_Y}{\sqrt{a_X^2 + a_Y^2}}$$

によって与えられる。ここで、 $a_X$ ,  $a_Y$  はそれぞれ加速度センサの  $\vec{X}$ -軸,  $\vec{Y}$ -軸成分である。斜面が水平面となす角がほぼ一定ならば、 $\sqrt{a_X^2 + a_Y^2}$  もほぼ一定である。 $|\Theta| \ll \pi/2$  のとき、 $\Theta \sim \sin \Theta$  なので、上のプログラムは、 $\Theta$  をフィードバックしていると考えられる。最急降下方向と進行方向のなす角はどのように表されるか? また、最急降下方向から 45° 傾いた方向と進行方向のなす角はどのように表されるか?

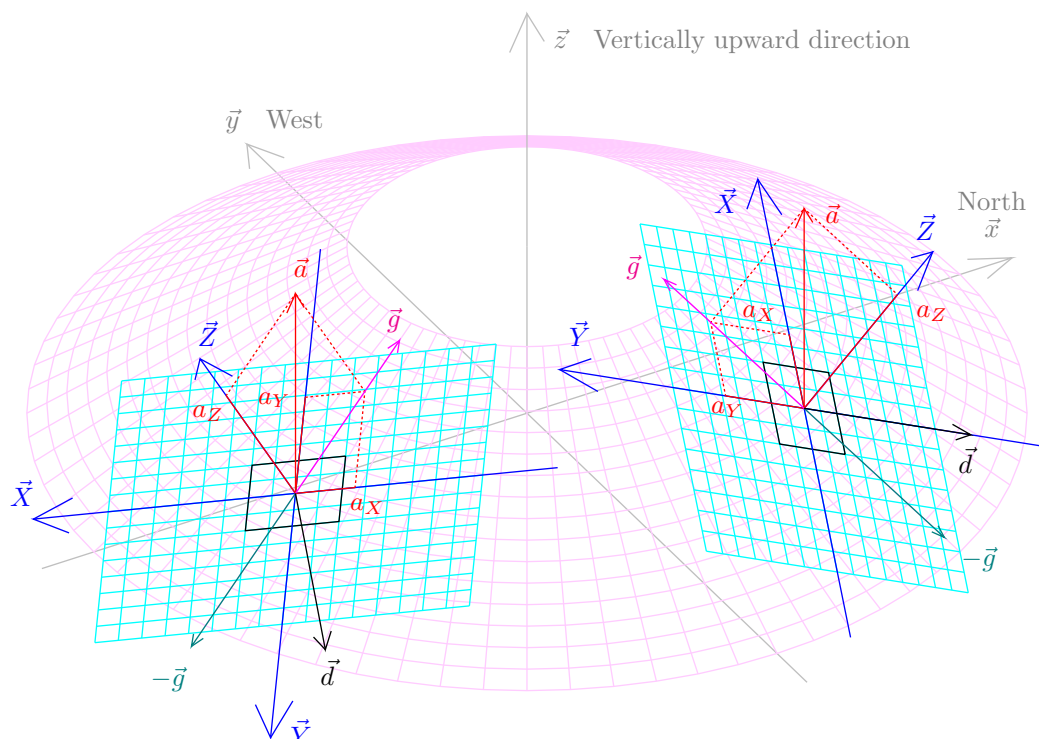


図 11-3.4.6: 接平面, ローカル座標系  $[\vec{X}, \vec{Y}, \vec{Z}]$ , 加速度  $\vec{a}$  (鉛直上向), 勾配ベクトル  $\vec{g}$ , 最急降下方向  $-\vec{g}$ , および,  $-\vec{g}$  を  $\vec{Z}$  軸まわりに  $+45^\circ$  回転したベクトル  $\vec{d}$

#### 〈課題 11-3.4.2〉山腹の周回

1. 山頂から下山する際に, 完全には山を下り切らずに, 山腹を3回周回するプログラムを作成せよ. その際, 車輪は平地または山頂に着かないこと.
2. 山腹を周回中(2周目以降)に, 斜面の角度が最小になる点と, 最大になる点で, それぞれ, 一時停止するようにプログラムを変更せよ.

ヒント: 山腹を周回するためには, P-制御だけでなく, I-制御が必要になる.

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau) d\tau \quad (11-3.3)$$

ここで,  $k_P$ ,  $k_I$  は比例ゲイン, 積分ゲイン,  $e(t) = r - a_x$  は偏差,  $r$  は目標値,  $a_x$  は(重力)加速度の進行方向成分,  $u(t)$  は制御入力(diffに相当)である.

山腹を回っているとどうしてもロボットがスリップしてしまうので, 目標値  $r$  はロボットがちょっと上向き方向になる様に与える. 山頂に近いほど, 曲率半径は小さいので, 山頂に近いほどスリップしやすくなる. 目標値  $r$  は山頂に登り切らない程度にすること.

