小型移動ロボットのための三次元位置測定システム

宮田, 好洋 / MIYATA, Yoshihiro

(発行年 / Year)
2006-03-24
(学位授与年月日 / Date of Granted)
2006-03-24
(学位名 / Degree Name)
修士(工学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

法政大学大学院　工学研究科　電気工学専攻

2005 年度　修士論文

小型移動ロボットのための
三次元位置測定システム

3D Terrain Measurement Sensor System for Mobile Robot

小林尚登研究室　04R3244　　宮田　好洋
ミヤタ　ヨシヒロ

指導教授

法政大学	小林 尚登
	中村 秀男
理化学研究所	川端 邦明
東海大学	稲垣 克彦
東京農業大学	樹野 淳也

Abstract

This paper proposes a 3D terrain measurement sensor system for mobile robots. The system consists of an infrared sensor and a RC-servo. We explain the hardware of this system and its signal processing method. Several experiments by our prototype show the effectiveness of our proposal.

本論文の構成

本論文は，全体が６章から構成される．

１章では，本研究の背景について述べる．第２章で三次元センサシステムの概要，第３章では三次元センサシステムの構成，第４章ではセンシング方法ついて述べる．さらに，第５章では，提案機構の実験モデルを紹介して，実験により検証する．そして最後に，第６章で結論を述べる．

目 次

図 目 次

第 1 章 序論

災害現場において，発生直後の迅速な救助が大きな役割を果たすと言われている．この救助活動では，一人でも多くの被災者を助けなければならないが，倒壊した建物の屋内や屋外は未知環境であり不整地であると予想され，救助活動は救助作業者たちにとって人命救助や行方不明者の捜索など困難かつ二次被害をもたらす危険性もある．そのため危険性が高い未知環境では，レスキューロボットなど移動ロボットの活躍が期待されている．しかし，このような状況下では、克服しなければならないさまざまな障害がある．その為，オペレータへの情報提示・自己位置認識・軌道生成のための三次元マッピングを実行する障害物三次元センサシステムが必要不可欠である．

現在，障害物三次元センサシステムとして多数のセンサを円弧状に配置する物や，レーザレーダ・CCD カメラを使用したものがあげられる．しかし，重量・サイズ肥大化により，機動力の高い小型移動ロボットには搭載困難である．そこで，本研究では，小型移動ロボットのための三次元位置測定システムを提案する．

第 2 章 三次元センサシステムの概要

外界の情報を得るためのセンサとしては各種のものがある．中でも対象物までの距離を測定することのできる距離センサは，移動ロボットの環境認識を行う上で非常に有用であるといえる．しかし，距離センサから得られる情報は一次元であるため，三次元データを得るためにはセンサの向きを変化させる必要がある．よって，距離センサに能動関節を加えることにより，三次元データを得る小型センサシステムを提案する．提案するセンサシステムを開発するにあたり以下のような仕様を設定した．

- 省スペースかつ軽量
- 計測範囲はロボットの前方 $0.7 \pm 0.3[m]$，幅 $0.3[m]$，高さ $0.6[m]$
- 障害物の形状と位置を認識可能

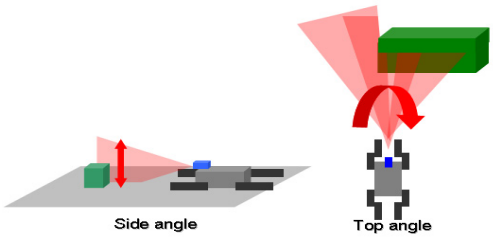


Fig 2.1 Image of Scaning

第 3 章 三次元センサシステムの構成

本章では，センサシステムを構成するセンサの選定，センサ最適化のためにセンサの構造変更・フィルタの設計と首振り機構を示す.

3.1 センサの選定

距離センサとして，主にレーザレーダ・超音波センサ・赤外線センサ・CCD カメラを用いたステレオビヨンシステムなど，多数存在する．レーザレンジファインダを用いたシステムは短時間で高精度の情報が得られる．しかし，装置が大きく，移動ロボットに搭載するにはスペースを必要とする．CCD カメラを用いた手法は外乱などに強いという長所があるが，多数の画像を処理しなければならないため，計算コストを必要としたり，平坦な画像では特徴点が抽出しにくいという短所も存在する．超音波距離計を用いたシステムは光学的な手法と比べて耐環境性が高いが，音波を利用しているため，指向性に乏しいといった欠点がある．赤外線距離計は指向性が高く，高精度で距離が測定可能という長所があり，短所は測定可能範囲が近距離に限られる点である．しかし，本システムの採用を考えた場合，1[m] 前方まで測定できればよいので，この短所はそれほど問題にはならないと考えられる．よって，本システムでは，赤外線距離計を用いる事とする.

3.2　赤外線距離計の構造変更

デバイスはシャープ製の GP2D12 という処理回路と赤外線センサが一体となったものを使用する．このデバイスは距離をアナログ電圧で出力する．また測定可能範囲は 0.1〜0.8[m] である．

この測定可能範囲では要求仕様を満たすことができないため，デバイスに小変更を加えた．具体的には，三角測量の原理で距離を測定しているため，PSD と赤外線 LED 間の距離を拡大した．(Fig.3.1,Fig.3.2)，電圧-距離特性を Fig.3.3 示す．これによって，測定可能範囲は 0.3[m]〜1.0[m] にオフセットすることが可能となった．

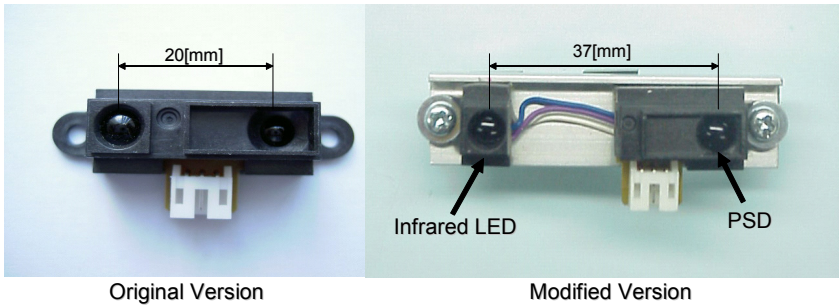


Fig 3.1 Overview of GP2D12

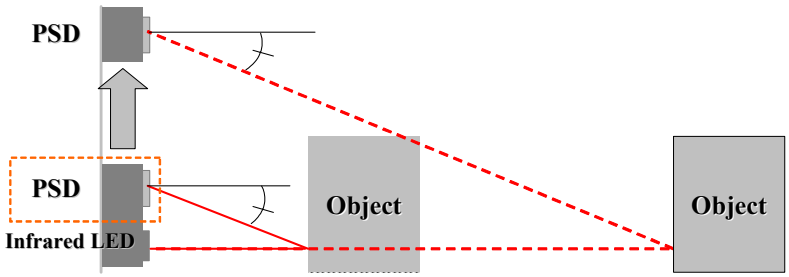


Fig 3.2 Principle of Distance Measurment

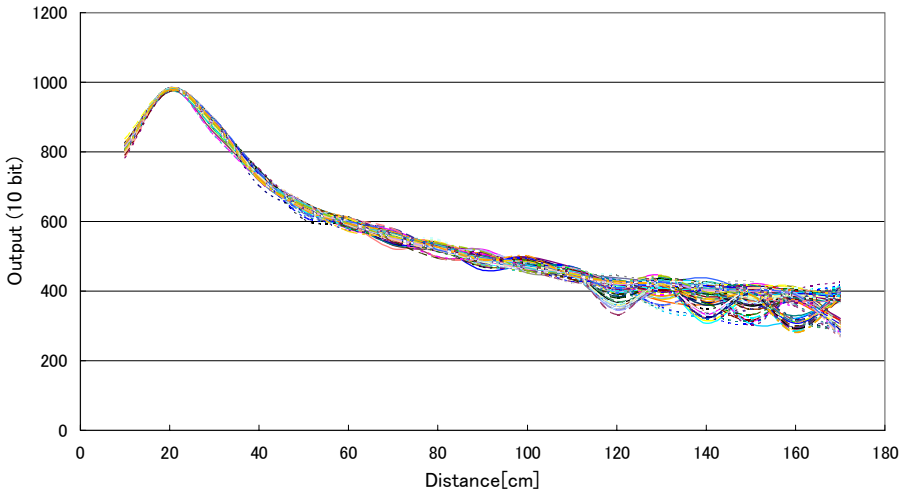


Fig 3.3 Characteristics of Modified GP2D12

3.3　フィルタの設計

Fig.3.3から距離が大きくなるにしたがって，出力電圧のばらつきが大きくなっていることが確認できる．したがって，この部分になんらかのノイズが存在している可能性がある．

よってこのノイズを除去するためにフィルタを設計する．そのために，デバイスの出力電圧をFFT変換しスペクトルを求めた(Fig.3.4)．この図から確認できるとおり，主要なノイズの周波数は8Hz付近にある．よってこの周波数帯を除去するノッチフィルタと高周波成分除去のためのローパスフィルタを設計した．このフィルターの回路，周波数特性をFig.3.5に示す．このフィルタを通して得られたデバイスの出力電圧の様子をFig.3.6に示す．フィルタの設置によりノイズが減少して精度向上・測距可能範囲向上（0.3 - 1.2$[cm]$）された．次に，この関係を数式により表現する．

又，Fig.3.6の曲線を累乗近似することにより，以下の距離-電圧特性の式を得た．(Fig.3.7)

