TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Soheil Zavari

Sensors and Actuators Communication and Synchronization for a Mobile Manipulator

Master's thesis

Examiners: Professor Kalevi Huhtala
Dr. Reza Ghabcheloo
Examiner and topic approved by
the Faculty Council of the
Faculty of Engineering Science on
8 May 2013

Abstract

TAMPERE UNIVERSITY OF TECHNOLOGY
International Master's Degree Programme in Machine Automation
ZAVARI,SOHEIL: Sensors and Actuators Communication and Synchronization for a Mobile Manipulator
Master of Science Thesis, 50 pages
December 2013
Mechatronic Engineering
Examiners: Professor Kalevi Huhtala and Dr. Reza Ghabcheloo
Keywords: CAN open, EPOS controller, Servo motors, Real-time OS, Xenomai, SPI

Modern mobile manipulator hardware architecture is combination of mechanical, electrical, software and control units. Integrating numbers of mechanical and electrical components in the system rise the number of parameters to control. Hence the issues of controlling such systems to achieve the best performance is critical. A key aspect for this purpose is integration of sensors and actuators to provide a low level of mobile manipulator control and my thesis involve in providing a low level of control for iMoro mobile manipulator.

To operate such a mobile manipulator several intelligent components needs to be installed and cooperate at same time. Due to the application that iMoro made for, the platform is equipped with 8 actuators and number of sensors. Therefore control of such a system is complicated and demands accurate synchronization and communication among four legs.

Since iMoro robot is equipped with IMU, in following chapter IMU is calibrated and the process of providing meaningful data out of raw data explained by modeling the IMU. In addition, an equation is developed for robust calibration of IMU and camera.

Preface

This thesis work was part of project PURESAFE (Preventing hUman intervention for incREased SAfety in inFrastructures Emitting ionizing radiation) and it was done at department of Intelligent Hydraulics and Automation of Tampere University of Technology.

First of all, I feel honored to work under supervision Dr. Reza Ghabcheloo because of his great experience and suggestions in this field. I am also grateful from Prof. Kalevi Huhtala for his support. Furthermore I thank Reza Oftadeh who share his superb knowledge during my thesis period.

Finally I have gratitude to my father Siavash Zavari because of his efforts to support me and special thank to my kind-hearted mother Sharareh Zavari.
15.11.2013 Tampere
Soheil Zavari

List of Terms and Abbreviations

Notations	Description
AC	Alternating Current
API	Application Programming Interface
CAN	Controller Area Network
CiA	CAN in Automation
DC	Direct Current
DCM	Direction Cosine Matrix
DS	Device Profile
EC	Electronically Commutated
ECI	Earth-Centered Inertial
ECEF	Earth-Centered Earth-Fixed
EPOS	Electronic Positioning System
GPS	Globall Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IP	Internet Protocol
NED	North-East-Down
NMT	Network Management
OD	Object Dictionary
OS	Operating System
PDO	Process Data Object
RTDM	Real-Time Driver Module
UDP	User Datagram Protocol

CONTENTS

LIST OF TABLES

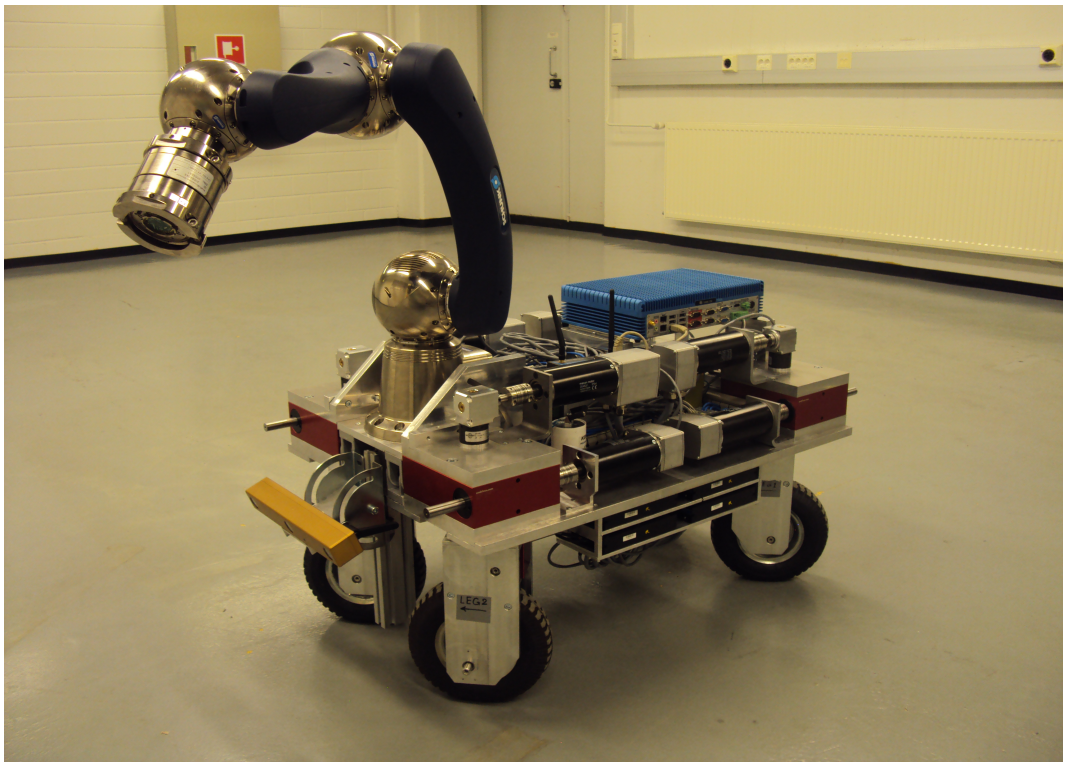
LIST OF FIGURES

Introdution

Mobile robot is machine able to move in different environments based on its locomotion mechanism. Locomotion mechanism determines the ability of the mobile robot to walk, slide, jump or fly. Wheeled locomotion is a common mechanism for flat surface with high efficiency and there has been variety of research in this area recently.

iMoro robot is a four wheel steered platform for mobile manipulation under investigation for high maneuverability in high speed motion. As it is demonstrated in the picture the dimension of iMoro is $655mm \times 335mm$ with 120 kg weight. For more detail refer to [1].

I interlaced IMU and encoder and 8 actuators and controller drivers in wise order within a CANbus base on design and expectation from system and provide a level of communication for sending command and receiving feedback within real-time environment (Xenomai OS). That implies the crucial role of timing to achieve better level of control. Also it is essential to make software interface for each sensors for communicating with Embedded PC.



iMoro 4 wheel steered mobile robot

Among different communication protocols, Controller Area Network is chosen for synchronization between all actuators. Thus CAN network among different components of system is implemented. Real-time software interfaces for sensors and message queue services for communication among several software modules are de-

veloped to provide a foundation for controlling iMoro in real-time environment and that is what explained in chapter 2.

Since iMoro is equipped Inertial Measurement Unit (IMU) on top of platform, the process of receiving meaningful data to estimate the position and velocity of robot body is done. Later in chapter 3 IMU characteristics is explained and a method for finding IMU-camera transformation matrix is presented.

1. THEORETICAL BACKGROUND

1.1 Inertial Sensors

Broad range of sensors used in mobile robots can measure from simple internal values like encoders or complicated terms from environment and interpret them for mobile robot like laser range scanner. In a general categorization, proprioceptive sensors measure internal states of the robot like rotary encoders or Inertial Measurment Unit (IMU) and exteroceptive sensors gain information from environment like camera.(See for instance Q. Zhang [2])

Proprioceptive sensors available in iMoro robot are incremental and absolute encoders, hall sensors, inertial measurement unit and exteroceptive sensors are vision sensors like camera, laser range scanner.

To choose a sensor for a specific type of application on a mobile robot, first you need to evaluate its performance. There are variety of terms that express sensor characteristic and performance, some of them are Dynamic range, Resolution, linearity, sensitivity, accuracy, error.

Individual sensors has range of errors that can interfere with making a right decision of robot. Thus it is essential to identify the type of errors individually and compensate them. Some errors are predictable and they can be modeled to exclude from system. Some errors has random nature and they are expressed with probabilistic terms and more complicated to calculate them. See section[3.2.4]

Sometimes to achieve more accuracy, output data from different sensors are combined and lead to more complete and accurate data. This process is called sensor fusion and there are several algorithm to perform sensor fusion like Kalman Filtering (For more detail refer to [3]). A key factor in this process is calibration between several sensors. See section[3.3] Some sensors express their output as a value in a standard range, on the other hand there are some sensors that need a coordinate frame to express their output like vision sensors or IMU (See for instance J. Vagany [4]). See section[3.2]

Output of IMU based on Inertial navigation equation can be expressed in different coordinate frames depends on which the form of equations change, some popular coordinate frame are : Inertial Navigation Frame, Earth Frame, Local Navigation Frame .

Earth-Centered Inertial Frame (ECI) denoted by i is fixed at center of earth. Z axis points along the earths axis of rotation from center to the north pole and Y axis lies 90 degree ahead of x axis in direction of rotation and both lie on equatorial plane. For more detail refer to[5] .

Earth-Centered Earth-Fixed Frame (ECEF) is indicated by e and is located at center of earth with z axis aligned with earth axis of rotation and it rotates with earth. X axis aligned with zero meridian (0 ° longitude) from center to equator and Y axis in 90 ° east meridian from center to equator. Fig 1.1 shows ECEF frame.

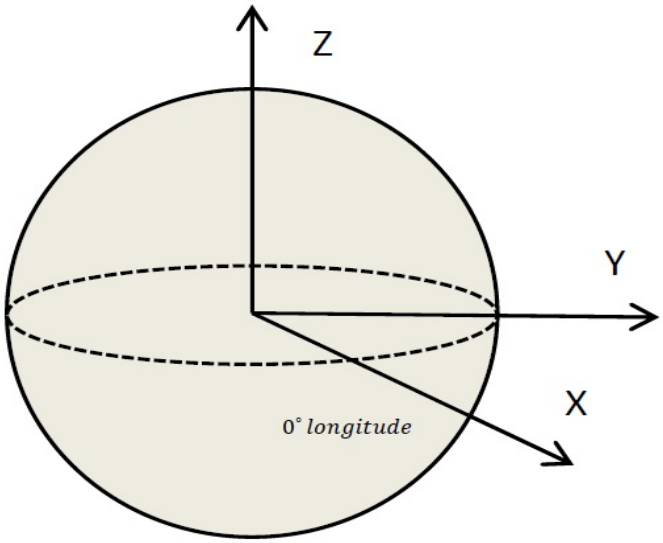


Fig. 1.1: Earth-Centered Earth-Fixed Frame

Local Navigation Frame or Geodetic is indicated by n and its origin could locate on mobile robot as it is shown in Fig 1.2. Z axis of geodetic coordinator is normal to the surface of ellipsoid earth from mobile robot to the center of earth and it corresponds to the gravity vector. X axis is projection of a line direction to the north on a plane normal to the z axis and Y axis is in east direction. Geodetic frame also known as NED frame that denotes direction of x, y, z axes in North, East and Down.

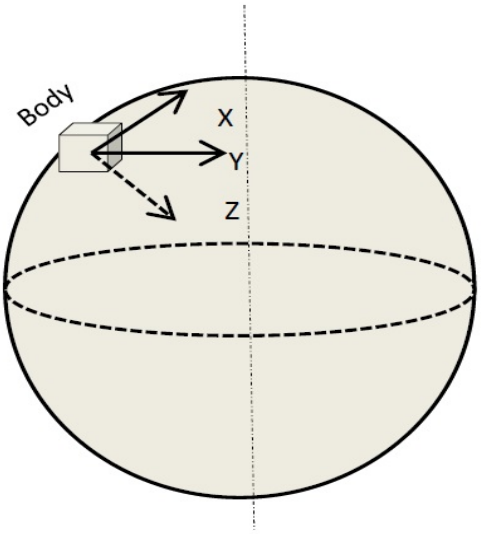


Fig. 1.2: Local Navigation Frame

Body Frame is indicated by b and its origin is located on the body (mobile robot) like NED frame. Z axis pointing down and x axis pointing forward and y must complete the right hand rule of coordinate frames. Another representation of body frame is roll, pitch, yaw axes which correspond to rotation around x, y and z axes respectively.