#### 〈発展課題 11-3.4.1〉Processing による表示

1. 付録 A~付録 Eを参考に, Processing を用いて Zumo ロボットの状態(姿勢, 進行方向など)をわかりやすく描画せよ. たとえば, Zumo ロボットは直方体で表し, 地面に固定した座標系上に, 姿勢や進行方向がわかるように 3D 描画する, など.

## 11-3.5 レポート課題

### [レポート 11-3.5.1]

演習 11-3.4.1 のプログラムの実行状況を説明せよ。その際、roll 角の正負と加速度の  $y$  方向分の正負の関係、pitch 角の正負と加速度の  $x$  方向分の正負の関係などを表にまとめよ。

### [レポート 11-3.5.2]

課題 11-3.4.1 で最終的に出来上がった `zone3beta()` 関数における `mode_G` の各値における動作を日本語または英語で説明せよ。

### [レポート 11-3.5.3]

課題 11-3.4.2-1 で作成した Arduino スケッチの PI-制御の部分をリストとして報告せよ。大域変数、または、static 変数を用いた場合は、その宣言部分や初期値も報告すること。また、設計パラメータ ( $k_P$ ,  $k_I$ ,  $r$  など) の値も報告すること。課題 11-3.4.2-2 で作成した Arduino スケッチを日本語または英語で説明せよ。

### [発展課題レポート 11-3.5.1]

発展課題 11-3.4.1 で作成した Processing スケッチの特徴的なスクリーンショットを図として掲載し、ロボットの状態をどのように表示しているか日本語または英語で説明せよ。その他、工夫した点を説明せよ。

## 参考文献

- [1] H. Goldstein, C. Poole, and J. Safko, 古典力学 (上), (原著第 3 版), 矢野忠, 江沢康生, 淵崎員弘訳, 吉岡書店, 2006.
- [2] 足立修一, デジタル信号とシステム, 東京電機大学出版局, 2002.
- [3] Pololu Corporation, ‘‘Pololu Zumo Shield Libraries,’’<http://www.pololu.com/>.
- [4] Pololu Corporation, ‘‘Arduino library for Pololu LSM303 boards,’’<http://www.pololu.com/>.
- [5] 山崎弘郎, センサ工学の基礎, 昭晃堂, 1985.

# 付録

## 付録 A 投影

3次元空間上の物体を2次元平面に投影することによって、2次元ディスプレイ上に描画することができる。そのプロセスは射影と座標変換からなる。まず、視点  $\vec{v}_p = [v_x, v_y, v_z]^\top$  を一つ定める。ここでは、簡単のため、視点の  $\vec{x}$  軸成分は0 ( $v_x = 0$ )、 $\vec{y}$  軸成分は非正 ( $v_y \leq 0$ )、 $\vec{z}$  軸成分は正 ( $v_z > 0$ ) とする。 $v_x \neq 0$  や  $v_y > 0$  の場合は、3次元空間の座標軸を  $\vec{z}$  軸に対して回転してから投影すればよい。また、ベクトル  $\vec{v}_p$  は方向だけが問題になり、長さは問題にならないので、長さ1に正規化しておく。すなわち、

$$\vec{v}_p = \begin{bmatrix} 0 \\ v_y \\ v_z \end{bmatrix}, \quad v_y^2 + v_z^2 = 1, \quad v_y \leq 0, \quad v_z > 0 \quad (11-3.4)$$

**射影:**  $\vec{v}_p$  を法線ベクトルにもつ平面を  $S$  とし、3次元空間上の点  $\vec{a} = [a_x, a_y, a_z]^\top$  を  $S$  上に投影する。そのような投影は、射影行列  $\Pi = I - \vec{v}_p \vec{v}_p^\top$  を用いて、

$$\vec{\alpha} = \begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{bmatrix} = \Pi \vec{a} = \vec{a} - \vec{v}_p (\vec{v}_p^\top \vec{a}) = \begin{bmatrix} a_x \\ a_y - \vec{v}_p^\top \vec{a} \cdot v_y \\ a_z - \vec{v}_p^\top \vec{a} \cdot v_z \end{bmatrix} \quad (11-3.5)$$

と表される。実際の数値計算では、 $\Pi$  を計算しておいてから行列  $\Pi$  に  $\vec{a}$  を掛けるのではなく、内積  $c = \langle \vec{v}_p, \vec{a} \rangle = \vec{v}_p^\top \vec{a}$  を計算し、 $\vec{a}$  から  $\vec{v}_p$  の  $c$  倍 (スカラー倍) を引く。これによって、計算量は、 $(n^3 + n^2)$  から  $2n^2$  に減らすことができる。 $(n = 3)$  なのであまり違いはないが...