$$d = 5.0 \times 10^{7} v^{-2.12896}$$

d ： 対象物までの距離 $[cm]$

v ： $GP2D12$出力電圧（$10bit$）

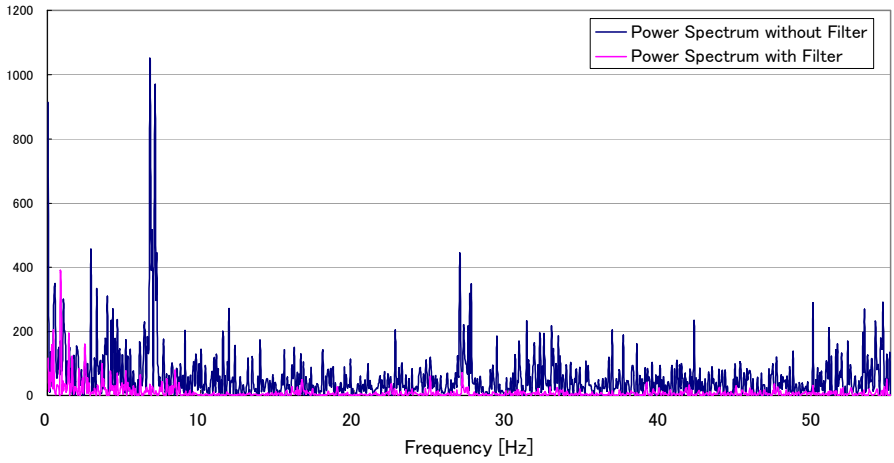


Fig 3.4 Power Spectrum

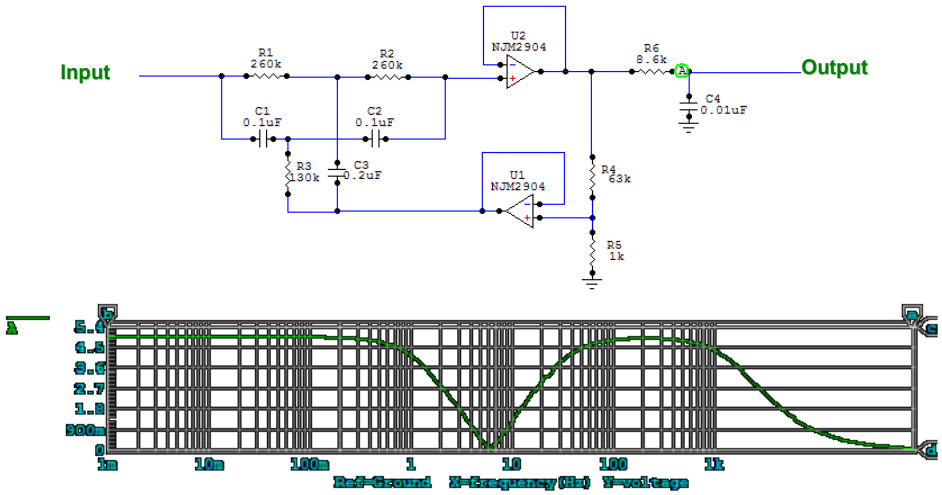


Fig 3.5 BEF,LPF・Frequency Response

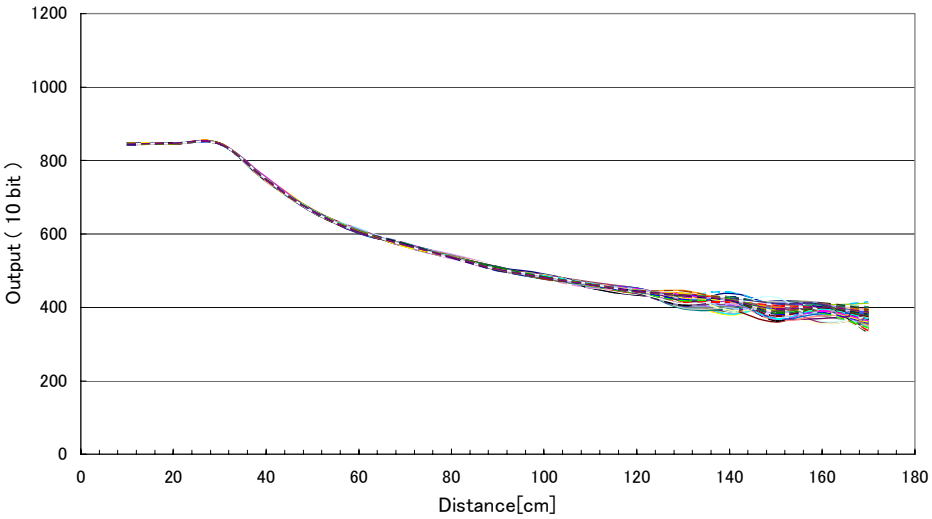


Fig 3.6 Characteristics GP12D12 with Filter

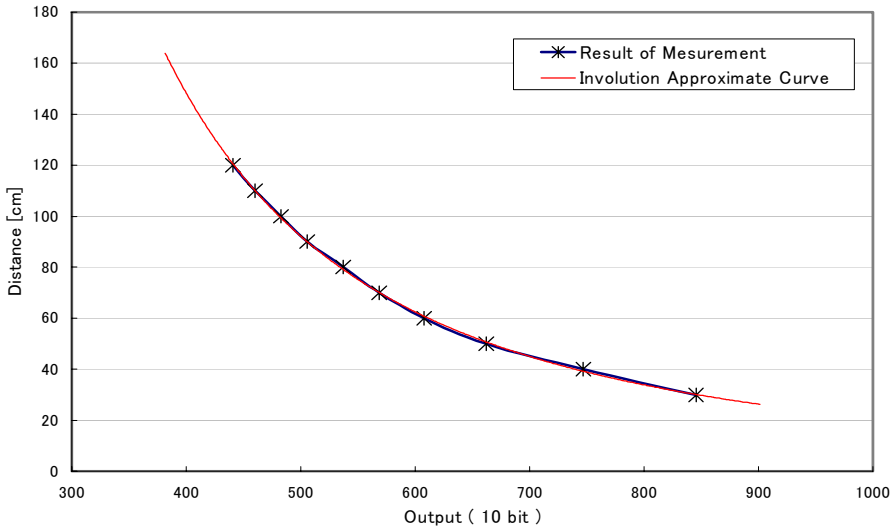


Fig 3.7 GP2D12 (Output - Distance)

3.4　走査のための首振り

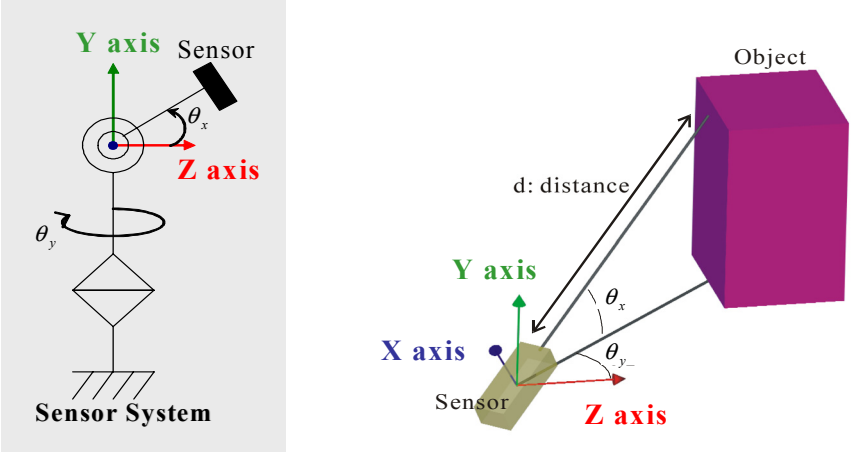


Fig 3.8 Definition of Parameters

距離センサにピッチ・ヨー方向の２つの能動関節機構を持たせる．これにより，3次元データが得られることとなる．その情報は，ヨー方向の回転角度，ピッチ方向の回転角度，センサから対象物までの距離，の3つである (Fig.3.8)．すなわち，球座標系上の情報になる．

センサーから得られた情報を容易に可視化するために，座標系の変換を行う．すなわち，極座標系から直交座標系への変換である．次に，球座標系の情報を直交座標系に変換する．なお，この直交座標系はセンサ基部におけるローカル座標系である．よって，この変換の数式は以下のように表せられる．

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix}$$

$$\theta_x \quad : \quad x\text{軸周り角度}\,[deg]$$

$$\theta_y \quad : \quad y\text{軸周り角度}\,[deg]$$

第 4 章 センシング方法

本章では，スキャン軌道の提案と障害物の寸法認識について述べる．

4.1 スキャン軌道の提案

センサ部の首振りは，Fig4.1 のように縦横網を描写する軌道を描く。これにより障害物をスライス状に認識でき，すばやく形状を読み取ることができる．又，$GP2D12$ は内臓の信号処理回路により，サンプリングタイムが 38.3$[ms]$ となっている．そのためにセンサをステップ状に動かして，測定中は動きを止める．よりセンシング速度を向上させるために，計測距離が 1.0$[cm]$ 以上の時はステップ幅を広げる．

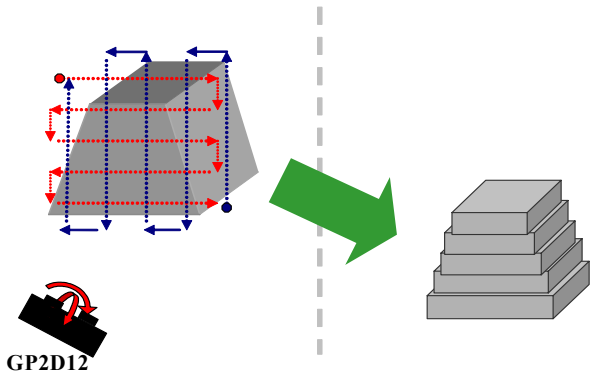


Fig 4.1 Image of Scaning

4.2　盲点部の計測

距離センサは一次データを読み取るために，Fig.4.2 のような盲点が存在する. このような盲点部を計測するため，Fig.4.2 のように何らかの機構を使用してセンサを持ち上げる必要がある．又，蛇型ロボットのような場合は，ロボット自身の自由度を最大限に活用することで，センサ単体では測定不可能な範囲も測定可能範囲にすることができ，測定可能範囲を大きく拡大することが可能となる.

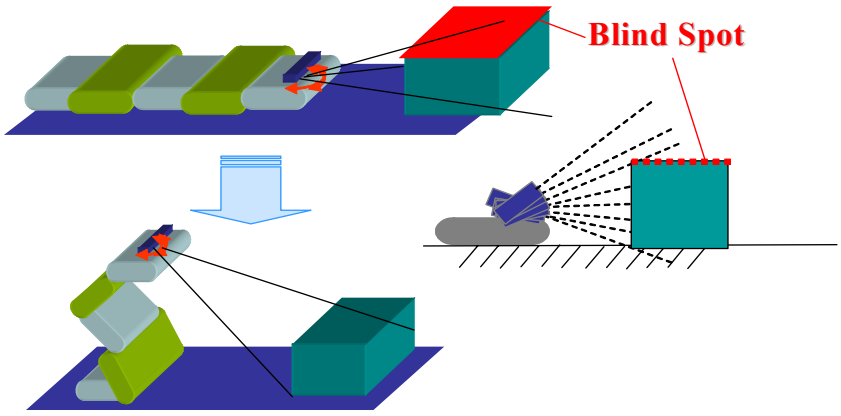


Fig 4.2 Image of Motion Sensor

4.3　障害物の寸法認識

障害物寸法はエッジを検出することにより導出できる．しかし，Fig.4.3のようにエッジ部かつ盲点の境を読み取ると，なまった値が出力される．これはFig.4.3に示す様に出力の赤外線が幅を持っているために，盲点左右の反射を同時に読み取っているためだと思われる．そこで，エッジを抽出するためにセンサより出力された値とその前後の値を使って，最小二乗法により関数の傾きを導き出す．

$$a_{n+1} = \frac{n\sum_{i=0}^{2n+1} x_i y_i - \sum_{i=0}^{(2n+1)/2} x_i \sum_{i=0}^{(2n+1)/2} y_i}{n\sum_{i=1}^{(2n+1)/2} x_i^2 - (\sum_{i=0}^{(2n+1)/2} x_i)^2}$$

$$(n = 1, 2, \cdots, n)$$

点(x_{n+1}, a_{n+1})より描かれる曲線の傾き0となる点を導出することにより，エッジを抽出する．

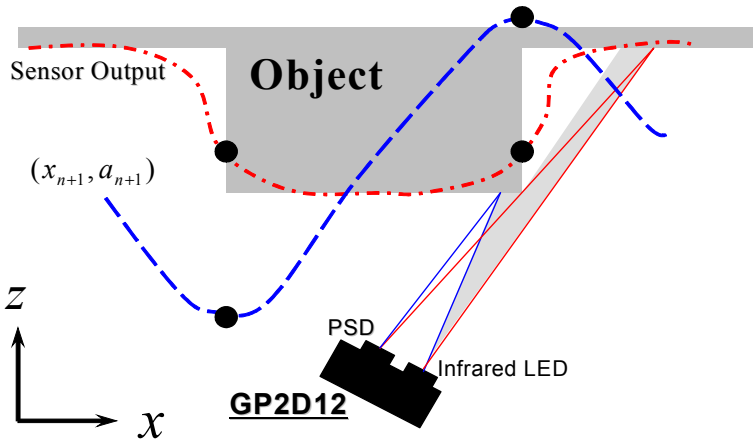


Fig 4.3 Extraction of Edge

第 5 章 実験

5.1 実験モデル - センサシステム

実験モデルを Fig.5.2 に示す．小型 $RC-SERVO$ を二個用いて能動関節を作成した．RC-servo はそれぞれピッチ・ヨー方向への首振りを可能にしている (Fig.5.2)．又，同原点座標になるように設計した．角度の情報は，RC-servo の指令値ではなく，絶対的な RC-servo 内のポテンショメータから得っている．制御は PIC（16F873）を使用して，結果を PC に出力する．