1.2 Attitude Representation

There are different methods to represent a vector in the coordinate frame. Sometimes to simplify an equation, it is helpful to change the representation format. In our case during developing Matlab Simulink or calculating formula different representation formats are used, thus here is brief explanation about changing between coordinate systems before delivering the main formula. For more detail refer to [6]:

1.2.1 Rotation Matrix Rate

Rotation matrix rotates a vector and represent it in a new coordinate frame by preserving vector length. A rigid body has a 3 by 3 proper rotation matrix with $det(R) = 1$ and $R^{-1} = R^T$.

To transform a vector expressed in body frame to world frame :

$$Z^w = R_b^w \cdot Z^b \tag{1.1}$$

Note that in following chapters R_α^β express the rotation matrix from frame α to frame β in equations, both of α and β can be replaced with desire frame:

To transform a point, we must consider the offset between two coordinate system origins :

$$X^w = R_b^w \cdot (X^b - X_w^b) \tag{1.2}$$

The origin of coordinate system always satisfy $X_b^w = -R_b^w \cdot X_w^b$
In above equations X^w is an arbitrary point in world frame and X_b^w is origin of body frame expressed in world frame, also X_w^b is origin of world frame expressed in body frame. To transform a point we can also use transformation matrix directly.

In case the orientation of two coordinate frames b as body frame and w as world frame are changing with respect to one another, then rotation matrix rate exists and its relation with angular velocity is:

$$\dot{R}_b^w \times A = R_b^w(\omega_w^b \times A) \tag{1.3}$$

A is arbitrary vector.

Because of cross product equation $p \times q = \Omega(p)q$, in which $\Omega(p)$ is skew symmetric matrix of p and is equal to:

$$R_w^b = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} \tag{1.4}$$

we rewrite the equation to:

$$\dot{R}_b^w = R_b^w \cdot \Omega_{wb}^b \tag{1.5}$$

Ω_{wb}^b is skew symmetric matrix of angular velocity ω_{wb}^b.

Direction cosine matrix (DCM) is a matrix that indicates the cosine of unsigned angles between two coordinate frames which is equivalent with rotation matrix. If the world axes coordinate system is denoted by (x, y, z) and body axes coordinate system is denoted by $\acute{x}, \acute{y}, \acute{z}$, then cosine matrix would be :

$$R_w^b = \begin{bmatrix} cos(\theta_{\acute{x},x}) & cos(\theta_{\acute{x},y}) & cos(\theta_{\acute{x},z}) \\ cos(\theta_{\acute{y},x}) & cos(\theta_{\acute{y},y}) & cos(\theta_{\acute{y},z}) \\ cos(\theta_{\acute{z},x}) & cos(\theta_{\acute{z},y}) & cos(\theta_{\acute{z},z}) \end{bmatrix} \tag{1.6}$$

1.2.2 Euler Angle Rate

Another form of rotation representation is using Euler angels. Three coordinate rotation $u = (\phi, \theta, \psi)$ in sequence with no two consecutive rotation about same axis can express any rotation.
Rotation matrix from world frame to body frame (R_w^b)is calculated by multiplying elementary rotation matrices along single axes in correct order:

$$R_{ijk}(\phi, \theta, \psi) = R_i(\phi) \cdot R_j(\theta) \cdot R_k(\psi) \tag{1.7}$$

ψ is first rotation about z axis of world frame and θ and ϕ are the second and third rotation about y and x axes respectively.

If $\hat{e}_i, \hat{e}_j, \hat{e}_k$ are unit vectors, then matrix E can be calculated:

$$E_{ijk}(\phi, \theta, \psi) = [\hat{e}_i, R_i(\phi)\hat{e}_j, R_i(\phi)R_j(\theta)\hat{e}_k] \tag{1.8}$$

The matrix that correlates Euler angle rate to angular velocity in body frame is:

$$\omega_{nb}^b = E_{ijk}(u) \cdot \dot{u} \tag{1.9}$$

and base on that :

$$\dot{u} = E_{ijk}^{-1} \cdot \omega_{nb}^b \tag{1.10}$$

There are 12 possible sequence of Euler angle that can lead to rotation matrix. Between 12 possible sequence, (1, 2, 3) rotation sequence is more common and is indicated by (ϕ, θ, ψ) as (roll, pitch, yaw) angles that can correspond to the body frame coordinator. This order of sequence is used in calculations in this thesis. The rotation matrix that yield from Euler angle is equal to:

$$R_{123}(\phi, \theta, \psi) = R_1(\phi) \cdot R_2(\theta) \cdot R_3(\psi) = \tag{1.11}$$
$$\begin{bmatrix} cos(\theta)cos(psi) & cos(\theta)sin(\psi) & -sin(\theta) \\ sin(\phi)sin(\theta)cos(\psi) - cos(\phi)sin(\psi) & sin(\phi)sin(\theta)sin(\psi) + cos(\phi)cos(\psi) & cos(\theta)sin(\phi) \\ cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi) & cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi) & cos(\theta)cos(\phi) \end{bmatrix}$$

To calculate Euler angles from rotation matrix, we can write Euler angles as function of rotation matrix:

$$u_{123}(R) = \begin{bmatrix} \phi_{123}(R) \\ \theta_{123}(R) \\ \psi_{123}(R) \end{bmatrix} = \begin{bmatrix} atan2(r_{23}, r_{33}) \\ -asin(r_{13}) \\ atan2(r_{12}, r_{11}) \end{bmatrix} \tag{1.12}$$

Inverse of Euler angle rate matrix E in body frame with (1, 2, 3) sequence is:

$$E_{123}^{-1}(\phi, \theta, \psi) = (1/cos(\theta)) \begin{bmatrix} cos(\theta) & sin(\phi)sin(\theta) & cos(\phi)sin(\theta) \\ 0 & cos(\phi)cos(\theta) & -cos(\theta)sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{1.13}$$

1.2.3 Quaternion

Quaternion $q \in \mathbf{H}$ is a popular method to represent the attitude of mobile robots and it is denoted with vector q as combination of one real and three imaginary parts,

the real part shows the magnitude of rotation and the imaginary vector shows the axis around which rotation take places:

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{1.14}$$

or

$$q = q_0 + q_1 i + q_2 j + q_3 k$$

quaternion has definition for different mathematical operation and here I mentioned some of them: quaternion conjugate (transpose of complex conjugate) is :

$$\bar{q} = [q_0, -q_1, -q_2, -q_3] \tag{1.15}$$

If we indicate quaternion q with $[q_0, \mathsf{q}]^T$ and quaternion p with $[p_0, \mathsf{p}]^T$,then multiplication between quaternions q and p is equal to :

$$q \cdot p = \begin{bmatrix} q_0 p_0 - \mathsf{q}^T \mathsf{p} \\ q_0 \mathsf{p} + p_0 \mathsf{q} - \mathsf{q} \times \mathsf{p} \end{bmatrix} \tag{1.16}$$

Norm and inverse of quaternion are defined as:

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{1.17}$$

$$q^{-1} = \bar{q}/|q| \tag{1.18}$$

Mapping between quaternion rate and angular velocity is :

$$\dot{q} = (1/2) \cdot W(q) \cdot \omega_{nb}^b \tag{1.19}$$

In which W(q) is called quaternion rate matrix that maps $H \longrightarrow R_{3\times 4}$

$$W(q) = \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \tag{1.20}$$

1.3 Velocity and Changing Coordinate Frame

If $\omega_{\beta\alpha}^{\gamma}$ would be angular rate rotation of object frame α with respect to frame β (reference frame) and represented in frame γ, then the representation frame can be changed by multiplying to suitable coordinate transformation matrix. Simply angular rate vectors with same resolving frame can be added easily:

$$\omega_{\beta\delta}^{\gamma} = \omega_{\beta\alpha}^{\gamma} + \omega_{\alpha\delta}^{\gamma} \tag{1.21}$$

Angular rate rotation vector reverse by sign if the object frame and reference frame replace with each other in a same resolving frame (representation frame γ):

$$\omega^{\gamma}_{\alpha\beta} = -\omega^{\gamma}_{\beta\alpha} \tag{1.22}$$

Velocity is produced as result of moving the origins of reference β and object frame α or as result of rotating the axes of reference frame with respect to object frame (notice that no velocity is produced by moving or rotating the axes of reference frame). If resolving frame γ and reference frame β has angular motion with respect to each other, then velocity $V^{\gamma}_{\beta\alpha}$ is not the only term in correlation with time derivative of position $\dot{r}^{\gamma}_{\beta\alpha}$:

$$\dot{r}^{\gamma}_{\beta\alpha} = \dot{R}^{\gamma}_{\beta} \cdot r^{\beta}_{\beta\alpha} + R^{\gamma}_{\beta} \cdot \dot{r}^{\beta}_{\beta\alpha} = \dot{R}^{\gamma}_{\beta} \cdot r^{\beta}_{\beta\alpha} + V^{\gamma}_{\beta\alpha} \tag{1.23}$$

In case there is no angular motion between reference frame and object frame, interchanging the reference and object frame is possible by reversing the sign of velocity vector, otherwise equation extends to:

$$V^{\gamma}_{\beta\alpha} = -V^{\gamma}_{\beta\alpha} - R^{\gamma}_{\alpha}\dot{R}^{\alpha}_{\beta}r^{\beta}_{\beta\alpha} \tag{1.24}$$

Moreover if $\dot{R}^{\beta}_{\alpha} = 0$, then:

$$V^{\gamma}_{\alpha\delta} = V^{\gamma}_{\alpha\beta} + V^{\gamma}_{\beta\delta} \tag{1.25}$$

1.4 Relevant CANopen Introduction

CAN is a serial bus communication with multi master capabilities based on message priority and message identifier, moreover CANopen came to existence by developing standardized application and functionality base on CAN.

1.4.1 Object Dictionary

Object Dictionary (OD) is a set of accessible data each with specific ID and name which are ordered within a table. Each entry of object dictionary differentiated by an index and subindex. Index consists of 16 bits that assigns an ID to entry and it can contain up to 256 sub entries each 8 bits in size.

Mainly OD categorized into data type entries, communication entries, manufacture specific and device profile parameter. Therefore each sensor could have its own object dictionary corresponds to $DS-301$ standard. Each entry has data type from Standard data type list or from complex data type list. Standard data type includes Boolean and Integer. Complex data includes predefined or manufacturer data type within index 0001 to 0FFF. Moreover each entry has access type that indicates to write in an entry or read from an entry. RW, WO, RO abbreviations stands for Read and Write, Write Only, Read Only .
Index range 1000 to 1FFF is assigned to communication entries that contains different services of CANopen communication.
Index range 2000 to 5FFF is assigned to manufacturing specific entries that allow manufacturer of specific node to store data.

1.4.2 Data Transmission

CANopen provide two type of data transmission. Service Data Object provides a service to access the data in object dictionary; therefore a node as master or client can read or write data to the object dictionary of another node as server. COB ID of SDO messages is 0x600+node ID from master to slave and 0x580+node ID from slave to master. Table 1.1 indicates the data frame of SDO message type.

byte 0	byte 1	byte 2	byte 3
Specifier	OD Index lower byte	OD Index higher byte	Subindex

Tab. 1.1: SDO frame

Here is most common use of SDO command specifier for transferring maximum 4 bytes of data (in hexadecimal):
Download Request: 22
Download Response: 60
Upload Request: 40
Upload Response: 42

The second type message for real-time transmission data is Process Data Object (PDO) that provides a communication level for transmitting data up to 8 bytes in

every message that can be triggered by several means. This message is divided into two type of Receive Process Data Object (RPDO) and Transmit Process Data Object (TPDO), and they indicate to generate a message by a node (TPDO) or consume the same message by another node (RPDO). TPDO and RPDO have to be configured by communication parameter that shows on table 1.2.

Subindex	Name
0	Number of entries
1	COB ID
2	Transmission type
3	Inhibit Time
4	Reserved
5	Event Time

Tab. 1.2: communication parameters TPDO or RPDO

Each index in TPDO communication parameters correlate to an index in TPDO mapping parameters, for instance TPDO1 communication parameters are at index 1800 and its mapping parameters are at index 1A00.

One means to trigger a TPDO is Event driven, which transmits the message as soon as a change in data occurs. Data can be anything from device profile and it is also possible to define an inhibit time to determine a minimum interval to transmit a message with next event after inhibit time is expired. Another means to trigger a message is Time driven ,which transmits a message based on a constant period of time. This time is a local timer on each CANopen nodes.