**座標変換:** 上の射影で変換された  $\vec{\alpha} = [\alpha_x, \alpha_y, \alpha_z]^\top$  は  $S$  上の点であるが、その座標はまだ、もとの3次元空間上の  $xyz$ -座標で表されている。このベクトル  $\vec{\alpha}$  を  $S$  上の座標  $\vec{\alpha}_S = [\alpha_X, \alpha_Y, 0]$  に変換する必要がある。平面  $S$  の  $\vec{Z}$  軸は  $\vec{v}_p$  に選んで、 $\vec{X}\vec{Y}$  軸を  $S$  上に取ることにする。また、 $v_x = 0$  となるように選んでおいたので、平面  $S$  の  $\vec{X}$  軸は3次元空間の  $\vec{x}$  軸と同じに選んでおけば簡単である ( $\alpha_X = \alpha_x$ )。平面  $S$  の  $\vec{Y}$  軸は、3次元空間の  $\vec{y}$  軸を  $\vec{z}$  軸方向へ回転したもので、その角度は、3次元空間の  $\vec{z}$  軸と  $\vec{v}_p$  のなす角度に等しい。座標軸の回転なので、ベクトルの長さは変わらない。したがって、

$$\vec{\alpha}_S = \begin{bmatrix} \alpha_X \\ \alpha_Y \\ \alpha_Z \end{bmatrix} = \begin{bmatrix} \alpha_x \\ \text{sgn}(\alpha_y) \sqrt{\alpha_y^2 + \alpha_z^2} \\ 0 \end{bmatrix}. \quad (11-3.6)$$

描画のための  $\vec{X}\vec{Y}$  軸成分は、 $(\alpha_X, \alpha_Y)$  となる。cf.  $\vec{\alpha}$  と  $\vec{\alpha}_S$  はベクトルとしては同じもので、座標軸 (基底) が違うだけである。 ( $[\vec{x}, \vec{y}, \vec{z}] \vec{\alpha} = \alpha_x \vec{x} + \alpha_y \vec{y} + \alpha_z \vec{z} = \alpha_X \vec{X} + \alpha_Y \vec{Y} = [\vec{X}, \vec{Y}, \vec{Z}] \vec{\alpha}_S$ )

注.  $\|\vec{v}_p\|_2 \neq 1$  のとき、

$$\Pi = I - \frac{\vec{v}_p \vec{v}_p^\top}{\vec{v}_p^\top \vec{v}_p}$$

となる。一般に、 $P^2 = P$  (冪等 (べきとう, idempotent)) となるとき、変換  $P: \vec{x} \mapsto P\vec{x}$  を射影という。

$$\Pi^2 = \left( I - \frac{\vec{v}_p \vec{v}_p^\top}{\vec{v}_p^\top \vec{v}_p} \right) \left( I - \frac{\vec{v}_p \vec{v}_p^\top}{\vec{v}_p^\top \vec{v}_p} \right) = I - 2 \frac{\vec{v}_p \vec{v}_p^\top}{\vec{v}_p^\top \vec{v}_p} + \frac{\vec{v}_p (\vec{v}_p^\top \vec{v}_p) \vec{v}_p^\top}{(\vec{v}_p^\top \vec{v}_p)(\vec{v}_p^\top \vec{v}_p)} = I - \frac{\vec{v}_p \vec{v}_p^\top}{\vec{v}_p^\top \vec{v}_p} = \Pi$$