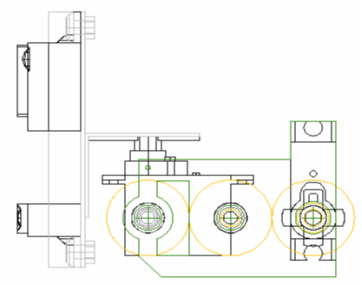


Fig 5.1 Design of Sensor System

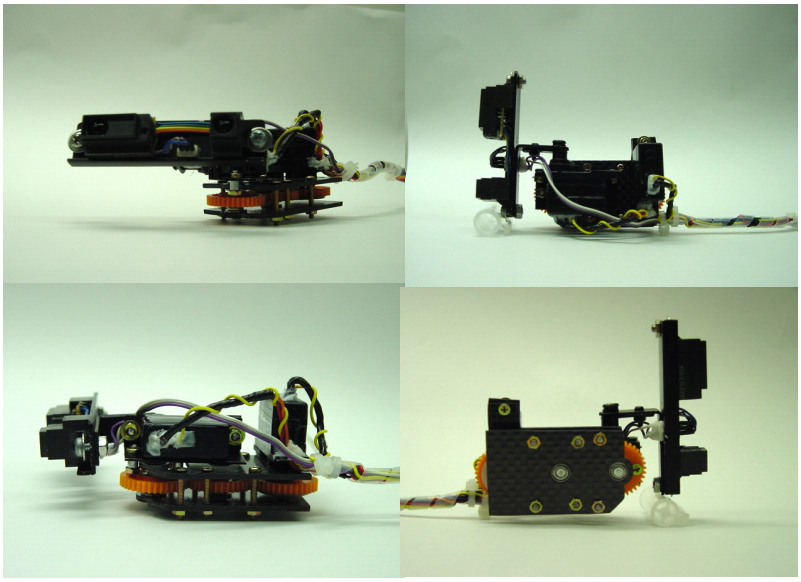


Fig 5.2 Mechanical of Sensor System

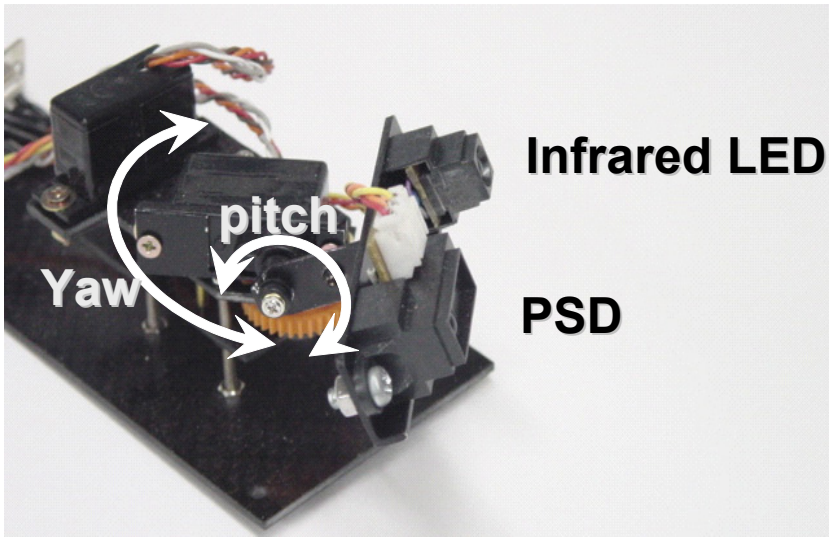


Fig 5.3 Operation of Sensor System

5.2　実験モデル - 移動ロボット

使用する連結クローラロボットの概観をFig.5.4に示す．このロボットは5つのクローラリンクから構成され，それぞれのクローラリンクはアクチュエータで駆動される能動関節により連結されている．また，クローラは各リンクの両サイドに装備され，別々のアクチュエータにより駆動される．この連結クローラロボットの駆動システムをFig.5.5に示す．制御系は階層制御構造とした．アクチュエータ2個に対し1つのサーボユニットを設け，シリアル通信によりPCと接続している．PCは各関節の目標角度と，目標速度をサーボユニットに送信し，それを受信したサーボユニットはPID制御によって，目標値に達成するよう制御を行う．Fig.5.6に提案したセンサシステムを装着した概観を示す．又，Fig.5.7に示す様にセンサシステムのリンク内部に，ピッチ・ロール方向を認識する傾斜角センサを設けた．これによって，障害物を絶対座標系で認識することが可能になっている．

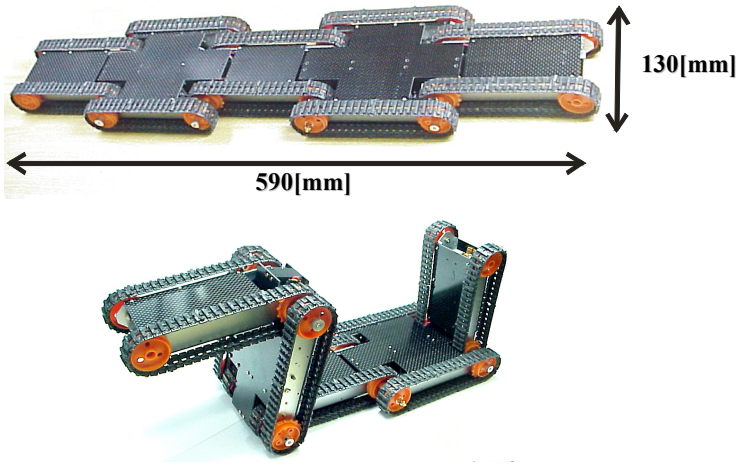


Fig 5.4 Operation of Sensor System

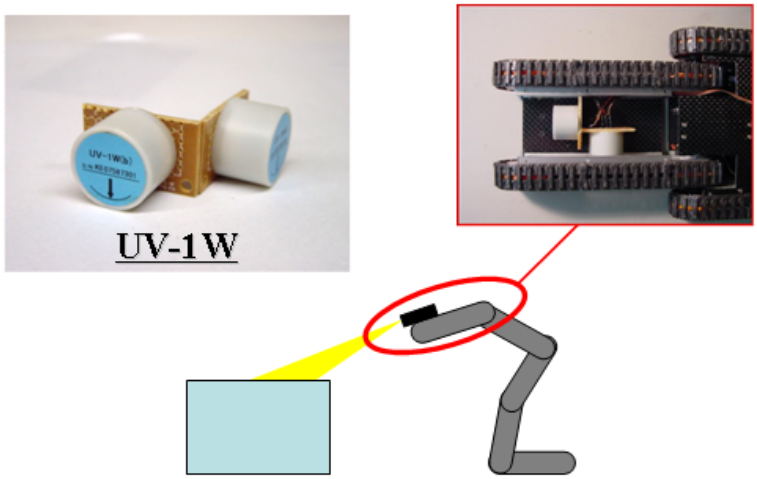


Fig 5.5 Overview of Robot with Angle of Gradient Sensor

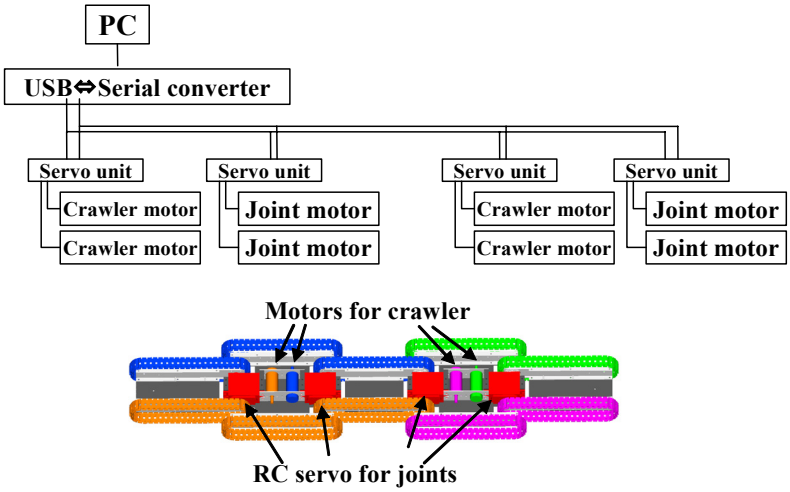


Fig 5.6 Control System,　Driving System

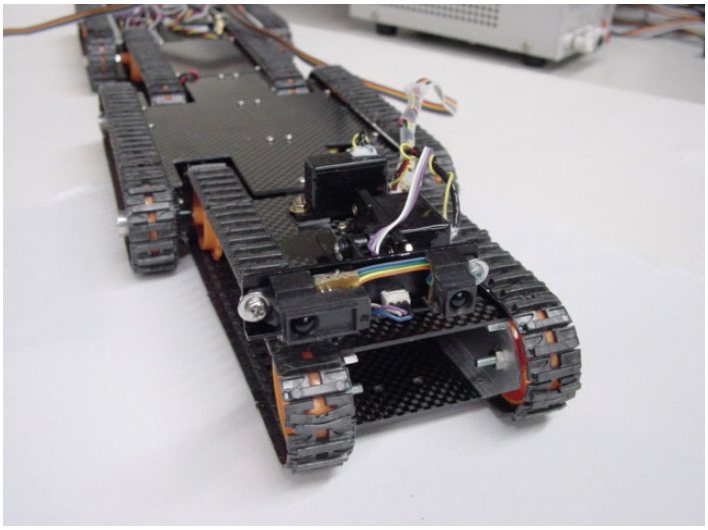


Fig 5.7 Overview of Robot with Sensor System

5.3 実験 1

5.3.1 実験方法

Fig.5.8 のように障害物として障害物（ダンボール箱）を設置して計測した．障害物-実験モデル間の距離 70[cm]，障害物の高さ 25[cm]，幅 28[cm]，奥行 38[cm] である．又，センサは床から 22[cm] の位置にした．