A common methods in robotic and drive system is Synchronized polling to trigger a message and it assures to receive all messages from different nodes at the same time. In this method a node in the system is producer of a signal that works as trigger for the other nodes and it is called SYNC signal. SYNC signal ID is configurable in object dictionary 1005 in each nodes. Also there could be several SYNC signal in a system that each of them could synchronize a specific group of nodes that has same ID in their OD.(By default the SYNC message COB ID is set to $80h$ based on hexadecimal).

It is worth mentioning a predefined range of COB ID is available for each CANopen node base on table 1.3.

Network management (NMT) implements transitions between different operating states of a slave node by sending series of commands by Mater node. There are several states for a node, some states transition by specific NMT commands and some are automatically. Basically there are 3 main states of Preoperational, operational, stopped. It is worth mentioning that there is merely in operational state that SDP, PDO, heartbeat, SYNC services are available and in the rest of states only few of them exist.

NMT master message starts with identifier (COB ID) 0 and consists of 2 bytes data, the first byte is command specified and second byte is node ID that can be filled with zero to address all slave nodes in the system.

OD Identifier	PDO type and number
181 to 1FF	First TPDO
201 to 27F	First RPDO
281 to 2FF	Second TPDO
301 to 37F	Second RPDO
381 to 3FF	Third TPDO
401 to 47F	Third RPDO
481 to 4FF	Fourth TPDO
501 to 5FF	Fourth RPDO

Tab. 1.3: COB ID for TPDO and RPDO

operational=1
Stopped=2
Preoperational=128
Reset nodes=129
Reset Communication=130

In order to raise the communication and safety of system with multiple nodes, every node must demonstrate its current state within 1 byte of message that indicates they are in operational or fault state. This protocol is called Heartbeat and its COB ID is 700.

2. IMORO HARDWARE ARCHITECTURE

2.1 Locomotion Mechanism

Design and control of a mobile robot with wheeled locomotion mechanism changes base on the number of degrees of freedoms. Different types of wheel configuration provide different level of strength in terms of traction, stability and maneuverability of robot. (See H. Rajaei [7] or R. Grepl [8])

Minimum of two wheels configuration can achieve static stability, but it provides limited maneuverability. Instead three or four wheel mechanism with two independent driven wheels improve maneuverability (For more detail refer to [9]). A step further are omni directional mobile robots that can move in any direction at any time, this means that they need more complicated control model because of rises in degree of freedom.(See R. Siegwart [10]) iMoro robot with four wheeled legs is considered as a semi-omnidirectional which makes it able to follow a given desired path.(For more detail refer to [11])

Therefore design of a mobile robot is a complicated process and depend on its application, it can be equipped with number of high or low precision sensors and actuators. iMoro robot is equipped with 8 Electronically Commutated motors (EC motors) and digital positioning controllers (known as EPOS) described in section 2.1.1 and section 2.3 described how to synchronize these 8 motors via controller driver and communication protocol.

For linking of intelligent electronic components within a system a bus or network needed to be established as communication protocol. Communication protocols in automation system reduce the number of controllers and physical size and cost, moreover it is a foundation for centralized control unit. Section 2.2.1 briefly described available communication protocols in the market and the advantages of the CANopen protocol as selected communication protocol for iMoro. In following installing the CAN bus between all nodes in iMoro described in section 2.2.2.

Section 2.5 explains software interface for IMU and absolute encoders in real-time operating system. For more detail a brief description about CANopen protocols used in iMoro is provided in section 1.4. Locomotion mechanism of a mobile robot determines what type of actuators can be employed. Among variety of linear and rotary actuators, DC or AC motors, Brush less DC motors are offered in smaller size and higher angular velocity that make them suitable for robotic applications. See section2.1.1

Because of high precision in mobile manipulator application brushless servo or stepper motors are the two most common types. They provide higher precision and

availability with different angular velocity and torque. Moreover the recent version of these motors provide an angular velocity feedback. For a drive system like iMoro an actuator is chosen base on require angular velocity and torque and precision of control. Moreover gearboxes after steering motors and after driving motors change the proportion of angular velocity to torque.

Nowadays EC motors are offered along with a different range of driver units. Those units work as controller for motors and they can send command and receive a feedback from motors. See section2.1.3

2.1.1 Electronically Commutated Motors

EC motors are brushless DC motors that consist of electronic communication block instead of mechanical communication that can invert DC electric power to AC electric signals with specific frequency and amplitude. Utilizing electronic communication increases the motor life and the maximum motor speed. Hall sensor detects the position of rotor (control magnet) and produces signals that are used as input for electronic communication block.

Fig. 2.1: Electrically Commutated motor

Two EC motors assigned for every wheel, one for steering and another for driving. Driving shaft passes through the steering gearbox with ratio (1:3) and it connects to wheel by a belt with ration (1:8) and steering shaft gearbox (1:60) and it connects to the wheel by holding the wheel axis from two sides.

2.1.2 EPOS Position Controller

EPOS positioning controller is a digital positioning system suitable for DC and EC motors with incremental encoders in a modular package. EPOS controller has several input connectors from motors side like hall sensor, incremental encoder, signals that is used for brake and it has two CAN bus connectors for communicating with other devices.

Each of controller devices has 8 dip switches are designed to set NOD ID by first 7 switches and last switch represents resistor when controller is located as first or last node in CAN bus it can.

Fig. 2.2: EPOS position controller

As it is demonstrated in Fig 2.3 controllers were located in racks under the platform on both side and they are wired to the motors through a hole in middle of platform. Each of controllers are connected to each other to make CAN bus line. See section 2.2.2.

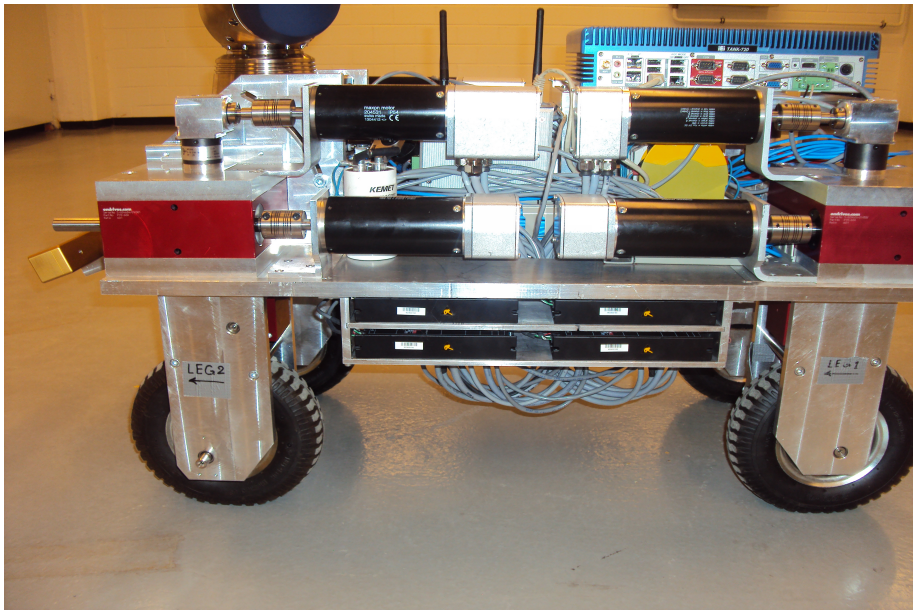


Fig. 2.3: EPOS controllers rack under the iMoro platform

2.1.3 Driving and Steering Motors

Among different types of modes available on each of the controller, position mode is used to operate on steering motors in which position is the input value (it must be set in OD 0x2062 known as position mode setting value), moreover maximum velocity and acceleration and position limit are terms that have to be configured.

On the other hand velocity mode is chosen as desire operation mode for driving motors. Therefore velocity value that is called velocity mode setting value is the input and two terms of maximum velocity and acceleration has to be defined as the configuration parameters.

In object dictionary, variable units are defined and they are configurable. By default device internal unit for position is "steps" or "quad counts" and rev/min and rpm/sec are assigned to velocity and acceleration respectively.

2.2 Communication Protocols

2.2.1 Standard Communication Protocols

Nowadays many communication protocols are available in the market like Ethercat, CAN, device net, RS232 and most protocols work based on the serial transmission. To choose a protocol several criteria must be considered based on your application for instance maximum distance between component, network layout, product availability.

Ethernet based protocol like EtherCAT has been brought fast and low cost hardware communication to automation applications and Profibus provides a communication protocol with capability of transmitting higher size of data frame in long distance and also variety of device profiles for different applications are available.

Controller Area Network first was designed for automobile industries and because of its high efficiency it was developed further to applications like cranes, railway vehicle, robotic. Up to now CAN has been developing day by day and low cost CAN protocol made manufacturer of different sensor and actuator brands to use this protocol in their products. Therefore by covering the malfunction of last produced protocols, CAN became suitable option for mobile machines applications. Another advantage of CAN for mobile robot integrated with multiple electronic components was providing a secure communication protocol that guarantees a safe transmission of data.

Communication interface of all nodes in iMoro follows application layer and communication profile the DS-301 of CiA CANopen standard. DS-301 standard is the foundation of CANopen and it mainly includes basic communication and data structuring and network management method (like SDO and NMT).

Depend on the application of CANopen device, additional standard can be added to the protocol that has already been defined by CiA. Basically, different standards in CANopen device are located in different level of protocol, mostly application profiles and device profiles (DS4xx)are in higher level and communication

profiles and frameworks (DSP3xx) are in lower level of communication.(For more detail refer to [12])

For instance the states of drive follows the DSP-402 known as device profile drives and motion control, base on that several internal behaviors of device like stopping and starting and variety of operation modes can be controlled by a sequence of commands. See section 2.4

2.2.2 Nodes Number In iMoro

As mentioned earlier in 1.4 to communicate with every node in the system, specific NOD ID must be assigned to them. Table 2.1 indicates specified NOD ID.

Device		Node ID
Drivring:		
	Rear	11
		21
	Front	31
		41
Steering:		
	Rear	12
		22
	Front	32
		42
Embedded PC		70
Encoders:		
	left	51
	right	52

Tab. 2.1: Node ID of motors and sensors and embedded PC in CAN bus

CAN low and CAN high are the two signals that must pass through all nodes within a network through wiring to establish CAN bus communication, thus wiring connection for a CAN bus implemented easily. Moreover, both end of the CAN low and CAN high must be terminated with 120 Ω resistor.

Every controller driver (motor driver) has two line of CAN bus with Molex Micro 4 pole connector that allows us to extend CAN bus to other nodes and embedded pc which is equipped with two M12 connectors. Because of the high number of nodes and consequently high number of CAN frame in the system, it is possible to lose data in maximum baud rate 1Mbps. That is why we considered to set two CAN bus to increase the reliability of CAN bus to avoid losing data. The first line includes motor drivers, embedded PC, absolute encoders and it is called CAN bus 1. The second line connects IMU to embedded PC directly and it is called CAN bus 2.

As it is demonstrated in Fig 2.4, CAN bus 1 starts from absolute encoder to node 11 and it passes through nodes number 12, 21, 22 to embedded PC and

it follows by motor driver on the other side of platform which are nodes number 31, 32, 41, 42 and it ends up to absolute encoder again. Finally CAN bus line is terminated with 120 Ω resistor at both ends.