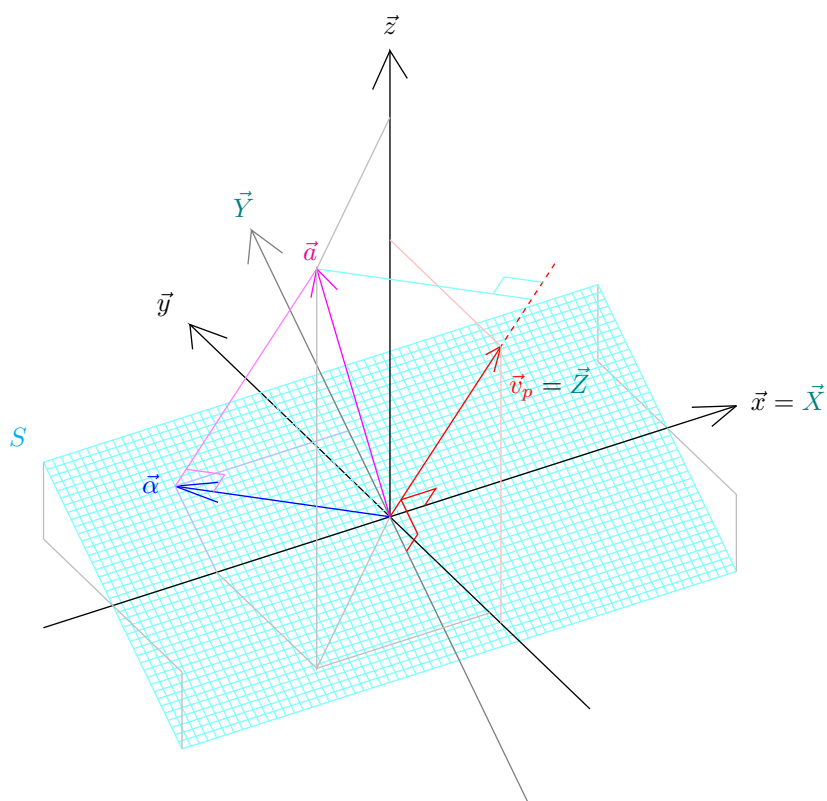


図 付録 A.7: 投影面  $S$  とその法線ベクトル  $\vec{v}_p$ , および, ベクトル  $\vec{a}$  とその投影  $\vec{a}$

となり, 確かに  $\Pi$  は射影行列である. また, 冪等性に加えて, 対称性  $P = P^\top$  も成り立つとき, 変換  $P$  を直交射影 (orthogonal projection) という.  $\Pi$  による変換は直交射影である. 直交射影でない射影を斜交射影 (oblique projection) ということもある. 投影面に対して斜めから光が当たる場合は斜交射影になる.

## 付録 B 陰線処理

視点からは隠れて見えない部分を表現しないようにするための処理が陰線処理である. 陰線処理を実現するアルゴリズムは多数あるが, ここでは, 法線ベクトル法と塗り重ね法について説明する.

物体の表面は  $N$  個の多角形 (ポリゴン)  $P_i$ ,  $i = 1, \dots, N$  で構成されている (直線と平面のみから構成されている) とし, 各多角形は凸多角形と仮定する. (もし凸でなければ, 多角形を分割して, 分割された多角形が凸となるようにする. cf. 三角形に分割すれば必ず凸である.) また, 法線ベクトル法では, 物体は凸多面体であると仮定する.

### 法線ベクトル法

物体を構成するポリゴン  $P_i$  の法線ベクトルを  $\vec{n}_i$ ,  $i = 1, \dots, N$  とする. 視点  $\vec{v}_p$  と  $\vec{n}_i$  の内積が正  $\langle \vec{v}_p, \vec{n}_i \rangle > 0$  の場合のみポリゴン  $P_i$  を描画する.

### 塗り重ね法

原点からポリゴン  $P_i$  の重心までのベクトルを  $\vec{c}_i$ ,  $i = 1, \dots, N$  とする. 視点  $\vec{v}_p$  との内積  $d_i = \langle \vec{v}_p, \vec{c}_i \rangle$  は奥行情報であり,  $\langle \vec{v}_p, \vec{c}_i \rangle > \langle \vec{v}_p, \vec{c}_j \rangle$  のとき,  $P_i$  の方が  $P_j$  よりも手前にあることを利用する.



$d_i$  によってソーティング後,  $d_i$  の小さいポリゴン  $P_i$  から順に描画する. (ポリゴンの内側は塗りつぶす.)

塗り重ね法は凸多面体でない物体へも適用可能である. ただし, ある面が別の面を貫通しているような場合はうまくいかない. このような場合は, ポリゴンを分割するなど工夫が必要である.

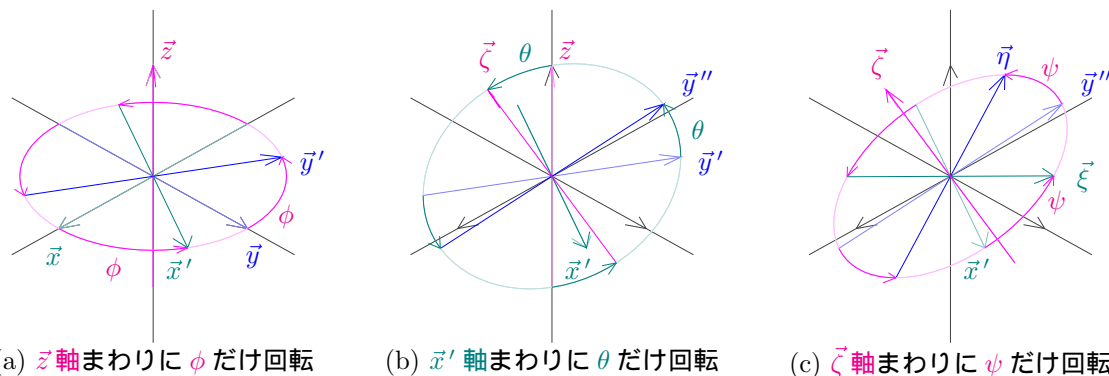
## 付録 C Euler 角

剛体の一般化座標は位置 (並進) と姿勢 (回転) によってあらわされる. 並進と回転はそれぞれ 3 つの独立な関数によってあらわされる. ここでは, 回転のみを考える. **グローバル座標系** (普通は慣性系) と剛体に固定された **ローカル座標系** の回転に関する関係を表す重要なものが **Euler 角** である. Euler 角は回転に関する座標変換を **3 回の引き続いて起こる回転の積** として表したものであり, **どの軸のまわりにどの順番で** 回転したかが重要である.