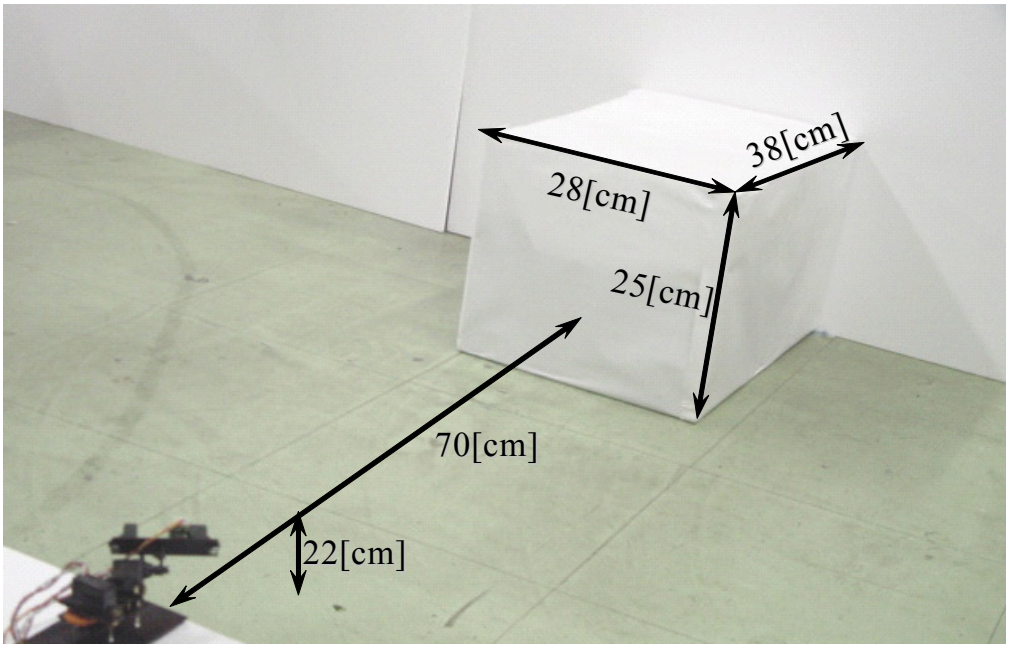


Fig 5.8 Overview of Experiment

5.3.2 実験結果

スキャンした結果を Fig.5.9，5.10，5.11 に示す．Fig.5.9 は，三次元形状を容易に認識できるよう *OpenGL* を使って結果を示した．図中の点がセンサから得られた障害物の形状である．箱状のものが実際の障害物である．Fig.5.10，5.11 はスキャン結果の断面図を表した．グラフ中の灰色の長方形は，実際の障害物形状を表した．Fig.5.10 は，床に対して，平行に高さ $22 \pm 2[cm]$ の横断面図，Fig.5.11 は床に対して垂直にセンサ中心 $\pm 2[cm]$ の縦断面図である．やはり，エッジ部は検出できないが，提案した方法を用いることにより，物体の幅・高さを検出することができている．

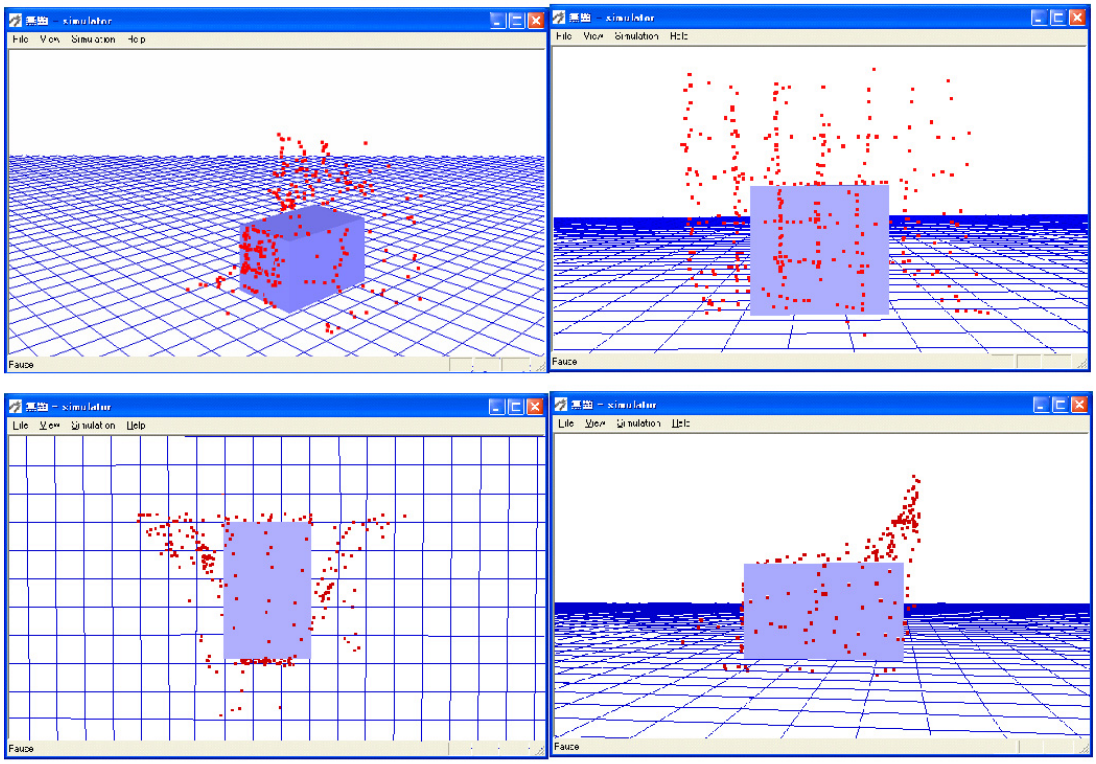


Fig 5.9 Obstacle Derived by Sensor System

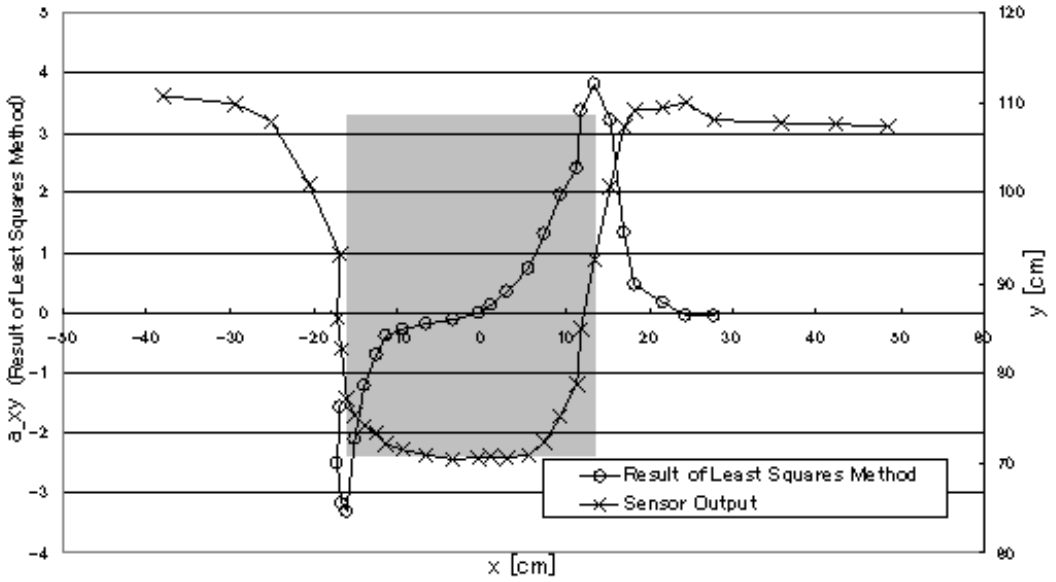


Fig 5.10 Section of Side Derived by Sensor System

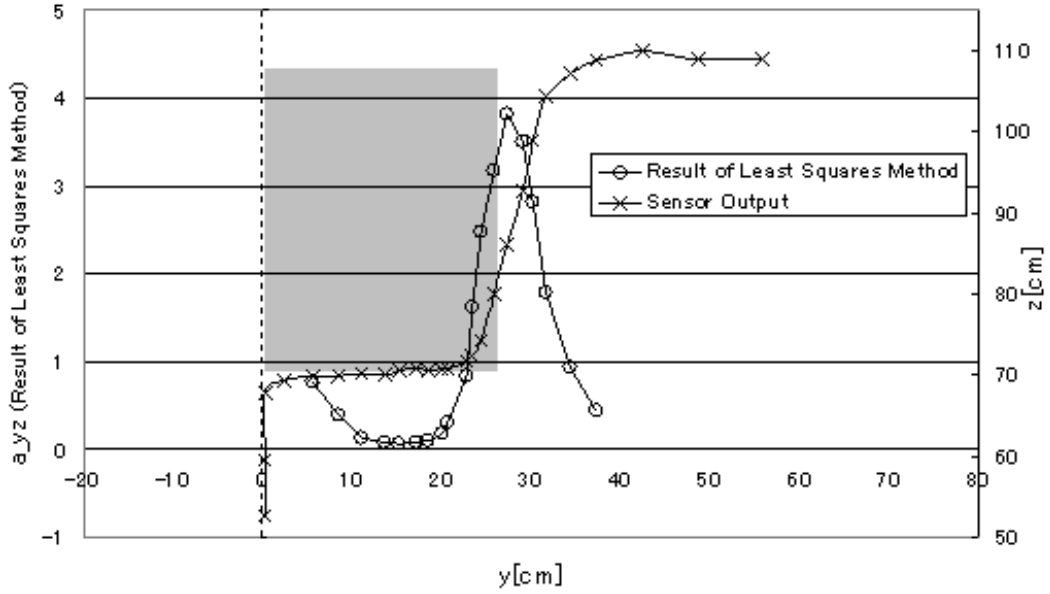


Fig 5.11 Section of Length Derived by Sensor System

5.4 実験 2

5.4.1 実験方法

Fig.5.12 のように障害物としてダンボール箱を設置し，ロボットの自由度を使用して高い位置から計測した．障害物 - 実験モデル間の距離 70[cm]，障害物の高さ 25[cm]，幅 28[cm]，奥行 38[cm] である．又，センサは床から 35[cm] の位置である．