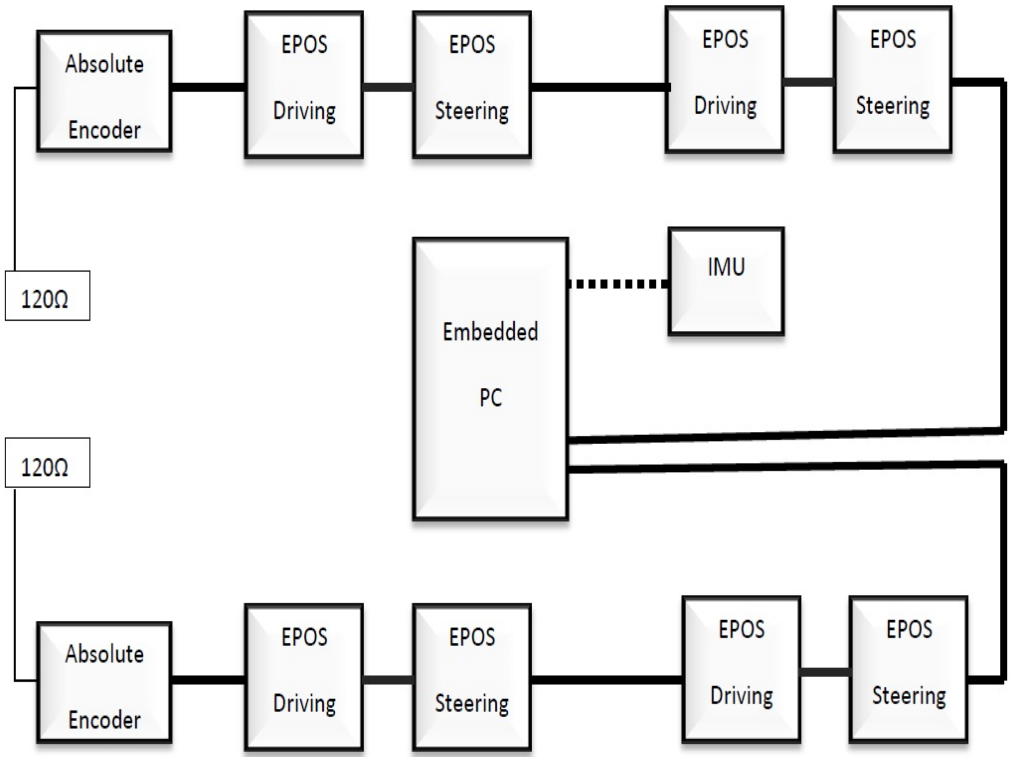


Fig. 2.4: CAN bus

2.3 Controller Set Point and Feedback

As steering motors are working in position mode, a PDO that consists of 2 bytes control word and 4 byte position send as set point to motor by master node. Same method is implemented for driving motors only by replacing velocity instead of position.

A SYNC signal sends by master node after sending set points in every 5 ms. Therefore each motor driver has one RPDO with synchronization signal as its transmission type (it processes data after receiving SYNC signal). On the other hand each motor driver sends two TPDO, the first TPDO consists of 2 bytes status word plus 4 bytes position actual value and second TPDO consists of 4 bytes velocity actual plus 2 bytes current value. TPDO are sent upon receiving the SYNC signal in every 5ms.This routine is implemented on a real-time task with 5ms period that causes to received PDO from all motor drivers at the same time as feedback.

As mentioned earlier in 1.4 to improve the reliability of system heartbeat protocol or node guarding can be added to system communication. To make sure about state of every node in each second, every motor driver and absolute encoder send heartbeat within 100ms and embedded PC sends heartbeat every 50ms.

CAN bus 2 is more simple than CAN bus 1, because IMU does not have any state except operational, therefore as soon as the device is turned on, it starts to send data every 7.8 ms and none of the above protocols like SYNC or Heart beat needs to be implemented for IMU. IMU sends two packages of CAN frame every 7.8ms, the first CAN frame consists of 6 bytes acceleration data and 2 byte diagnostic data. The second CAN frame consist of 6 bytes gyroscope data and 2 bytes temperature. Each of these 6 bytes of data in acceleration or gyroscope are divided to 2 bytes for each direction .

2.4 Control Word and Status Word

According to Fig 2.5 the state of drive can change between three main state of Power disable and Power Enable and Fault,Power Enable state is when the system is operational and ready to receive higher level of control command to drive the motor.
Power Disable is happened whenever the system is turned on and initialized, moreover whenever a fault occurred in system, state changes into Fault state and stops there till fault reset command will be sent to server.

These states change by sending 1byte commands message that is called Control Word in object dictionary 0x6040. Every state follow a specific number that is assigned to bits zero to 3 of control word and rest of bits are reserved or related to operation modes-specific.here are some important ones:
Shutdown: 6
Switch On when power disable: 7
Switch On when power enabled: 15
Disable Voltage: 0
Quick Stop: 2
Fault Reset: 128

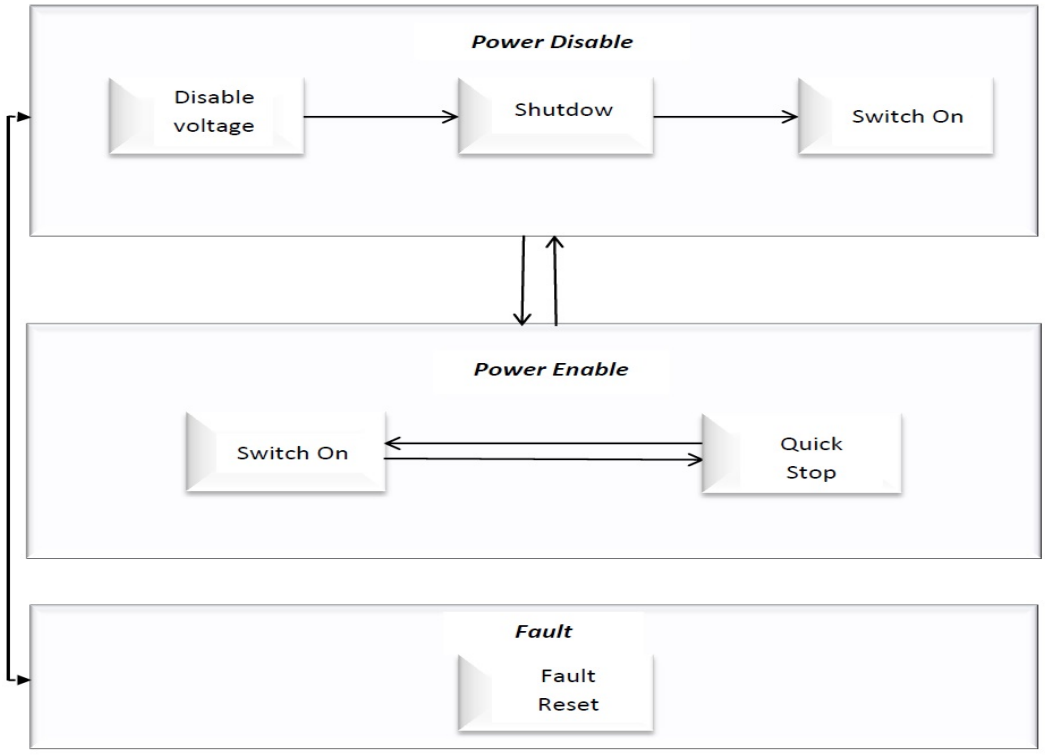


Fig. 2.5: Device Control (State of Drive)

States of drive can be read from object dictionary 0x6041 that is called Status Word and based on each state, a control word is expected to send to the drive for transition. Status word is 2 bytes message in which a specific number is assigned to bits zero to 6 and 8 and 14, the rest of bits are meant for operation mode.

2.5 Real-time Software Interface

Nowadays real-time systems are applied in control industries, chemical process industries, simulation systems, etc. Basically real-time operating system guarantee the action in response to events within specified amount of time.

iMoro is equipped with quad-core embedded PC as controller and has Xenomai real-time linux as the Operating System (OS). Xenomai is real-time OS that can employ time constraints for executing each task. Real-time programming contains a task that waits for an event or time expiration to be triggered. After that task starts to execute its function and then it stops again till next period.

Sometimes different independent tasks needs to communicate to achieve synchronization in real-time operating systems. There are several mechanisms that are offered services like Queue, Semaphore, mailbox. Queue is the service implemented on iMoro between IMU sensor or controller drives and embedded PC for sending commands and receiving data.

Queue provides a mechanism when tasks need to share data. One task sends data to queue and another task wait to receive data from queue. A queue can hold

multiple messages base on the size of queue when it is created and it can order messages based on higher priority.(For more detail refer to [13])

Since CANopen is a relevant network under real-time applications, the software interface for sensors with CANopen protocol was implemented using Xenomai RTDM library. Real-Time Driver Module (RTDM) is associated for developing device drivers and it is implemented on Xenomai, offers a service between applications requesting a service from a certain device and device driver.

RTDM supports message based devices like CAN by choosing protocol family and socket type and also it provides support for messages transmission. CAN protocol family (PF-CAN) has been implemented in socket layer for communication in CAN bus.

In addition Low level CAN Framework (LLCF) provides transport protocols like Broadcast-Manager or RAW socket, In brief the former is a socket interface that allows to send a message in CAN bus periodically and the latter is a socket interface that allows direct communication for a single message. In general CAN protocol has similarity to Internet protocol sockets with system calls like bind(), socket(), read(), send(). However there are some differences in address structure and data frame that are defined in include files pcan.h and af-can.h.

Communication between different parts of software in iMoro robot is implemented in real-time environment as much as possible, however wireless connection doesn't have properties of a real-time communication. Thus a UDP is established for a fast wireless connection.

UDP, User Datagram Protocol provide a fast, but unreliable packet delivery system. The size data in every packet can't exceed more than a single IP packet (65077 bytes) and the destination address uniquely will be identified by IP and port number of recipient. Delivery of UDP packages are out of order and there is no acknowledge part, that is a reason for fast and unreliable communication.

3. IMU AND CAMERA SIMULATION AND CALIBRATION

3.1 Navigation Technique

Navigation technique is a method for determining position and velocity of vehicles. The output of navigation technique is known as navigation solution. Most of Navigation technique are based on dead reckoning and position fixing method. For more detail refer to [5] A dead reckoning system integrated with position fixing provides a better navigation solution. In fact the position fixing system can be used by an estimation algorithm (like Kalman Filtering) to apply correction to the dead reckoning navigation system for example see S. Won [14].

Modern navigation systems like Global Navigation Satellite System (GNSS) utilize the advantage of satellite for navigation that provides the position of an object on or near to the earth surface. The accuracy of position navigation depends on receiver on earth and climate situation. While GPS has many advantages for different applications today, it is not helpful for indoor applications like indoor mobile robot. Therefore Inertial Navigation System (INS) still is key element for navigation applications for instance see T. Lee [15].

Current inertial sensors development are based on MEMS technology that improves the performance and reduces the size and sensor costs, however the process of IMU calibration by itself is essential step to receive precise and meaningful data from raw IMU data. Different methods can be used for calibration, for instance multi-positioning method that collects data by locating the IMU in different orientation and used for instance by Q. Cia [16] and also by A. Amirsadri [17], moreover auxiliary devices are introduced during calibration process such as optical tracking device see for example A. kim [18]. The process of receiving meaningful data out of raw data for iMoro IMU explained in this chapter.

Moreover, due to characteristic of IMU and camera (or laser range scanner) to present the motion of a vehicle, data fusion between these two sensors has became a popular research topic see for instance relevant recent works by F. M. Mirzaei [19] or James Underwood [20].

In this chapter in section 3.2 IMU characteristic is explained briefly. Moreover method to calibrate and reduce error is explained in sections 3.2.4 and 3.2.5. Finally Calibration method between IMU and camera by driving the mobile manipulator is described in section 3.3.

3.2 Inertial Measurement Unit

Inertial Navigation System provides three dimensional navigation solutions obtained from inertial sensor measurement. Mostly accelerometer and gyroscope are sensors that used in INS to provide the position of an object. The computation of inertial navigation solution would be an iterative process in navigation processor. Velocity and position of body is updated by integrating of acceleration and attitude of body is updated by integrating angular velocity.(See for instance J. Yi [21])

Inertial Measurement Unit is composed of three accelerometers and three gyroscops which are mounted in three directions orthogonally to produce acceleration and angular velocity. The produced acceleration a_{ib}^{b} and angular velocity w_{ib}^{b} are expressed with respect to inertial space in body frame.