北向きの単位ベクトル (大きさ 1 のベクトル) を  $\vec{x} \in \mathbb{R}^3$ , 西向きの単位ベクトルを  $\vec{y} \in \mathbb{R}^3$ , 鉛直上向きの単位ベクトルを  $\vec{z} \in \mathbb{R}^3$  とする. これらは 3 次元空間の **正規直交基底 (orthonormal basis)** をなす. グローバル座標系  $V_{\text{global}} = [\vec{x}, \vec{y}, \vec{z}] \in \mathbb{R}^{3 \times 3}$  とローカル座標系  $V_{\text{local}} = [\vec{\xi}, \vec{\eta}, \vec{\zeta}] \in \mathbb{R}^{3 \times 3}$  との間関係をつぎのような 3 回の回転として表す (図付録 C.8):

- 座標系  $V_{\text{global}}$  を  **$\vec{z}$  軸** まわりに  $\phi$  だけ回転した座標系を  $V' = [\vec{x}', \vec{y}', \vec{z}] \in \mathbb{R}^{3 \times 3}$  とする.
- つぎに  $V'$  を  **$\vec{x}'$  軸** まわりに  $\theta$  だけ回転した座標系を  $V'' = [\vec{x}', \vec{y}'', \vec{\zeta}] \in \mathbb{R}^{3 \times 3}$  とする.
- 最後に  $V''$  を  **$\vec{\zeta}$  軸** まわりに  $\psi$  だけ回転した座標系を  $V_{\text{local}} = [\vec{\xi}, \vec{\eta}, \vec{\zeta}] \in \mathbb{R}^{3 \times 3}$  とする.

このとき,  $(\phi, \theta, \psi)$  を **Euler 角** という.  $\vec{x}, \vec{y}, \vec{z}$  軸すべてが対等ではなく  $\vec{z}, \vec{x}', \vec{\zeta}$  軸の順番に回転することに注意. 上記の取り方を  **$z$ - $x$ - $z$  convention** あるいは  **$x$  規約** という. この他  **$y$  規約**,  **$xyz$  規約** などがよく使われる. 以下では,  $x$ -規約を用いることにする.



(a)  $\vec{z}$  軸まわりに  $\phi$  だけ回転 (b)  $\vec{x}'$  軸まわりに  $\theta$  だけ回転 (c)  $\vec{\zeta}$  軸まわりに  $\psi$  だけ回転

図 付録 C.8: Euler 角:  $\phi = 60^\circ, \theta = 30^\circ, \psi = 45^\circ$  のときの座標系の回転.

簡単のため, グローバル座標系  $V_{\text{global}}$  を **標準基底** (後述) にとると,  $V_{\text{global}}, V', V'', V_{\text{local}}$  の各座

標系の間の関係は**回転行列**  $R_z(\phi)$ ,  $R_x(\theta)$ ,  $R_z(\psi)$  を使ってつぎのように表現される.

$$\begin{aligned}
 V' = [\vec{x}', \vec{y}', \vec{z}] &= \begin{bmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix} [\vec{x}, \vec{y}, \vec{z}] = R_z(\phi) V_{\text{global}} \\
 &\quad (z \text{ 軸まわりに剛体が } \phi \text{ だけ回転}) \\
 V'' = [\vec{x}', \vec{y}'', \vec{\zeta}] &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} [\vec{x}', \vec{y}', \vec{z}] = R_x(\theta) V', \\
 &\quad (x' \text{ 軸まわりに剛体が } \theta \text{ だけ回転}) \\
 V_{\text{local}} = [\vec{\xi}, \vec{\eta}, \vec{\zeta}] &= \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} [\vec{x}', \vec{y}'', \vec{\zeta}] = R_z(\psi) V'', \\
 &\quad (\zeta \text{ 軸まわりに剛体が } \psi \text{ だけ回転})
 \end{aligned}$$

注.  $R_z(\psi)$ ,  $R_x(\theta)$ ,  $R_z(\phi)$  は**直交行列 (orthogonal matrix)** である. たとえば,  $R_z(\psi)^\top R_z(\psi) = R_z(\psi)R_z(\psi)^\top = I$ . したがって,  $R_z(\psi)^{-1} = R_z(\psi)^\top = R_z(-\psi)$ .  $R_x(\theta)$  や  $R_z(\phi)$  も同様. また,  $V_{\text{global}}$  は正規直交基底だったので, 直交行列で変換された,  $V'$ ,  $V''$ ,  $V_{\text{local}}$  も正規直交基底である.