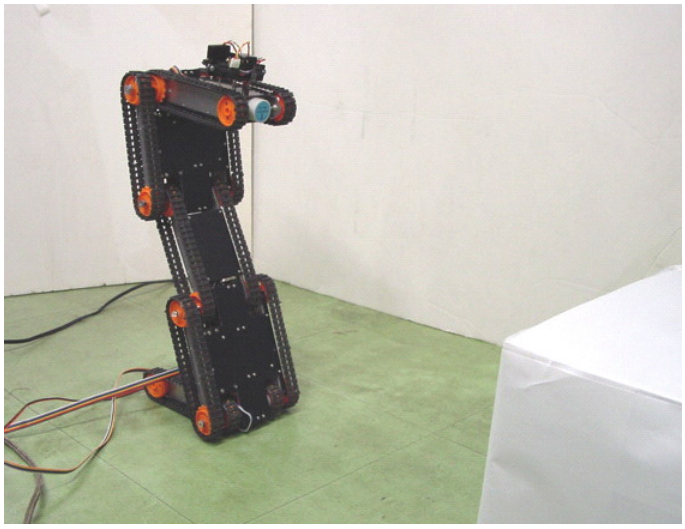


Fig 5.12 Overview of Experiment

5.4.2 実験結果

スキャンした結果を Fig.5.13，5.14 に示す．三次元形状を容易に認識できるよう $OpenGL$ を使って結果を示した．図中の点がセンサから得られた障害物の形状である．箱状のものが実際の障害物である．

Fig.5.14 は，ロボットの自由度を使用したスキャン結果における，床に対して垂直にセンサ中心± 2$[cm]$ の縦断面図である．グラフ中の灰色の長方形は，実際の障害物形状を表した．ロボット自身の自由度を活用することで，実験 1 と違い盲点部 (障害物上面) を計測できていることが確認できた．

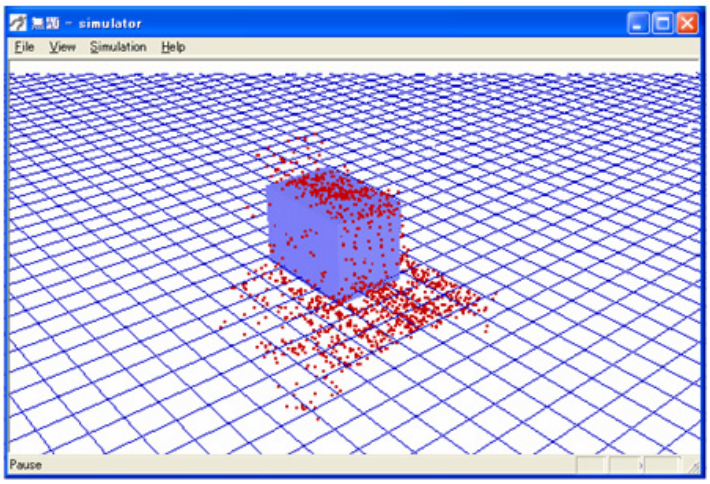


Fig 5.13 Obstacle Derived by Sensor System

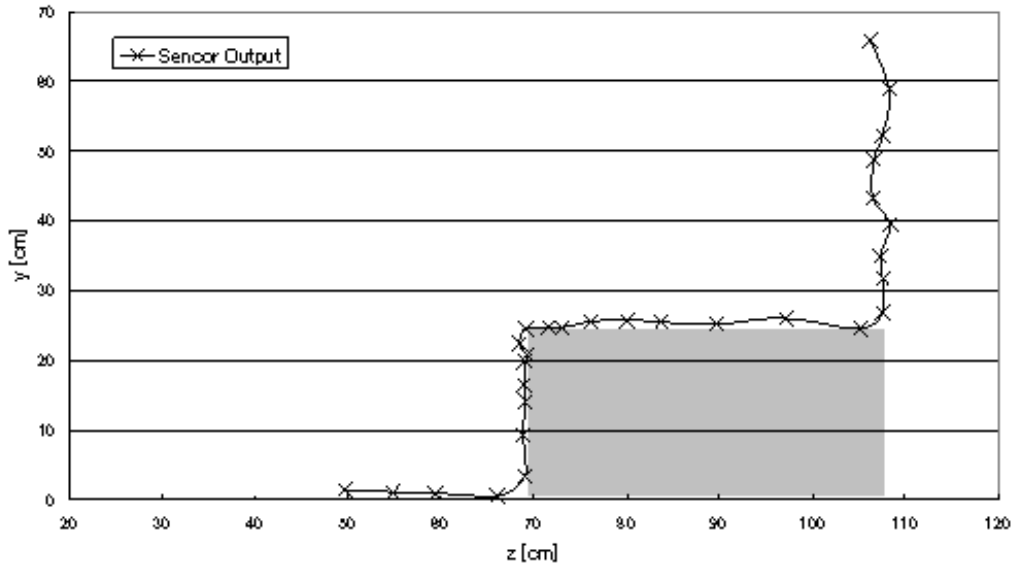


Fig 5.14 Section of Length Derived by Sensor System

第 6 章 結論

本稿では，小型移動ロボットのための三次元位置測定システムを提案した．センサシステムの構成とセンシング方法を示し，実験モデルよりその結果について報告した．この実験により提案したセンサシステムの有用性が確認できた．又，ロボット自身の自由度を最大限に活用することで，センサ単体では測定不可能な範囲も測定可能範囲にすることができ，測定可能範囲を大きく拡大することが可能となった．

今後の課題として，実環境での運用に当たりロボット走行中におけるセンシングが挙げられる．

謝辞

まず，小林尚登教授に感謝の意を申し上げます．教授には，研究の知識とアイデアを提供して頂いただけでなく，様々な面でご指導いただきました．

中村秀男先生には，予算の面において多大な労力を費やしていただき大変感謝しております．

3年間川端先生，稲垣先生，樹野先生, には大変お世話になりました．それぞれ正規の勤務地における活動がお忙しいにもかかわらず，わざわざ法政大学まで足を運んでいたき，われわれの研究について親身になって相談に乗っていただきありがとうございます．特に，川端先生には様々なアイデアの提供，論文の構成，発表の練習等，言葉には表せないほどのご迷惑をおかけしました．稲垣先生，樹野先生は，豊富な知識と経験でご指導アドバイスをいただきました．

最後に，本研究を共に励んできた先輩後輩の皆様に対して，感謝の意を申し上げて本論文の結びとさせていただきます．

参考文献

[1] 石田，永谷，五福：“不整地移動ロボットのための３次元自己位置推定と環境地図の構築”，第十二回日本ロボット学会学術講演会　論文集．1L2a，2003.

[2] 山浦, 鎌田, 平, 山崎：“可動型赤外線センサを用いた自律移動ロボットの地図生成システムの設計と実装”, ロボティクスメカトロニクス講演会05,1A2 S 023, 2005.

[3] E．Schulenburg，T．Weigel：“Self-Localization in Dynamic Environments based on Laser and Vision Data”，Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol.1, pp.998-1004, 2003.

[4] C.Drocourt, L.Delahoche, B.Marhic, A.Clerentin: “Simultaneous localization and map construction method using omnidirectional stereoscopic information.”, Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Vol.1, pp.894-899, 2002.

[5] F.J.Toledo, J.D.Luis, L.M.Tomas, M.A.Zamora, H.Martines: “Map building with ultrasonic sensors of indoor environments using neural networks.”, Proceedings of the 2000 IEEE International Conference on Systems, Man and Cybernetics, Vol.2, pp.920-950, 2000.

[6] N．Achour, R．Toumi: “Building an Environment Map Using a Sweeping System Based on a Single Ultrasonic Sensor.”, Proceed-

ings of the 2002 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp.1329-1333, 2002.

[7] T. Tsubouchi, A. Tanah, A. Ishioka, M. Tomono and S. Yuta, "A SLAM Based Teleoperation and Interface System for Indoor Environment Reconnaissance in Rescue Activities", Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp.1096-1102,2004.

[8] T. Tsubouchi, A. Tanah, A. Ishioka, M. Tomono and S. Yuta, "Construction of the 3-dimensional Maps for the collaps environment presentation in rubble by integration a range sensor and a image sensor", Proc. JSME conference on Robotics and Mechatoronics ROBOMEC'04, 2P2-H-23 (2004-06),2004.

[9] S. thrum. et Al," A System for Volumetric Robotic Mopping of Abandoned Mines," Proc. of ICRA2003,2003

[10] S.H.Park，J.R.Kim："Develpment of a Practical Sensor Fusion Module for Environment Modeling"，IEEE/ASME'Advanced Inteligent Mechatronics (AIM 2003)

[11] M. Tomono, "Map building and Global Localization for a Mobile Robot using LRF Scan Matching", Proc. of the 9th Robotics symposia. pp. 32-37, 2004.

[12] 馬場清太郎："トランジスタ技術SPECIAL増刊　OPアンプによる実用回路設計" CQ出版，2001

[13] 岡本由夫："OPアンプ回路の設計" CQ出版，2004

第 A 章 付録：回路構成

A.1 計算処理構成

今回作成したセンサシステムの回路構成をを Fig.A.1 に示す．PC との通信に時間がかかるため，RC-servo 制御・センサ値を PC に送信の，２つのマイコンを用意した．マイコン同士は，能動関節動作と PC へ通信のタイミングを通信している．Fig.A.1 に示すように，処理高速化のために，通信と能動関節動作をずらしてある．能動間接の角度は，時間のずれを防止するために RC-servo の指令値ではなく，RC-servo に内蔵されているポテンショメータから値をとっている．

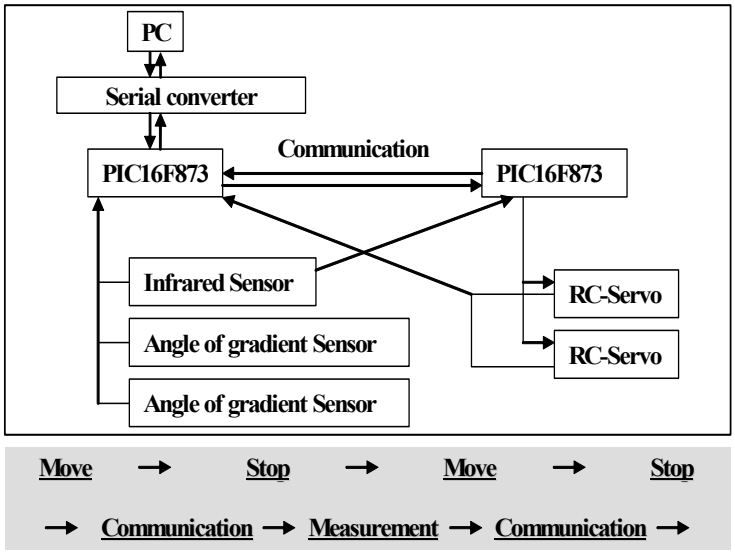


Fig A.1 Processing Composition

A.2　センサ，RC-servo 規格表

本稿で使用した赤外線距離計 ($GP2D12$)・傾斜角センサ ($UV-1W$) の規格表を Fig.A.2，A.3，A.4 に示す (シャープ株式会社・株式会社緑測器 HP より引用)．又，RC-servo($PICO-BB$) 規格表 (GWS 社　HP より引用)・RC-servo に内蔵されてるポテンショメータの位置を Fig.A.5，A.6 に示す.