3.2.1 IMU Installation

ADIS 16385 shows in Fig 3.1 is the IMU used for iMoro with tri-axis accelerometer and tri-axis gyroscope which support serial peripheral interface (SPI) with approximate dimension $36mm \times 47mm \times 39mm$. Table 3.1 shows ADIS16385 specification briefly. (For more detail refer to [22])

Fig. 3.1: Inertial Measurement Unit ADIS16385

The IMU is located in a metal box and it is fitted on front of iMoro between the gear box as it is shown in Fig 3.2. Therefore the IMU z axis (yaw) coordinate frame corresponds to gravity. The position of the body coordinate frame origin is

21.4, 8.1 centimeter.Two electric plugs located on the edge of box, one is feeded 24 voltage for the IMU and another is connected to CAN bus 2. See section 2.2.2

IMU box dimension	$36mm \times 47mm \times 39mm$
Body frame position:	x=21.4 cm y=8.1 cm
Supply Voltage CAN connection	24 v CAN bus 2

Tab. 3.1: IMU Installation Specification

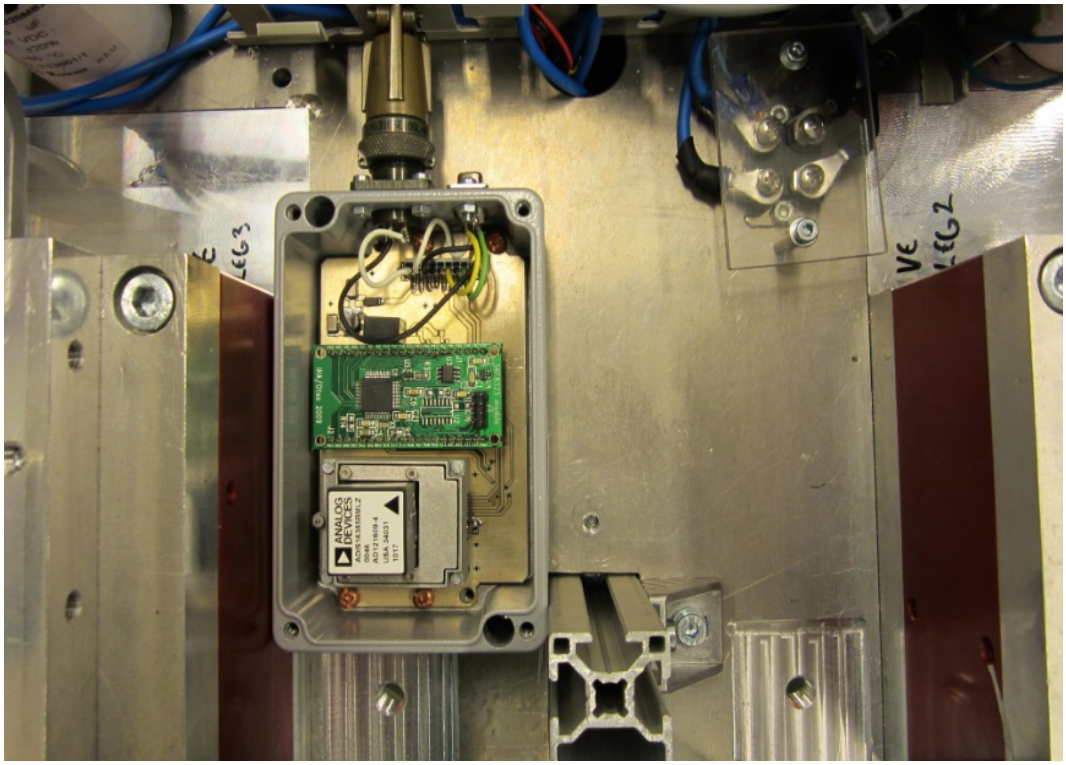


Fig. 3.2: Location of IMU in front of platform between the gearboxes

3.2.2 IMU Acceleration

Specific force is the non gravitational force per unit mass on body, sensed with respect to an inertial frame. IMU measures the specific force of IMU body with respect to inertial frame that is expressed in body frame and is indicated by f_{ib}^{b} as follow:

$$f_{ib}^{b} = M_a \cdot a_{sens} - b_a \tag{3.1}$$

In which a_{sens} is the acceleration output from IMU and b_a is bias, also M_a is misalignment matrix and its components calculated in section 3.2.5.

A sample of raw acceleration data from IMU is shown in Fig 3.3, in this test IMU was at rest for 2 minutes with 7.8 millisecond sampling time and around 15000

data collected. If IMU has misalignment matrix equal to identity matrix and biases equal to zero, then in stationary condition the value of acceleration in 2 directions must be equal to zero and equal to gravity in third direction.

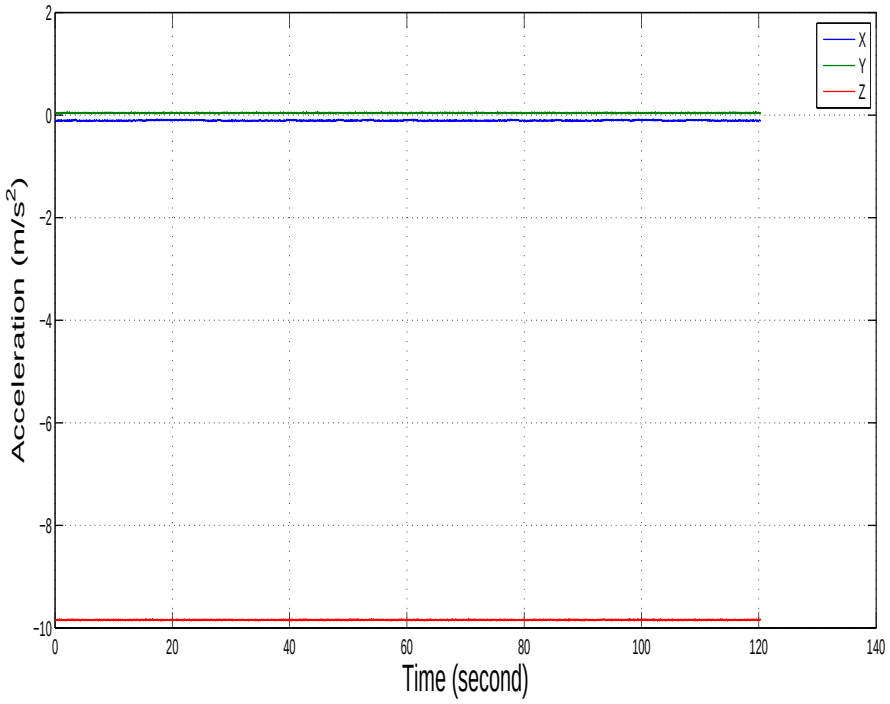


Fig. 3.3: Raw acceleration data

3.2.3 IMU Angular Velocity

Angular velocity as output of IMU consists of angular rate of body where IMU (w_{ib}^{b}) is attached and bias (b_g):

$$w_{sens} = w_{ib}^{b} + b_g \tag{3.2}$$

Raw angular velocity data is indicated in Fig 3.4, As IMU is in stationary position for 2 minutes, the value of angular velocity must be equal to zero. Noise in output data eliminated by averaging over 2 minutes.

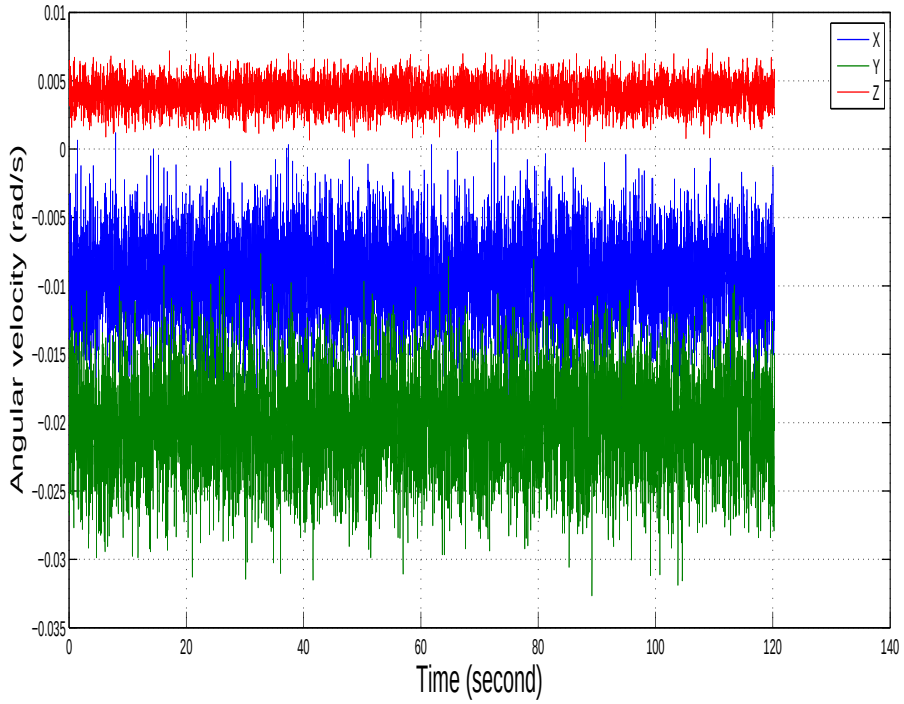


Fig. 3.4: Raw angular velocity data

3.2.4 IMU Errors

IMU errors are categorized into deterministic and nondeterministic or stochastic errors. There is direct relation between input (acceleration or angular velocity) and output(signal) in deterministic error but stochastic errors have random nature and they change within time, so that the models are determined with different algorithm and experiments in laboratory.(For more detail refer to [18] and [5])

Two main part of errors are about biases and scale factors. To attain better understanding of scale factors first we should consider the fact that accelerometers and gyroscopes must be mounted orthogonally to produce proper output data. But from mechanical aspect this alignment never can be 90 ° exactly, therefore misalignments exist in IMU and they are called orthogonal alignment error and package alignment error.

Misalignment matrix is 3 by 3 matrix with nine components that express non orthogonality of each axis in compare with other two axes and diagonal elements are called scale factor (scale factor consist of a fixed and temperature variation part):

$$\begin{bmatrix} S_x & M_{xx} & M_{xz} \\ M_{yx} & S_y & M_{yz} \\ M_{zx} & M_{zy} & S_z \end{bmatrix} \tag{3.3}$$

Each IMU has a unique misalignment matrix for accelerometers and gyroscopes.

Another major part of IMU errors is biases that is divided to fixed term, temperature induced, turn on variation and in run (or instability) variation. The first term is classified as deterministic errors.(see E. Nebot [23])

Stochastic errors in my model included random variation of bias, scale factor, random sensor noise and white noise:

$$b = b_0 + b_1 + b_2 + w_3 \tag{3.4}$$

b_0 is fixed term of bias
b_1 is in run variation or bias instability
b_2 is rate random walk
w_3 is zero mean white noise.

All of these terms must be defined for every accelerometer and gyroscope in x,y,z direction. Thus we provide a linear model like:

$$\begin{cases} \dot{X}_g = A_g \cdot X_g + B_g \cdot \eta_{g_1} \\ b_g = C_g \cdot X_g + D_g \cdot \eta_{g_2} \end{cases} \tag{3.5}$$

All coefficients for gyroscope are introduced here one by one. First X_g is a 6 by 1 vector:

$$X_g = \begin{bmatrix} b_{x_{g_1}} & b_{x_{g_2}} & b_{y_{g_1}} & b_{y_{g_2}} & b_{z_{g_1}} & b_{z_{g_2}} \end{bmatrix}^T \tag{3.6}$$

And b_g is equivalent with b in equation (3.4).
η_{g_1} is 6 by 1 matrix of white noise:

$$\eta_{g_1} = \begin{bmatrix} \omega_{x_{g_1}} & \omega_{x_{g_2}} & \omega_{y_{g_1}} & \omega_{y_{g_2}} & \omega_{z_{g_1}} & \omega_{z_{g_2}} \end{bmatrix}^T \tag{3.7}$$

A_g matrix is 6 by 6 matrix that contains correlation time:

$$A_g = \begin{bmatrix} 1/T_{x_g} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/T_{y_g} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/T_{z_g} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.8}$$

B_g is a 6 by 6 matrix that defines bias instability and rate random noise:

$$B_g = \begin{bmatrix} B_{x_g} & 0 & 0 & 0 & 0 & 0 \\ 0 & K_{x_g} & 0 & 0 & 0 & 0 \\ 0 & 0 & B_{y_g} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{y_g} & 0 & 0 \\ 0 & 0 & 0 & 0 & B_{z_g} & 0 \\ 0 & 0 & 0 & 0 & 0 & K_{z_g} \end{bmatrix} \tag{3.9}$$

C_g is a 3 by 6 matrix that defines as:

$$C_g = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.10}$$

D_g is 3 by 3 random noise matrix

$$D_g \begin{bmatrix} D_{x_g} & 0 & 0 \\ 0 & D_{y_g} & 0 \\ 0 & 0 & D_{z_g} \end{bmatrix} \tag{3.11}$$

η_{g_2} is 6 by 1 matrix of white noise:

$$\eta_{g_2} = \begin{bmatrix} \omega_{x_{g_3}} & \omega_{y_{g_3}} & \omega_{z_{g_3}} \end{bmatrix}^T \tag{3.12}$$

Same equation model like gyroscope defined for accelerometer to estimate biases:

$$\begin{cases} \dot{X}_a = A_a \cdot X_a + B_a \cdot \eta_a \\ b_a = C_a \cdot X_a + D_a \cdot \eta_a \end{cases} \tag{3.13}$$

In a general case deterministic part of gyroscope errors can be omitted by averaging over time as long as the IMU is at rest, that implies no angular velocity applied to the IMU and the output is bias. Moreover deterministic error of accelerometer calculated in a same way after reducing the gravity acceleration, often it is called zero-g bias.