以上をまとめると,

$$\begin{aligned}
 V_{\text{local}} &= \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \\
 &\quad \cdot \begin{bmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix} V_{\text{global}}, \\
 &= R_z(\psi)R_x(\theta)R_z(\phi)V_{\text{global}} = R_{zxz}(\phi, \theta, \psi)V_{\text{global}}. \tag{11-3.7}
 \end{aligned}$$

ここで,

$$R_{zxz}(\phi, \theta, \psi) = \begin{bmatrix} \cos \psi \cos \phi - \cos \theta \sin \psi \sin \phi & \cos \psi \sin \phi + \cos \theta \sin \psi \cos \phi & \sin \psi \sin \theta \\ -\sin \psi \cos \phi - \cos \theta \cos \psi \sin \phi & -\sin \psi \sin \phi + \cos \theta \cos \psi \cos \phi & \cos \psi \sin \theta \\ \sin \theta \sin \phi & -\sin \theta \cos \phi & \cos \theta \end{bmatrix}. \tag{11-3.8}$$

上の注より,

$$\begin{aligned}
 R_{zxz}(\phi, \theta, \psi)^{-1} &= \{R_z(\psi)R_x(\theta)R_z(\phi)\}^{-1} = R_z(\phi)^{-1}R_x(\theta)^{-1}R_z(\psi)^{-1} = R_z(-\phi)R_x(-\theta)R_z(-\psi) \\
 &= R_{zxz}(-\psi, -\theta, -\phi).
 \end{aligned}$$

## 付録 D 各座標系 (基底) で表した数ベクトルの間の関係

$V_{\text{global}}$  の列ベクトルの組は 3 次元空間の**基底**をなすので, ベクトル  $\vec{a}$  を

$$\vec{a} = V_{\text{global}} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = a_x \vec{x} + a_y \vec{y} + a_z \vec{z}$$



と数ベクトル  $\vec{a}_{\text{global}} = [a_x, a_y, a_z]^\top$  として表現できる. この  $V_{\text{global}}$  は 標準基底 (canonical basis) となるように選んでおくのが自然であろう. すなわち, 第  $i$  成分が 1 で他の成分は 0 となる単位ベクトルを  $e_i \in \mathbb{R}^3$  ( $i = 1, 2, 3$ ) とすると,

$$V_{\text{global}} = [\vec{x}, \vec{y}, \vec{z}] = [\vec{e}_1, \vec{e}_2, \vec{e}_3] = I.$$

(cf. 実は前節の回転行列の表現はこの標準基底によって数ベクトルとして表現することを前提としていた.) 一方,  $V_{\text{local}}$  の列ベクトルの組も 3 次元空間の (正規直交) 基底をなすので, 同様に,

$$\vec{a} = V_{\text{local}} \begin{bmatrix} a_\xi \\ a_\eta \\ a_\zeta \end{bmatrix} = a_\xi \vec{\xi} + a_\eta \vec{\eta} + a_\zeta \vec{\zeta}$$

と数ベクトル  $\vec{a}_{\text{local}} = [a_\xi, a_\eta, a_\zeta]^\top$  として表現できる. したがって,  $\vec{a}_{\text{global}}$  と  $\vec{a}_{\text{local}}$  の間の関係は,

$$\vec{a} = V_{\text{local}} \vec{a}_{\text{local}} = V_{\text{global}} \vec{a}_{\text{global}}$$

となる.  $V_{\text{global}} = I$ ,  $V_{\text{local}} = R_{zz}(\phi, \theta, \psi)$  に注意すると,

$$\vec{a}_{\text{local}} = V_{\text{local}}^{-1} V_{\text{global}} \vec{a}_{\text{global}} = R_{zz}(-\psi, -\theta, -\phi) \vec{a}_{\text{global}} = R_{zz}(\alpha, \beta, \gamma) \vec{a}_{\text{global}} \quad (11-3.9)$$

ここで,

$$\alpha = -\psi, \quad \beta = -\theta, \quad \gamma = -\phi \quad (11-3.10)$$

である.

注. 重力加速度および地磁気の磁束密度をグローバル座標系で表した数ベクトルを,  $\vec{a}_{\text{global}} = [0, 0, g]^\top$  および  $\vec{m}_{\text{global}} = [m_x, 0, m_z]^\top$  とし, ローカル座標系で表した数ベクトルを  $\vec{a}_{\text{local}}$  および  $\vec{m}_{\text{local}}$  とする. ロボットが静止状態にあるとき, ロボット上の加速度センサや地磁気センサの値は,  $\vec{a}_{\text{local}}$  や  $\vec{m}_{\text{local}}$  の定数倍 (各センサのゲイン倍) に他ならない.