A.3　センサシステム回路図

提案したセンサシステムの回路図を Fig.A.7 に示す.

A.4　センサシステムのプログラム

提案したセンサシステムのプログラムを示す．PC 側では，“ Microsot visual　C++ ”を用いた.

PC 用プログラム：sensor.cpp

PIC 用プログラム：servo.c(RC-servo 制御用)，link.c(通信用)

SHARP

GP2D12/GP2D15　　測距センサユニット

普及型測距センサユニット

■ 概　要

シャープ測距センサGP2D12/GP2D15は、PSD(*)、赤外発光ダイオード、信号処理回路を一体化した普及型の測距センサユニットです。

反射物の色、反射率、周囲の明るさによる影響を受けにくく、精度良い測距が可能です。

*PSD:Position Sensitive Detector

■ 特　長

（1）反射物の色、反射率による影響を受けにくい

（2）距離出力型と距離判定型をﾗｲﾝｱｯﾌﾟ

距離(ｱﾅﾛｸﾞ電圧)出力型：GP2D12

検出距離：10cm～80cm

距離判定型：GP2D15

判定距離：24cm

（判定距離は10～80cm

にてカスタム対応可）

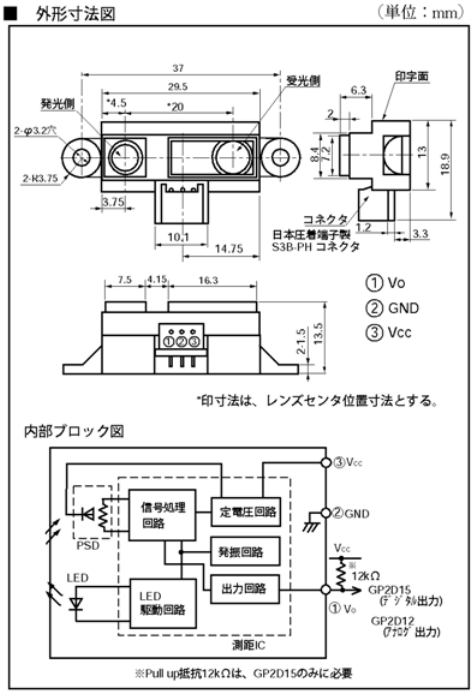
（3）外部制御用回路が不要。ﾏｲｺﾝに直結可能

（4）普及型

■ 用　途

（1）民生機器、ＯＡ機器等の各種物体検出、人体検出用

■ 外形寸法図　（単位：mm）



*印寸法は、レンズセンタ位置寸法とする。

内部ブロック図

※Pull up抵抗12kΩは、GP2D15のみに必要

■ 絶対最大定格　（Ta＝25℃、Vcc＝5V）

項　目		記号	定　格	単位	備　考
電源電圧		Vcc	-0.3 ～ +7	V	-
出力端子電圧	GP2D12	Vo	-0.3 ～Vcc +0.3	V	-
	GP2D15	Vo	-0.3 ～Vcc +0.3	V	オープンコレクタ出力
動作温度		Topr	－10 ～ ＋60	℃	-
保存温度		Tstg	－40 ～ ＋70	℃	-

（おことわり）
・本資料に掲載されている製品をご使用の際には、必ず最新の仕様書をご用命のうえ、その内容をご確認頂きますようお願いします。掲載製品につき、仕様書に記載されている絶対最大定格や使用上の注意事項等を逸脱して使用され、万一掲載製品の使用機器に瑕疵が生じ、それに伴う損害が発生しましても、弊社はその責を負いませんのでご了承下さい。なお、本資料に関してご不明な点がございましたら、事前に弊社販売窓口までご連絡頂きますようお願い致します。
・本製品は開発機種ですので、製品改良のため予告なしに仕様の一部を変更することがあります。
（インターネットへの公開）
・弊社オプトデバイス/パワーデバイスのデータをインターネット上で公開しています。(アドレス　http://www.sharp.co.jp/ecg/)

SHARP

技応970501－A

Fig A.2 Outline of GP2D12 (1)

SHARP

GP2D12/GP2D15 測距センサユニット

■ 動作電源電圧

記号	定 格	単位	備 考
Vcc	4.5 ～5.5	V	-

■ 電気的光学的特性

GP2D12電気的光学的特性 （Ta＝25℃、Vcc＝5V）

項 目	記号	条 件	最小値	標準値	最大値	単位
測距範囲	ΔL	- (注1)	10	-	80	cm
出力端子電圧	VO	L＝80cm (注1)	0.25	0.4	0.55	V
出力電圧差	ΔVO	L変化量(80cmから10cm)時の出力変化量 (注1)	1.75	2.0	2.25	V
平均消費電流	ICC	L＝80cm (注1)	-	33	50	mA

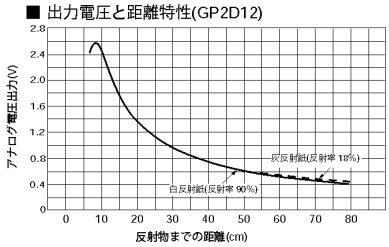
GP2D15電気的光学的特性 （Ta＝25℃、Vcc＝5V）

項 目	記号	条 件	最小値	標準値	最大値	単位
測距範囲	ΔL	(注3)	10	-	80	cm
出力端子電圧	VOH	High時出力電圧 (注1)	Vcc-0.3	-	-	V
	VOL	Low時出力電圧 (注1)	-	-	0.6	V
出力の距離特性	L	(注1) (注2) (注4)	21	24	27	cm
平均消費電流	ICC	-	-	33	50	mA

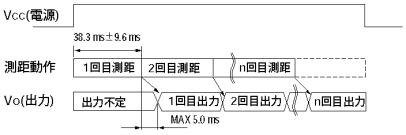
L：反射物までの距離

(注1) 使用反射物：白色紙(コダック社製グレーチャートR-27・白色面、反射率90％)
(注2) センサにて、出力切り替わり距離がL＝24cm±3cmとなる様に調整の上出荷。
(注3) 測距可能範囲(センサ光学系の測距可能範囲)
(注4) 出力切り替わりはヒステリシス幅を持っており、Voにて規定する距離特性は非検出(出力L)から検出(出力H)に切り替わる距離とする。

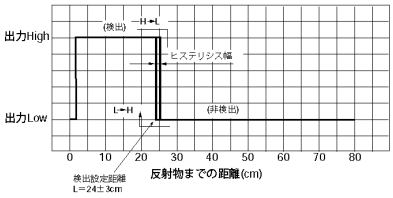
■ 出力電圧と距離特性(GP2D12)



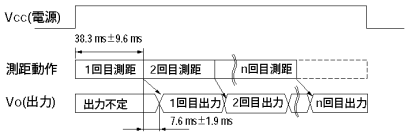
■ GP2D12の出力タイミングチャート



■ 出力電圧と距離特性(GP2D15)



■ GP2D15の出力タイミングチャート



この技術資料の内容は、1998年4月現在のものです。

SHARP

技応970501－A

Fig A.3 Outline of GP2D12 (2)

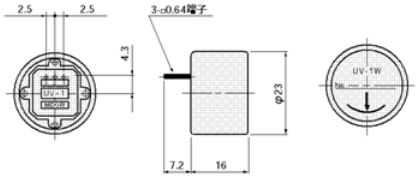
Blue Pot　無接触形（磁気抵抗素子使用）傾斜角センサ

UV-1W

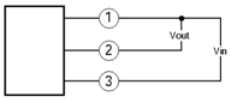
■特　長

・小型・軽量・広角度センサ
・プリント基板、コネクタへ簡単に接続可能
・軸受ボールベアリング
・ダンパー油使用で振動の影響が少ない

■外形図



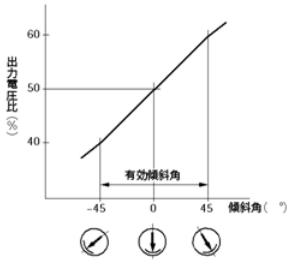
■結線図



■電気的仕様

有効傾斜角	±45°
傾斜角感度	2°以下（ヒステリシスを含む）
出力感度	2.5% Vin / 10°以上
インデックスポイント	50±5%
基点直線性	±3%FS（FS=90°）
入力インピーダンス	15kΩ±30%（25℃）
印加電圧	DC10V以下
負荷抵抗	5MΩ以上
絶縁抵抗	100MΩ以上 DC 500V
耐電圧	AC500V 1分間