3.2.5 Calculating Acceleration Scale factor

Scale factors are diagonal elements of misalignment matrix that directly effect on the value of acceleration.
In an ideal IMU scale factors are equal to one but in reality they have different value depends on the quality of IMU. Scale factor as deterministic error is calibrated by a test (For more detail refer to [17]). In principle the value of accelerometer at rest is fix and equal to $1g$ on the surface of ground, no matter of the orientation of IMU at rest, the outcome of acceleration should be equal to gravity of earth or $1g$.

$$\vec{a}_x^2 + \vec{a}_y^2 + \vec{a}_z^2 = \vec{g}^2 \tag{3.14}$$

In above equation a_x,a_y,a_z are output of accelerometers in x,y,z directions. By using equation (3.1) and replacing the M_a from equation (3.3), we have:

$$(S_x \cdot a_x - B_x)^2 + (S_y \cdot a_y - B_y)^2 + (S_z \cdot a_z - B_z)^2 = g^2 \tag{3.15}$$

To find the six unknowns in above equation we must find six equations, this has been achieved by placing the cubic housing of IMU on its six sides. As bias grows with time and can make a significant drift on output data, the whole test took $30s$ which makes to keep the IMU housing on each side for 5s at rest. Then acceleration data filtered and averaged values used as input for the equation, Finally the equation solve by *fmincon* function in matlab environment. As expected the answer are near to one:

$$S = \begin{bmatrix} S_x = .9986 \\ S_y = .9991 \\ S_z = .9984 \end{bmatrix} \tag{3.16}$$

3.2.6 Angular Velocity and Acceleration

Angular velocity of mobile manipulator with respect to NED frame (n) expressed in body frame is equal to:

$$\omega_{nb}^{b} = M_g(\omega_{sens} - R_n^b(\lambda) \cdot R_e^n(r_{eb}^e) \cdot \omega_{ie}^e) - b_g \tag{3.17}$$

M_g misalignment matrix of gyroscope is considered to be identity matrix.
ω_{sens} is the angular velocity output from IMU.
$R_n^b(\lambda)$ is rotation matrix from Geodetic to body frame which is a function of Euler angles, and $R_e^n(r_{eb}^e)$ is rotation matrix from ECI frame to Geodetic frame which is a function of distance between earth center to mobile robot.

As illustrated earlier, equations (1.10) and (1.13) link the angular velocity and Euler angle rate, as a result after computing angular velocity from equation (3.17) we can calculate Euler angle rate.

Earth rotates about common z axis of ECI and ECEF frame with $7.292e^{-5} rad/s$, therefore the earth rotation vector is:

$$\omega_{ie}^e = \omega_{ie}^i = \begin{bmatrix} 0 & 0 & \omega_{ie} \end{bmatrix}^T \tag{3.18}$$

A differential equation for velocity is available to estimate the value of velocity:

$$\dot{V}_{eb}^e = R_n^e(r_{eb}^e) \cdot R_b^n(\lambda) \cdot f_{ib}^b + R_n^e(r_{eb}^e) \cdot g_b^n(r_{eb}^e) - 2\Omega_{ie}^e \cdot V_{eb}^e \tag{3.19}$$

$g_b^n(r_{eb}^e)$ is the earth gravity which depends on the distance between mobile manipulator and earth center.
Ω_{ie}^e is the skew symmetric matrix of earth angular velocity.
Finally the position of mobile manipulator with respect to ECEF derived from velocity integral:

$$\dot{r}_{eb}^e = V_{eb}^e \tag{3.20}$$

Solving equations of (3.20), (3.19) and (3.17) involves integration of the equations with given an initial condition. This process called Inertial Navigation and it demonstrates the position and orientation of mobile robot.

Furthermore a model developed in Matlab Simulink to receive the raw data from IMU as input and calculate the position and orientation of robot in each sample time. It is crucial to calculate the initial orientation of body as accurate as possible, because it directly affects on estimation of position and orientation on next moments.

Two tests provided here that demonstrate the estimation of position and orientation after running the model by collected data from IMU. In both tests components of r_{eb}^e initialized at some point in Tampere region (base on meter) :

$$\text{Initial position} = \begin{bmatrix} 2794852.511 \\ 1236477.835 \\ 5579670.98 \end{bmatrix} \tag{3.21}$$

The first test indicates the robot when it is at rest for 61 seconds and then it starts driving in a straight path with the length of 108 centimeters till to 68^{th}

second. During the stationary period we stop the integration process of acceleration. During the time that robot is driven, bias of accelerometer is calculated by average of acceleration over the stationary period and initial gravity vector. For simplicity the value of bias is considered to be constant and thus it is subtracted from acceleration during driving period and during period the robot stopped to move.

Fig 3.5 shows the result of displacement of r^e_{eb} components in x,y,z directions in meter on vertical axis and time in second on horizontal axis. As you see in Fig 3.5 displacements before 61^{th} second is zero, but from 61^{th} to 68^{th} the displacement in x direction is around 127 centimeters, which has 19 centimeters error. Also there is displacement error in y and z direction around 10 and 8 centimeters respectively. After 68^{th} second the robot stops, but because of biases the value of displacement in all directions increasing fast.

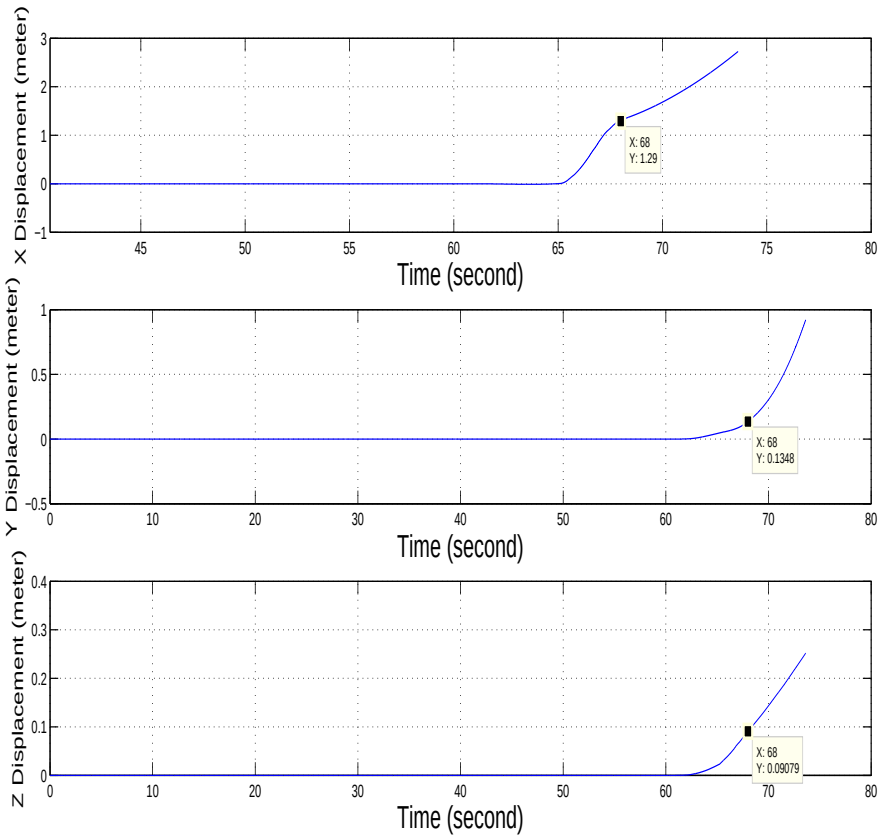


Fig. 3.5: Displacement test on straight line in X direction

In second test the heading of robot is changed by 90° through driving on a quadrant with 90 cm radius. Fig 3.6 shows the result in a diagram with degree on vertical axis and time in second on horizontal axis. After 60 seconds when robot was at rest, it starts driving on quadrant path till to 65^{th} second and the heading changed almost 90°. Also pitch and roll angles remained around zero. After 65^th seconds the robot stops again and biases increase the value of orientation in each direction. During stationary period we stopped integration process from angular

velocity. During the time that robot is driven, bias of gyroscope is calculated by average of angular velocity over the stationary period and then it is subtracted from angular velocity during driving period and during peridod that robot stopped to move.

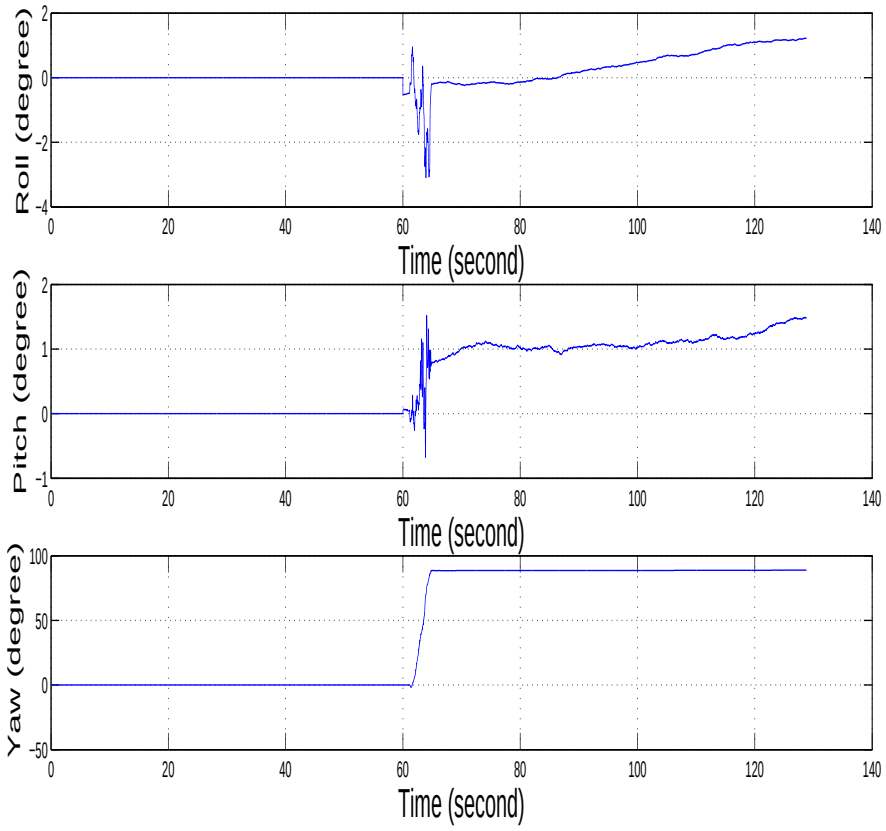


Fig. 3.6: Changing yaw angle

3.3 Inertial Measurement Unit and Camera Calibration

Data provided by each sensor is expressed in its own coordinate frame. In the case of sensor fusion, first we must provide a common coordinate frame for multiple sensors to express the data (For more detail refer to [24]). Therefore the process of calibration between sensors and deriving the rigid transformation matrix for each of them is a fundamental process to do. Calibration process of extroceptive sensors are more straight forward, but calibration of IMU with respect to a camera bring more difficulties because of different nature of sensors (see EM. Foxlin [25] or H. Zhao [26]). IMU calibration must be done with regard to kinematic and dynamic characteristic of vehicle.

Therefore a simulation model of mobile robot is implemented in Matlab Sim-Mechanics to test the calibration method easier and faster before real experiments. Therefore a model from the mobile robot is designed that consists of four actuators

to enable the robot to move in X and Y direction, changing the yaw and pitch angles. Two body sensors considered as IMU and camera and attached on main body and a marker placed in 10 meters away from mobile robot. All dimensions try to be same with real mobile robot. Moreover simulation made it possible to control the motion and defines the desire path for mobile robot. Even small changes in the system can be tested and the result can be seen by running the simulation.