## 付録 E 加速度・地磁気センサ値から Euler 角への変換

ロボットは静止していると仮定し, そのときの加速度センサ値  $\vec{a}_{\text{local}} = [a_\xi, a_\eta, a_\zeta]^\top$ , および, 地磁気センサ値  $\vec{m}_{\text{local}} = [m_\xi, m_\eta, m_\zeta]^\top$  からロボットの姿勢を表す Euler 角  $(\phi, \theta, \psi)$  を求める. グローバル座標系で表されている重力加速度や磁束密度からローカル座標系で表されている各センサ値に変換する Euler 角を  $(\alpha, \beta, \gamma)$  とすると,

$$\phi = -\gamma, \quad \theta = -\beta, \quad \psi = -\alpha \quad (11-3.11)$$

である. 加速度・地磁気センサの値は, 方向のみが重要なので,  $\|\vec{a}_{\text{local}}\|_2 = 1$ ,  $\|\vec{m}_{\text{local}}\|_2 = 1$  と正規化されているものとする.

まず, 加速度センサの値  $\vec{a}_{\text{local}}$  は  $\vec{a}_{\text{global}} = [0, 0, 1]^\top$  を  $R_{zz}(\alpha, \beta, \gamma)$  で変換したものである, (11-3.9) 式より,

$$\vec{a}_{\text{local}} = \begin{bmatrix} a_\xi \\ a_\eta \\ a_\zeta \end{bmatrix} = \begin{bmatrix} \sin \gamma \sin \beta \\ \cos \gamma \sin \beta \\ \cos \beta \end{bmatrix} \quad (11-3.12)$$

となる. これより,

$$\cos \beta = a_\zeta, \quad \sin \beta = \text{sgn}(a_\eta) \sqrt{1 - a_\zeta^2} \quad (11-3.13)$$

となる. つぎに,  $|\sin \beta| \sim 0$  のときは水平面内での回転なので, 地磁気センサの  $\xi\eta$  成分だけから,

$$\sin \gamma = -cm_\eta, \quad \cos \gamma = cm_\xi, \quad \alpha = 0, \quad \beta = \begin{cases} 0 & a_\zeta \geq 0 \\ \pi & \text{otherwise} \end{cases} \quad (11-3.14)$$

となる．ここで、 $c = 1/\sqrt{m_\xi^2 + m_\eta^2}$  であるが、 $\gamma$  を計算する際に、 $\text{atan2}()$  関数を用いることにすれば、 $c$  を具体的に計算する必要はない ( $\gamma = \text{atan2}(-m_y, m_x)$ )．これで、 $|\sin \beta| \sim 0$  のときは終了である (Euler 角がすべて求まった)．一方、 $\sin \beta \neq 0$  のときは、

$$\sin \gamma = a_\xi / \sin \beta, \quad \cos \gamma = a_\eta / \sin \beta \quad (11-3.15)$$

となる．最後に、 $\alpha$  を求める．地磁気の方法は水平面内だけでなく  $z$  軸方向成分ももつ (水平面となす角を**伏角**という) ので、3 次元的な動きをするロボットの正確な姿勢を知るためには、磁北の向きの 3 次元ベクトルが必要である．それを  $\vec{\mu}_{\text{global}} = [\mu_x, \mu_y, \mu_z]^\top$  とおく．(地磁気センサのキャリブレーションの際に、 $z$  軸方向もキャリブレーションしておくこと．) **磁北の向きは、ロボットを水平面に置いたときのセンサ値**  $\vec{m}_0 = [m_{0x}, m_{0y}, m_{0z}]^\top$  から、

$$\vec{\mu}_{\text{global}} = \begin{bmatrix} \mu_x \\ 0 \\ \mu_z \end{bmatrix} = \frac{1}{\sqrt{m_{0x}^2 + m_{0y}^2 + m_{0z}^2}} \begin{bmatrix} \sqrt{m_{0x}^2 + m_{0y}^2} \\ 0 \\ m_{0z} \end{bmatrix} \quad (11-3.16)$$

となる．ここで、 $\vec{\mu}_{\text{global}}$  は大きさ 1 に正規化されていることに注意．(11-3.9) 式と同様に、 $\vec{m}_{\text{local}} = R_{zz}(\alpha, \beta, \gamma) \vec{\mu}_{\text{global}} - \mu_z \vec{a}_{\text{local}}$  である． $\vec{b} = \vec{m}_{\text{local}} - \mu_z \vec{a}_{\text{local}}$  とおくと、 $\|\vec{a}_{\text{local}}\|_2 = 1$  に正規化されているので、

$$\begin{aligned} \vec{b} &= R_{zz}(\alpha, \beta, \gamma) \vec{\mu}_{\text{global}} - \mu_z \vec{a}_{\text{local}} \\ &= R_{zz}(\alpha, \beta, \gamma) \begin{bmatrix} \mu_x \\ 0 \\ 0 \end{bmatrix} + R_{zz}(\alpha, \beta, \gamma) \begin{bmatrix} 0 \\ 0 \\ \mu_z \end{bmatrix} - \mu_z R_{zz}(\alpha, \beta, \gamma) \vec{a}_{\text{global}} = R_{zz}(\alpha, \beta, \gamma) \begin{bmatrix} \mu_x \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