■出力特性図



■機械的仕様

傾斜角	90°以上
応答時間	約0.4秒（ステップ応答）
質量	約10g

■環境特性

使用温度範囲	0～60℃
保存温度範囲	−40～60℃

特殊仕様
・直並列抵抗による温度補償抵抗内蔵可能

温度補償範囲と温度特性

温度補償記号	TCA
温度範囲	0～60℃
0°位置	±0.3°
±10°位置	±0.5°

■取付方法（例）

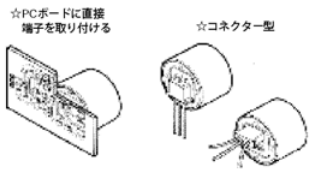


Fig A.4 Outline of UV-1W

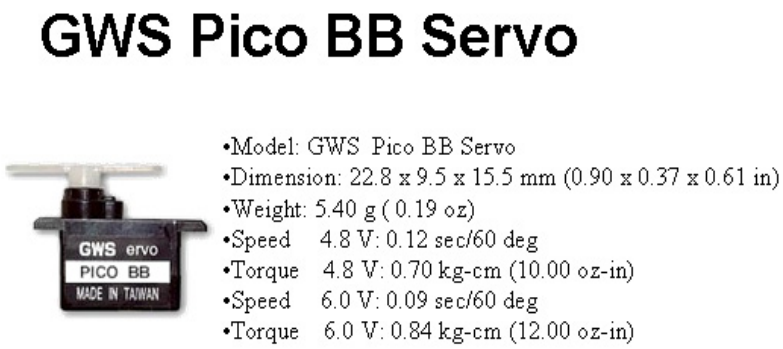


Fig A.5 Outline of PICO BB

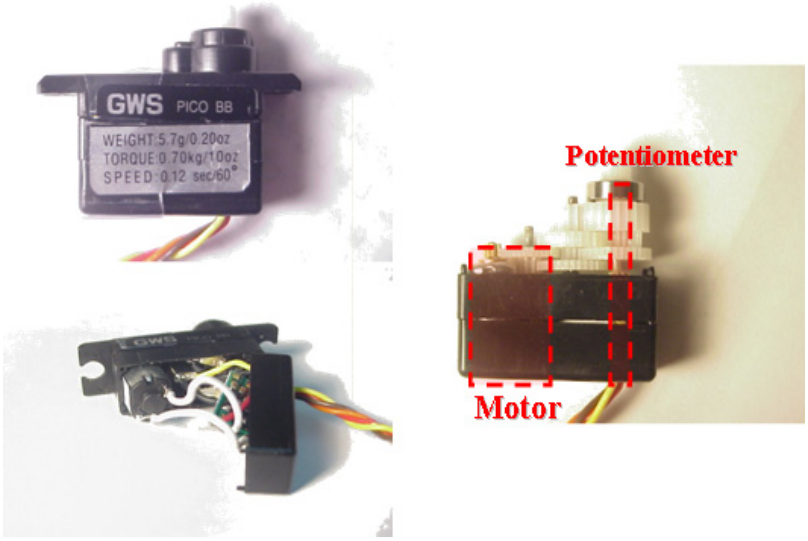


Fig A.6 Overview of PICO BB

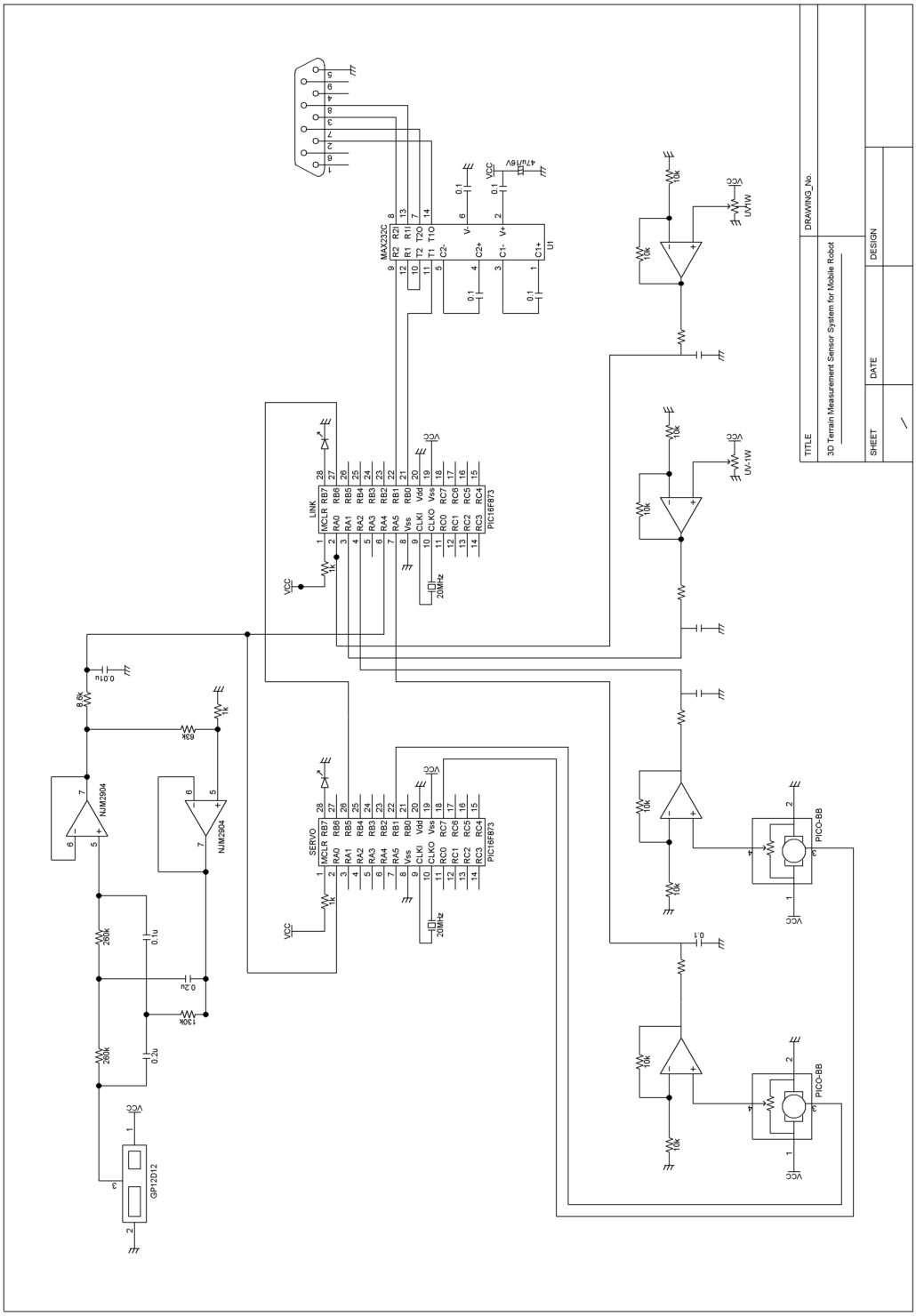


Fig A.7 Circuit of Sensor System

PC用プログラム：sensor.cpp

```
#include <windows.h>
#include <iostream.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <time.h>
#include <gsl/gsl_fit.h>

#define DATA_SIZE 16
#define SAVE_SIZE 8

void ErrorHandler(char *message, DWORD error)
{
        cout << message << endl;
        cout << "エラー番号 = " <<error << endl;
        ExitProcess(1);
]

int main()
{
                        printf("MIYASKY sencor Ver.1.0¥n¥n");
   FILE *fp;
          fp=fopen("01.dat","w");

        int p0,p1,x0,x1,y0,y1,s1,s2,s3,s4,count=0,pool_count=0,load_count=0,save_count=0;
        double psd_cm,psd_x_deg,psd_y_deg,s_rool_deg,s_pitch_deg,a_now,sub_load_count=0;
        double psd_x_rad,psd_y_rad,s_pitch_rad,s_rool_rad,x,y,z;
        long psd_16,psd_x_16,psd_y_16,s_rool_16,s_pitch_16;

        double b,a, cov00, cov01, cov11, sumsq;

        double x_pool[DATA_SIZE]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};          //最小二乗法 x 保管
        double y_pool[DATA_SIZE]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};          //最小二乗法 y 保管
        double z_pool[DATA_SIZE]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};          //最小二乗法 y 保管

        double a_pool[SAVE_SIZE]={0,0,0,0,0,0};//aの保存先

clock_t t1,t2;
        t1=clock();
        BOOL   fSuccess;
        DCB         dcb;
        HANDLE hComm;    /* シリアルポートのハンドル */
        DWORD dwErrors;  /* エラー情報 */
   COMSTAT ComStat; /* デバイスの状態 */
   DWORD dwCount;   /* 受信データのバイト数 */
   BYTE pszBuf[100];/* 読み出しデータバッファ */

   DWORD dwRead;    /* ポートから読み出したバイト数 */

        COMMTIMEOUTS timeouts;



 hComm = CreateFile(
                "COM1",
                        GENERIC_READ | GENERIC_WRITE,
                 0,
                        NULL,
                        OPEN_EXISTING,
                        FILE_ATTRIBUTE_NORMAL,
                NULL
                                );
```

```
if (hComm == INVALID_HANDLE_VALUE)
        {
                //シリアルポートがオープンできないとき
                printf("シリアルポートオープンできません。¥n");
        ]
        else{
                //シリアルポートがオープンできた
                printf("シリアルポートオープンしました。¥n");
        ]

        fSuccess = GetCommState(hComm, &dcb);
        if(!fSuccess)
        {
                CloseHandle(hComm);
        ]

        dcb.BaudRate = CBR_38400;
        dcb.ByteSize = 8;
        dcb.Parity          = NOPARITY;
        dcb.fParity = TRUE;
        dcb.StopBits = ONESTOPBIT;
        dcb.fBinary = TRUE;

        fSuccess = SetCommState(hComm, &dcb);


        /*ポートのタイムアウトの設定*/
  timeouts.ReadIntervalTimeout =MAXWORD;
        timeouts.ReadTotalTimeoutMultiplier = 0;
        timeouts.ReadIntervalTimeout = 0;
        timeouts.WriteTotalTimeoutMultiplier = 0;
  timeouts.WriteTotalTimeoutConstant = 0;
        SetCommTimeouts( hComm , &timeouts );

        EscapeCommFunction( hComm , SETDTR );

  if (!fSuccess)
        {
                //DCBが設定出来ないとき
                printf("DCBを設定出来ません。¥n¥n");
                CloseHandle(hComm);
        ]
        else{
                //DCBが設定できたとき
                printf("DCBが設定できました。¥n");
                                printf("準備完了しました。¥n");
                                                printf("通信開始してください。¥n¥n");
        ]

        t1=clock();

printf("Enter キーを押すと終了します。¥n"); /* メッセージを表示します。*/



        while(!kbhit()){
                t2=clock();

                 while(count!=3){
                ClearCommError(hComm, &dwErrors, &ComStat);
 dwCount = ComStat.cbInQue;
 fSuccess = ReadFile(hComm, pszBuf, 10, &dwRead, 0);

 if(!fSuccess)
```

```
        ErrorHandler("ReadFileでエラー発生",GetLastError());
 pszBuf[dwRead] = '¥0';

        count++;
 //---------------- センサ値受信  8bit+8bit --------------------


 p0=pszBuf[0];  //PSDセンサ出力
 p1=pszBuf[1];

 y0=pszBuf[2];  //y軸周り角度-サーボ
 y1=pszBuf[3];

 x0=pszBuf[4];  //x軸周り角度-サーボ
 x1=pszBuf[5];

 s1=pszBuf[6];  //傾斜角センサ-ロール
 s2=pszBuf[7];

 s3=pszBuf[8];  //傾斜角センサ-ピッチ
 s4=pszBuf[9];


//---------------- センサ値結合  8bit → 16bit --------------------

        psd_16 = psd_16+p1*256+p0;              //  PSDセンサ出力[16bit]
        psd_y_16 = psd_y_16+y1*256+y0;                  //y軸周り角度-サーボ[16bit]
        psd_x_16 = psd_x_16+x1*256+x0;                  //x軸周り角度-サーボ[16bit]
        s_rool_16 =s_rool_16+s2*256+s1;         //傾斜角センサ-ロール[16bit]
        s_pitch_16 =s_pitch_16+ s4*256+s3;      //傾斜角センサ-ピッチ[16bit]

                }

count=0;

 psd_16 = psd_16/3;             //  PSDセンサ出力[16bit]
        psd_y_16 = psd_y_16/3;                  //y軸周り角度-サーボ[16bit]
        psd_x_16 = psd_x_16/3;                  //x軸周り角度-サーボ[16bit]
        s_rool_16 = s_rool_16/3;            //傾斜角センサ-ロール[16bit]
        s_pitch_16 = s_pitch_16/3;          //傾斜角センサ-ピッチ[16bit]


//---------------- センサ値変換  16bit → SI単位 --------------------

        psd_cm = 27704457.574*pow(psd_16,-1.95755222)+2.7;              //  PSDセンサ出力[cm]
        psd_y_deg = -0.2166* psd_y_16 +148.27+3.6-20;                           //y軸周り角度-サーボ[deg]
        psd_x_deg = -0.4332* psd_x_16 +164.7412;                        //x軸周り角度-サーボ[deg]
        s_rool_deg = -0.22* s_rool_16 +144.33;                                  //傾斜角センサ-ロール[deg]
        s_pitch_deg = -0.22* s_pitch_16 +148.0683;                      //傾斜角センサ-ピッチ[deg]

//--------------  パラメータ初期化  ----------------------
        psd_16 = 0;                 //  PSDセンサ出力[16bit]
        psd_y_16 = 0;                       //y軸周り角度-サーボ[16bit]
        psd_x_16 = 0;                       //x軸周り角度-サーボ[16bit]
        s_rool_16 =0;               //傾斜角センサ-ロール[16bit]
        s_pitch_16 =0;    //傾斜角センサ-ピッチ[16bit]


        if(psd_cm < 120 && psd_cm > 50){
//---------------- deg → rad 単位変換 --------------------

        psd_x_rad = psd_x_deg*(3.14/180);
        psd_y_rad = psd_y_deg*(3.14/180);
        s_pitch_rad = s_pitch_deg*(3.14/180);
```

```
        s_rool_rad = s_rool_deg*(3.14/180);

//--------------- 座標変換 ----------------------------

x=psd_cm*cos(psd_x_rad)*sin(psd_y_rad);
y=psd_cm*sin(psd_x_rad)+22;
z=psd_cm*cos(psd_y_rad)*cos(psd_x_rad);

//--------------- 最小二乗法計算---------------------
x_pool[pool_count]=x;
y_pool[pool_count]=y;
z_pool[pool_count]=z;
pool_count++;

if(pool_count==16)pool_count=0;


gsl_fit_linear (x_pool, 1, z_pool, 1, DATA_SIZE, &b, &a, &cov00, &cov01, &cov11, &sumsq);

sub_load_count=pool_count-8;

if(sub_load_count<0)load_count=sub_load_count+16;
else load_count=sub_load_count;

x=x_pool[load_count];
y=y_pool[load_count];
z=z_pool[load_count];

//------------- 出力命令 -------------------------


printf,"%3.4f      %3.4f     %3.4f     %3.4f¥n",x,y,z,a);
fprintf(fp,"%3.4f  %3.4f     %3.4f     %3.4f¥n",x,y,z,a);

]
  EscapeCommFunction(hComm,CLRDTR);
]
  fclose(fp);
  CloseHandle(hComm);
  return 0;

]
```