Now to find the transformation matrix between IMU and camera, an equation base on the position of camera, IMU, marker is developed. IMU is located in a same position as body frame on mobile robot. Also the full pose of marker in camera is given (P_{MC}^{M} and R_{C}^{M}) and we assume the position of inertial coordinate frame is on the marker, thus:

$$R_{M}^{i} \cdot P_{MC}^{M} = P_{iB}^{i} + R_{B}^{i} \cdot P_{BC}^{B} \tag{3.22}$$

Where:
M, i, C, B are marker, inertial, camera and body frame respectively.
R_{M}^{i} and P_{BC}^{B} are the terms that we are looking for.
P_{MC}^{M} is already known from camera and P_{iB}^{i} is derived by integrating from IMU acceleration output expressed in inertial frame.
R_{B}^{i} is the rotation matrix from body frame to inertial frame which is calculated from IMU data.

If the value of P_{BC}^{B} and R_{M}^{i} would be calculated, it is possible to determine R_{C}^{B} as the rotation matrix of camera coordinate frame and body frame with respect to each other (refer to equation (3.34)). Thus when P_{BC}^{B} and R_{C}^{B} are cleared, the full pose of IMU and camera with respect to each other is cleared and they are calibrated with respect to each other.

To acquire P_{iB}^{i} we need to calculate two times acceleration integral from IMU. Even in the case of insignificant IMU error, two times acceleration integration involves more error in our calculation and that is what affect in result accuracy and we have to avoid. But the derivation of the above equation with respect to time is what that can help to replace real data with less error inside the equation

$$R_{M}^{i} \cdot V_{MC}^{M} = V_{iB}^{i} + \dot{R}_{B}^{i} \cdot P_{BC}^{B} \tag{3.23}$$

In above equation V_{MC}^{M} is velocity of camera with respect to marker and V_{iB}^{i} is the acceleration integral from IMU (velocity) represented in inertia frame. Note that the value of R_{M}^{i} and P_{BC}^{B} in equation (3.23) are constant over the time, but the value of rest terms can change with time.

$\dot{R}_{B}^{i}$ is the rate of rotation matrix that discussed earlier in equation (1.5) and based on that it is equal to:

$$\dot{R}_{B}^{i} = R_{B}^{i} \cdot \Omega_{iB}^{B} \tag{3.24}$$

and Ω_{iB}^{B} is skew symmetric matrix (based on (1.4)) of IMU angular velocity after removing biases terms :

$$\Omega_{iB}^{B} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{3.25}$$

Thus the value of matrix R_B^i in every second computed by integration from equation (3.24).

To calculate the value of V_{iB}^i, we do the same process like in (3.19). First we need to transform the IMU acceleration from body frame to inertia frame by multiplying with rotation matrix R_B^i. Secondly we need to subtract the value of acceleration due to gravity and bias from IMU acceleration, at this point velocity of body frame at every second is computed by a simple integration over time. Since the mobile robot start to move from stationary position, the initial value of V_{iB}^i in integration process is equal to zero vector.

On the other side of equation (3.23) there is V_{MC}^M and since the P_{MC}^M is known the position differentiation with respect to time can provide us the velocity.

Since equation (3.23) is linear with respect to measurement, we can apply linear operators like summation over the time . Therefore equation is reformed during period t_0 to t_n:

$$R_M^i \cdot (\sum_{n=t_0}^{n=t_n} V_{MC}^M) = \sum_{n=t_0}^{n=t_n} V_{iB}^i + (\sum_{n=t_0}^{n=t_n} \dot{R}_B^i) \cdot P_{BC}^B \tag{3.26}$$

Above equation solved with nonlinear optimization method known as fmincon in Matlab, in which linear inequalities constraints provided for the distance between IMU and camera P_{BC}^B and rotation angle between marker frame and inertia frame R_M^i.
Equation (3.26) can lead us to answer if it fulfills some properties such as:

Property I) Coefficient matrices of equation (3.26) must be full rank to avoid indefinite answers of equation. This means that summation of $\dot{R}_B^i$ over time should be full rank. In practice rotation about merely one of x, y, z axes, always produce a singular derivative rotation matrix (a row or column of zero component). As a result changing only the heading (or pitch or roll) angle of mobile robot is not enough and at least two different angles axes must change during t_0 to t_n to provide a full rank rotation matrix.
There are two methods here that we can change two angle axis at same time, first is to run the mobile robot only by changing the heading and save the data and next time run mobile robot only by changing pitch angle and save the data. At this point we can add two set of data base on equation (3.26).
Second method is to drive mobile robot in a way that both its heading and pitch angles changes with time, thus by substituting the data in equation(3.26), the final rotation matrix is:

$$R_B^i = (R_B^i)_{roll} \cdot (R_B^i)_{pitch} \cdot (R_B^i)_{yaw} \tag{3.27}$$

and since the roll angle remains constant, derivative of above formula is:

$$\dot{R}_B^i = (\dot{R}_B^i)_{yaw} \cdot (R_B^i)_{pitch} \cdot (R_B^i)_{roll} + (R_B^i)_{yaw} \cdot (\dot{R}_B^i)_{pitch} \cdot (R_B^i)_{roll} \tag{3.28}$$

Therefore the final derivative rotation matrix is affected by $(\dot{R}_B^i)_{yaw}$ and $(\dot{R}_B^i)_{pitch}$.

Property II) Second condition that equation (3.26) must contain become clear if we extract the last part of equation (3.26) at second t_n base on (1.5), then:

$$\dot{R}^i_B \cdot P^B_{BC} = R^i_B(\omega^B_{iB} \times P^B_{BC}) = (R^i_B\omega^B_{iB}) \times (R^i_B P^B_{BC}) = (\omega^i_{iB}) \times (R^i_B P^B_{BC}) \quad (3.29)$$

Again this equation illustrates the fact that if two components of angular velocity vector are zero, third component of cross product would be equal to zero and the equation regresses, thus we lose the accessibility to calculate unknown terms in equation.

Property III) Variable P^B_{BC} in equation (3.26) could be equal to zero if mobile robot is driven in a path periodically from. By driving mobile robot collected data for V^M_{MC} over the time indicates that there are points with the same value, but negative sign. Therefore the summation of data $\sum V^M_{MC}$ over the time in (3.26) would be equal to zero. The same thing happens for V^i_{iB} over the same path and finally it leads the variable in equation (3.26) to be equall to zero.

As an example the mobile robot start from an arbitrary point and drives in a circular path with 2 meter radius and $\pi/4$ rad/s as yaw rate for 20 second. With the help of equation (3.26), the variables during 20 second calculated. Only second component of variable P^B_{BC} based on meter is demonstrated in Fig 3.7 as an example and the graph indicates a curve shape that merge to a constant value which is about 0.519 meter equivalent to $(P^B_{BC})_y$. But as you see in the graph there is a jump at 10th which happened when mobile robot complete one round of circle and the next jump happened 8 second later at 18th second, because the period of circular path is 8 second.

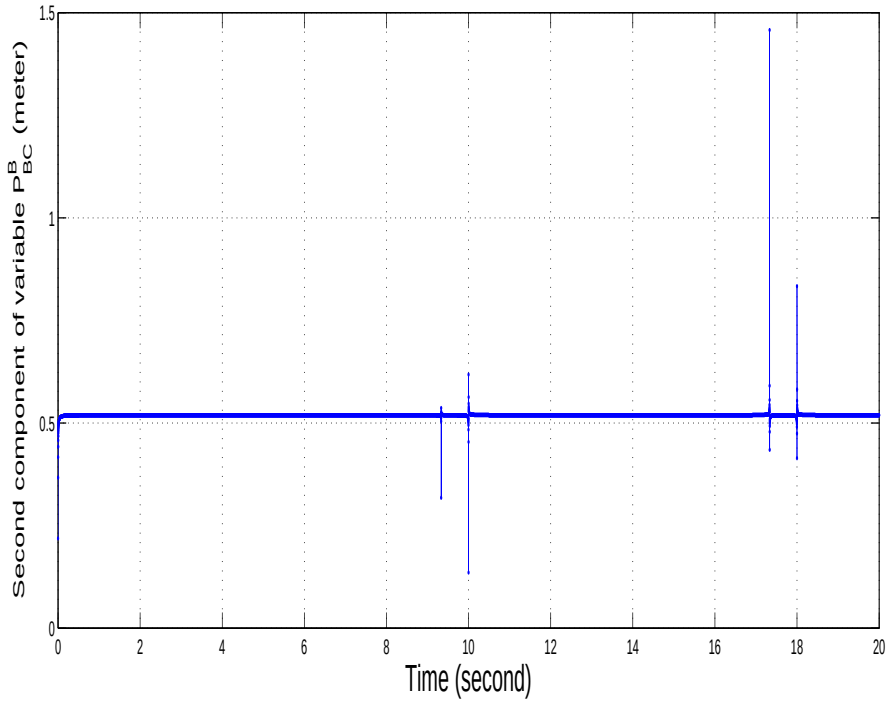


Fig. 3.7: Calculating $(P^B_{BC})_y$ component of variable P^B_{BC} over 20s in periodic circular path

By regarding the facts that mentioned above, the values of R_M^i and P_{BC}^B can be computed successfully by driving the mobile manipulator over a quadrant. As it is demonstrated in figure 3.8 in our experiment we drove the mobile manipulator with $\pi/20$ (rad/s) as yaw rate for 10 second with .31 (m/s) velocity to go through quadrant with 2 meters radius and also with $\pi/90$ (rad/s) as pitch rate for 10 second with same speed.

By replacing the data in equation (3.26) and solving the nonlinear equation, unknown terms calculated after 60 times iteration. The value of variable P_{BC}^B base on centimeter is calculated:

$$P_{BC}^B = \begin{bmatrix} -5.13 \\ 51.92 \\ 28.37 \end{bmatrix} \tag{3.30}$$

and the value of R_M^i base on radian is equal to:

$$R_M^i = \begin{bmatrix} 0.9187 & -0.2614 & 0.2961 \\ 0.2983 & 0.9505 & -0.0864 \\ -0.2588 & 0.1677 & 0.9513 \end{bmatrix} \tag{3.31}$$

which are very near to real values that equal to:

$$(P_{BC}^B)_{Real} = \begin{bmatrix} -5.1763 \\ 51.925 \\ 28.354 \end{bmatrix} \tag{3.32}$$

and

$$(R_M^i)_{Real} = \begin{bmatrix} 0.9187 & -0.2616 & 0.2961 \\ 0.2985 & 0.9505 & -0.0864 \\ -0.2588 & 0.1677 & 0.9513 \end{bmatrix} \tag{3.33}$$

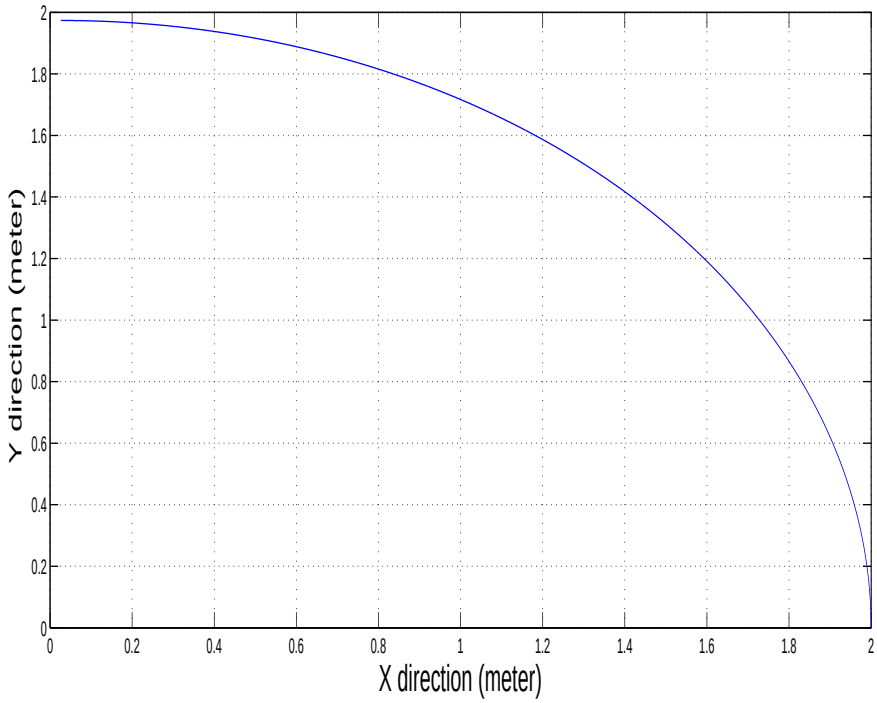


Fig. 3.8: Quadrant path with 2 meter radius

Now it is time to calculate R_C^B to complete the full pose of IMU and camra. For this means equation (3.34) indicates the relation between coordinate frames of marker, camera, body or imu:

$$R_B^i \cdot R_C^B = R_M^i \cdot R_C^M \tag{3.34}$$

In above equation R_B^i is the rotation matrix from body frame to inertial frame computed from equation (3.24).
R_M^i is rotation matrix from marker to inertial frame computed from equation (3.26).
R_C^M is rotation matrix from camera to marker that estimated from camera data. By differentiating with respect to time, we have:

$$\dot{R}_B^i \cdot R_C^B = R_M^i \cdot \dot{R}_C^M \tag{3.35}$$

That makes it possible to average over entire sets of data to determine R_C^B.