(11-3.7), (11-3.10) 式を考慮すると、

$$\vec{b} = \mu_x \begin{bmatrix} \cos \gamma & -\cos \beta \sin \gamma \\ -\sin \gamma & -\cos \beta \cos \gamma \\ 0 & \sin \beta \end{bmatrix} \begin{bmatrix} \cos \alpha \\ \sin \alpha \end{bmatrix} = \mu_x A \begin{bmatrix} \cos \alpha \\ \sin \alpha \end{bmatrix} \quad (11-3.17)$$

これを  $[\cos \alpha, \sin \alpha]^\top$  について解けばよい． $\|\vec{m}_{\text{local}}\|_2 = 1$  に注意すると、 $A^\top A = I$  より、

$$\begin{bmatrix} \cos \alpha \\ \sin \alpha \end{bmatrix} = \frac{1}{\mu_x} (A^\top A)^{-1} A^\top \vec{b} = \frac{1}{\mu_x} A^\top \vec{b} = \frac{1}{\mu_x} \begin{bmatrix} \cos \gamma & -\cos \beta \sin \gamma \\ -\sin \gamma & -\cos \beta \cos \gamma \\ 0 & \sin \beta \end{bmatrix}^\top \begin{bmatrix} m_\xi - \mu_z \sin \gamma \sin \beta \\ m_\eta - \mu_z \cos \gamma \sin \beta \\ m_\zeta - \mu_z \cos \beta \end{bmatrix}. \quad (11-3.18)$$

プログラムとして実装する際に用いる式を、**マジENTA**で表しておいた．

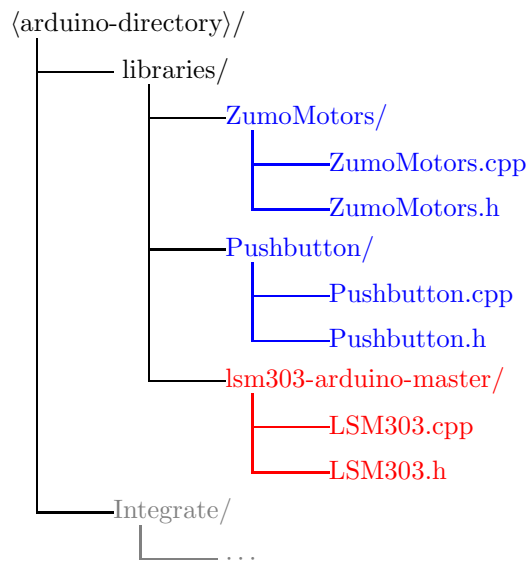
## 付録 F ライブラリの追加

**ZumoMotors**, **Pushbutton**, **LSM303** のライブラリがすでにインストールされている場合は、本節はスキップしてよい．

ZumoShield には LSM303 という**加速度センサ**と**地磁気センサ**がついている．カラーセンサ同様 I<sup>2</sup>C (Inter-Integrated Circuit) というシリアル通信方式によって、これらのセンサの値を取得することができる．加速度センサ・地磁気センサの値を読み取るためのライブラリ **LSM303** が用意されている．

**LSM303 ライブラリ** [4] がまだ追加されていない場合は、**ZumoMotors ライブラリ** [3] と同様に追加せよ．libraries 以下のディレクトリは、表付録 F.1 のようになるはずである．表中の <arduino-directory> は、例えば、つぎのようになっているであろう：

表 付録 F.1: Arduino ライブラリの構造



Windows の場合	">PC> ドキュメント >Arduino"
Mac の場合	"書類/Arduino/"
Linux(デフォルト) の場合	"~/sketchbook/"
K313 の計算機の場合	"~/arduino/"

<arduino-directory> は, ArduinoIDE を立ち上げ, “ファイル”→ “環境設定” を選択すると, “スケッチブックの保存場所:” から確認・編集できる.

### ライブラリファイルの取得とインストール

- LSM303 のライブラリは, <https://github.com/pololu/lsm303-arduino/>の右側の “Clone or download”→ “Download ZIP” をクリックして得られる.
- lsm303-arduino-master.zip を展開すると, lsm303-arduino-master というフォルダができる. これを libraries/ の下にコピーすればよい. あるいは, つぎのようにしてもよい: ArduinoIDE から, “スケッチ” → “ライブラリをインクルード” → “.ZIP 形式のライブラリをインストール...” → ZIP ファイルを指定. (ライブラリによってはこの手法が使えないこともある)



ディレクトリ “lsm303-arduino-maser” は “<arduino-directory>/libraries/” の直下に配置せよ.