PIC用プログラム：servo.c

```
/*************************************************************/
//servo.c
//パルス幅データ範囲は1～624
/*************************************************************/
#include <16F873.h>
#fuses HS,NOWDT,NOPROTECT,NOPUT,NOBROWNOUT,NOWRT,NOLVP
#device ADC=10
#use delay(clock=20000000)
#byte RC = 7

#define ch1 PIN_C7
#define ch2 PIN_B1

int state=0,ac=0,bc=0,ab=0;  //状態切り替えカウンタ
long pwm1=300;  //サーボ１のｐｗｍデータ1~624
long pwm2=300;  //サーボ１のｐｗｍデータ1~624
long pwm_data=0;
long a=1,b=1,i,s;
long s_pitch_16;
int c=0,ki,kii,ai,aii,e=1,d;
int ofst,s_width,ps_width,spt,count;


//タイマのクリアと変数のカウントアップ////////////////
#INT_CCP2
void ccp2_int(){
   switch(state){
      case 0 : state=state+1;//カウントアップ
               CCP_2=pwm1+562;
               output_high(ch1);
               break;
      case 1 : state=state+1;//カウントアップ
               CCP_2=740-pwm1;
               output_low(ch1);
               break;
      case 2 : state=state+1;//カウントアップ
               CCP_2=562+pwm2;
               output_high(ch2);
               break;
     case 3 : state=state+1;//カウントアップ
               CCP_2=740-pwm2;
               output_low(ch2);
               break;
      case 4 : state=0;//カウントアップ
               CCP_2=9948; //16msec
               break;
   }
   set_timer1(0x0000);                 // TMR1クリア
}



/*************************************************************************/
/*************************  main文  **************************************/
/*************************************************************************/
main()
{
      setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
      //setup_ccp1(CCP_COMPARE_INT);        // CCP1コンペアマッチ割込み設定
      setup_ccp2(CCP_COMPARE_INT);        // CCP2コンペアマッチ割込み設定
      set_timer1(0x0000);                 // TMR1クリア
      //CCP_1 = 312;                        // 0.5ms
      CCP_2 = 625;                        // 1mS
      // set_tris_c(0x00);                   //RC 7-0:OUT
```

```
    enable_interrupts(INT_CCP1);        //CCP1コンペアマッチ割込み許可
    enable_interrupts(INT_CCP2);        //CCP2コンペアマッチ割込み許可
    //enable_interrupts(INT_RDA);
    enable_interrupts(GLOBAL);          //全設定割込み許可


    setup_adc_ports(ALL_ANALOG);  //RA0,RA2,RA3アナログ入力
        setup_adc(ADC_CLOCK_DIV_32);


//------------ パラメータ初期設定 ------------

ac = 1;
bc = 1;ab = 0;
a = 389 ;
b = 580;
d=1;
ofst =40;        //行+列移動　オフセット値
s_width = 8;      //サンプリング幅 障害物あり
ps_width = 30;    //サンプリング幅 障害物なし
spt = 200;         //サンプリングタイム


//-------------------------------------------------------------------------
//---------------------- サーボ動作指令文 --------------------------------
//-------------------------------------------------------------------------

while(1)
   {

   set_adc_channel(1);  // サーボ　ｙ軸回りポテンショ値受信
              delay_us(30);
              s_pitch_16 = read_adc();      //  →  s に格納

     //-------------------------------------------------
     //---------- 横方向スキャン運動 ----------------------
     //-------------------------------------------------

     if(d==1)
        {

              set_adc_channel(0);  // サーボ　ｙ軸回りポテンショ値受信
              delay_us(30);
              s = read_adc();      //  →  s に格納

              if(a >= 530) c = 0;
              if(a <= 220) c = 1;

              if(e != c)
                  {
                           if(b >= 595 && a <= 240)  d = 0;

                           else
                             {
                                b = b + ofst;
                                delay_ms(40);
                             }
                  }

            //-------------------------------------------------------
            //---------- サンプリング幅 オフセット命令 ---------------
            //-------------------------------------------------------
```

```
        //-------------- 障害物検知　あり --------------------

           if(s < 550 || s > 870)
               {
                 if(c == 1){a = a +ps_width;}
                 else{a = a - ps_width;}
               }

        //-------------- 障害物検知　なし --------------------

           if(s > 550 && s < 870)
              {
                 if(c == 1){a = a + s_width;}
                 else{a = a - s_width;}
              }


        pwm1 = a;   //PWMパルス幅出力 縦方向
        pwm2 = b;   //PWNパルス幅出力 横方向
        e = c;

       delay_ms(spt);   //サンプリングタイム

    }

//-----------------------------------------------------
//---------- 縦方向スキャン運動 -----------------------
//-----------------------------------------------------

if(d == 0)
   {
      set_adc_channel(0);
      delay_us(30);
      s = read_adc();

      if(b >= 595)c = 1;
      if(b <= 100)c = 0;

      if(e != c)
          {
            if(a >= 510 && b < 200) d = 1;

            else
              {
               a = a + ofst;
               delay_ms(40);
              }
           }

          //-----------------------------------------------------------
          //----------- サンプリング幅 オフセット命令 ---------------
          //-----------------------------------------------------------

           //-------------- 障害物検知　あり --------------------

      if(s < 550 || s > 870)
          {
            if(c == 0) b = b + ps_width;
            else b = b - ps_width;
          }
           //-------------- 障害物検知　なし --------------------
```

```
            if(s > 550 && s < 870)
                {
                  if(c == 0){b = b + s_width;}
                  else{b = b - s_width;}
                }

             pwm1 = a;   //PWMパルス幅出力 縦方向 a
             pwm2 = b;   //PWMパルス幅出力 横方向 b
             e = c;

             delay_ms(spt);    //サンプリングタイム
       }
//----------タイミング信号発生ー----------------
     if(count==1)
       {
              output_high(PIN_B5);
              output_high(PIN_B7);
              count=0;
       }
       else
       {
              output_low(PIN_B5);
              output_low(PIN_B7);
              count=1;
       }
   }
}
```

PIC用プログラム：link.c

```
/*********************************************************/
//link.c
/*********************************************************/

#include    <16f873.h>
#fuses HS,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP
#device ADC=10
#use delay(CLOCK = 20000000)//クロック周波数指定
#use rs232(baud=38400,xmit=PIN_B0,rcv=PIN_B1,parity=N,bits=8)
#byte RC = 7


///// メイン関数
void main()
{
       long data;

long r1,r2,sp1,sp2,sl;
int r1a,r1b,r2a,r2b,sp1a,sp1b,sp2a,sp2b,sla,slb,timer=0,timer2=0,count=0;
 set_tris_c(0xff);
 output_float(PIN_B6);


   setup_adc_ports(ALL_ANALOG);     //ALL アナログ入力
   setup_adc(ADC_CLOCK_DIV_32);

while(1){

//-----------タイミング処理ー----------------

timer = input(PIN_B6);       //タイミング入力

   if(timer!=timer2)
      {
        timer2=timer;
        output_high(PIN_B7);


//--------計測--------------

               while(count!=3){
               count++;

                  set_adc_channel(0);  // センサ値取得 ロール
                  delay_us(20);
                   r1 = read_adc();

                  set_adc_channel(1);  // センサ値取得 ピッチ
                  delay_us(20);
                   r2 = read_adc();

                   set_adc_channel(2); // ｘ軸周り角度-サーボ
                  delay_us(20);
                   sp1 = read_adc();

                  set_adc_channel(3);  // ｙ軸周り角度-サーボ
                  delay_us(20);
                   sp2 = read_adc();

                   set_adc_channel(4); // PSDセンサ
                  delay_us(20);
                   sl = read_adc();
```

```
//--------PCに送信-------------

                sla = sl & 0x00FF;
                slb = sl >> 8;     //上位８bit取得
                      putc(sla);   // PSDセンサ
                      putc(slb);



                sp2a = sp2 & 0x00FF;
                sp2b = sp2 >> 8;     //上位８bit取得
                      putc(sp2a);    // ｙ軸周り角度-サーボ
                      putc(sp2b);



                      sp1a= sp1 & 0x00FF;
                sp1b= sp1 >> 8;     //上位８bit取得
                      putc(sp1a);   // ｘ軸周り角度-サーボ
                      putc(sp1b);


                r1a=r1 & 0x00FF;
                r1b=r1 >> 8;        //上位８bit取得
                      putc(r1a);    // センサ値取得 ロール
                      putc(r1b);

                r2a=r2 & 0x00FF;
                r2b=r2 >> 8;        //上位８bit取得
                      putc(r2a);    // センサ値取得 ピッチ
                      putc(r2b);

                  delay_ms(50);

                   }
              count=0;
     }
    else
     {
       output_low(PIN_B7);
     }
  }
}
```