4. RESULT AND DISCUSSION

CANopen was developed as main communication interface to connect sensors and motor driver to embedded PC. Particularly Process Data Object protocols is implemented to send set points to the driver and Heart beat is implemented to monitor the status of components and Sync protocol is used to synchronize the actuators.

Since operating system of Embedded PC is Xenomai real-time, encoders and IMU were connected to embedded PC with real-time communication interface by Xenomai Real-time Driver Modules (RTDM) library. Programming in real-time environment (Xenmai) is quite helpful, because Xenomai provides sophisticated commands (API) for RTDM and message services to optimize time interval in communication with IMU and encoder and motor's driver. Hence different software modules are executed in different threads (multi-thread programming approach), real-time queue was implemented to share data among different software modules.

As it is demonstrated in Fig 4.1 now the network communication allows us to send control commands to motor's drive and receive feedback (from encoders) in exact time interval. We are able to control the steering position and driving speed of each wheel. Moreover sending control command to steering and driving wheels and using forward kinematic to calculate the velocity and pose of main body and using IMU output as feedback could be considered as future enhancement .

Besides many adavantages that CAN network provides, there were some cases that we couldn't run the robot because of mechanical malfunctions. For instance the CAN socket of motor drivers (EPOS) could be loose after a while or one of CAN low or CAN high wires were disconnected.

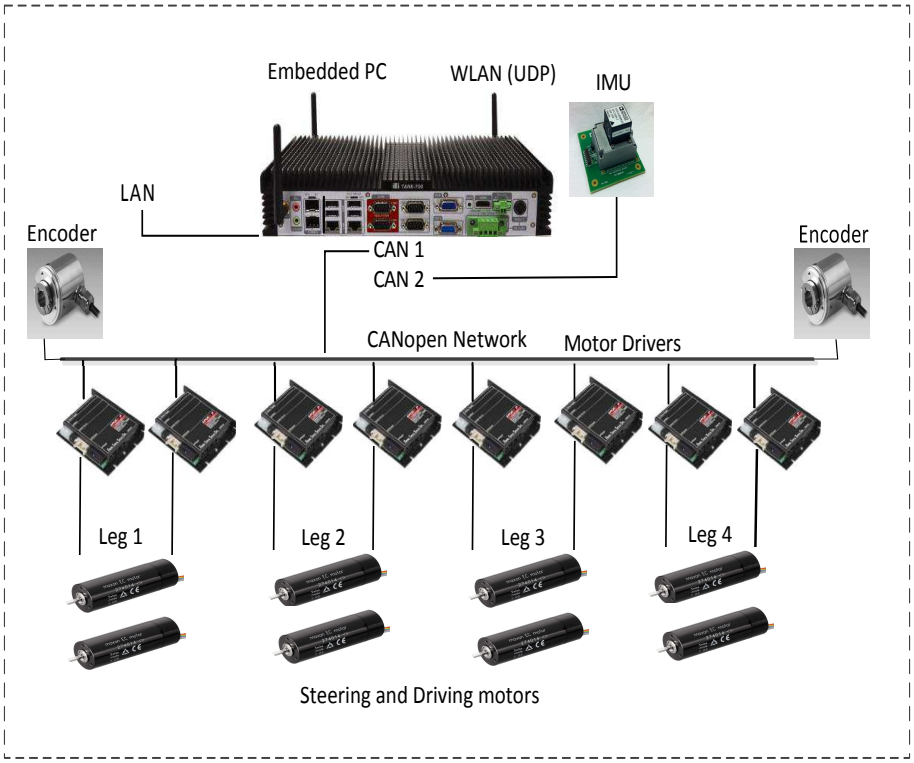


Fig. 4.1: Intelligent components and communication interfaces of iMoro platform

The process of receiving meaningful data from IMU and calculating the velocity and position of IMU body, which is attached on body of robot, is completed. However the modeling and calculations have been done in Matlab, this process needs time and concentration. By using Matlab code generation, now a model is designed on Matlab simulink to receive and read the IMU data real-time.(For more detail refer to [27])

During this process velocity and postion are calculated with regard to differential equations (3.17), (3.19), (3.20) and a test in section 3.2.5 is implemented to estimate the scale factor components of misalignment matrix. Also model (3.5) with four elements is considered to calculate biases of gyroscope or acceleration.

To improve postion estimation, IMU data can be fused with camera data (or GPS) by applying algorithms like Kalman Filtering as future enhancement of this section. (See M. Euston [28]).

As explained before the idea of sensor fusion has became a popular research topic recently (See P. nunez [29]). In section 3.3 the process of IMU and camera calibration is explained and the equation (3.26) is presented and solved when robot maintains some conditions as it follows a specific path. For example mobile robot must have rotation about two perpendicular axes, in our case by rotating about pitch and yaw axes. Moreover the path that robot follows must not be a periodic path, because solving the equation will lead to singularity. This properties is explained with more detail in 3.3.

Consequently if the robot follow Quadrant path in Fig 4.2, the collected data can solve the equation (3.26) and presents the transformation matrix between IMU

and camera. Thus the value of P_{BC}^{B} as position of camera with respect to body frame and is expressed in body frame based on centimeter is equal to (detail calculation is explained in section 3.3):

$$P_{BC}^{B} = \begin{bmatrix} -5.13 \\ 51.92 \\ 28.37 \end{bmatrix} \tag{4.1}$$

and the rotation matrix from marker to inertial frame based on radian is equal to:

$$R_{M}^{i} = \begin{bmatrix} 0.9187 & -0.2614 & 0.2961 \\ 0.2983 & 0.9505 & -0.0864 \\ -0.2588 & 0.1677 & 0.9513 \end{bmatrix} \tag{4.2}$$

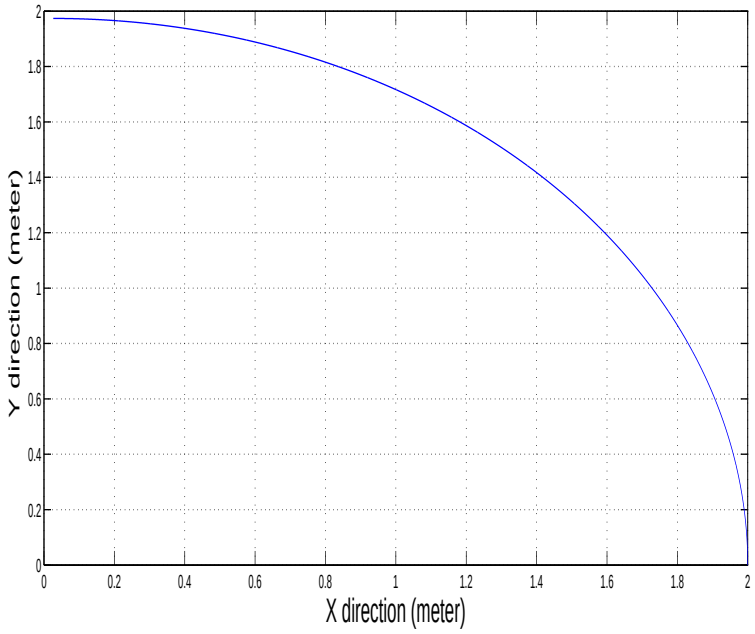


Fig. 4.2: Quadrant path with 2 meter radius

BIBLIOGRAPHY

[1] R. Oftadeh, M. M Aref, R. Ghabcheloo, and J. Mattila, "Mechatronic design of a four wheel steering mobile robot with fault-tolerant odometry feedback," in *Mechatronic Systems*, no. 1, 2013, pp. 663–669.

[2] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2301–2306.

[3] G. Welch and G. Bishop, "An introduction to the kalman filter," 1995.

[4] J. Vaganay, M.-J. Aldon, and A. Fournier, "Mobile robot attitude estimation by fusion of inertial data," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on.* IEEE, 1993, pp. 277–282.

[5] P. D. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*, 2013.

[6] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, 2006.

[7] H. Rajaie, O. Zweigle, K. Häussermann, U.-P. Käppeler, A. Tamke, and P. Levi, "Hardware design and distributed embedded control architecture of a mobile soccer robot," *Mechatronics*, vol. 21, no. 2, pp. 455–468, 2011.

[8] R. Grepl, J. Vejlupek, V. Lambersky, M. Jasansky, F. Vadlejch, and P. Coupek, "Development of 4ws/4wd experimental vehicle: platform for research and education in mechatronics," in *Mechatronics (ICM), 2011 IEEE International Conference on.* IEEE, 2011, pp. 893–898.

[9] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robotos.* The MIT press, 2004.

[10] R. Siegwart, P. Lamon, T. Estier, M. Lauria, and R. Piguet, "Innovative design for wheeled locomotion in rough terrain," *Robotics and Autonomous systems*, vol. 40, no. 2, pp. 151–162, 2002.

[11] R. Oftadeh, M. M. Aref, R. Ghabcheloo, and J. Mattila, "Bounded-velocity motion control of four wheel steered mobile robots," in *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on.* IEEE, 2013, pp. 255–260.

[12] O. Pfeiffer, A. Ayre, and C. Keydel, *Embedded networking with CAN and CANopen*. Copperhill Media, 2008.

[13] D. Abbott, *Linux for embedded and real-time applications*. Newnes, 2011.

[14] S.-h. Won, W. W. Melek, F. Golnaraghi *et al.*, "A kalman/particle filter-based position and orientation estimation method using a position sensor/inertial measurement unit hybrid system," *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 5, pp. 1787–1798, 2010.

[15] T. Lee, J. Shin, and D. Cho, "Position estimation for mobile robot using in-plane 3-axis imu and active beacon," in *Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on*. IEEE, 2009, pp. 1956–1961.

[16] Q. Cai, N. Song, G. Yang, and Y. Liu, "Accelerometer calibration with non-linear scale factor based on multi-position observation," *Measurement Science and Technology*, vol. 24, no. 10, p. 105002, 2013.

[17] A. Amirsadri, J. Kim, L. Petersson, and J. Trumpf, "Practical considerations in precise calibration of a low-cost mems imu for road-mapping applications," in *American Control Conference (ACC), 2012*. IEEE, 2012, pp. 881–888.

[18] A. Kim and M. Golnaraghi, "Initial calibration of an inertial measurement unit using an optical position tracking system," in *Position Location and Navigation Symposium, 2004. PLANS 2004*. IEEE, 2004, pp. 96–101.

[19] F. M. Mirzaei and S. I. Roumeliotis, "A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1143–1156, 2008.

[20] J. Underwood, A. Hill, and S. Scheding, "Calibration of range sensor pose on mobile platforms," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 3866–3871.

[21] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, "Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation," *Robotics, IEEE Transactions on*, vol. 25, no. 5, pp. 1087–1097, 2009.

[22] A. AUX_DAC, T.-A. MEMS, and A. RATE, "Six degrees of freedom inertial sensor adis16385."

[23] E. Nebot and H. Durrant-Whyte, "Initial calibration and alignment of low-cost inertial navigation units for land vehicle applications," *Journal of Robotic Systems*, vol. 16, no. 2, pp. 81–92, 1999.

[24] M. E. Antone and Y. Friedman, "Fully automated laser range calibration." in *BMVC*, 2007, pp. 1–10.

[25] E. M. Foxlin, "Generalized architecture for simultaneous localization, auto-calibration, and map-building," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 1. IEEE, 2002, pp. 527–533.

[26] H. Zhao, Y. Chen, and R. Shibasaki, "An efficient extrinsic calibration of a multiple laser scanners and cameras' sensor system on a mobile platform," in *Intelligent Vehicles Symposium, 2007 IEEE*. IEEE, 2007, pp. 422–427.

[27] R. Oftadeh, M. M. Aref, R. Ghabcheloo, and J. Mattila, "Unified framework for rapid prototyping of linux based real-time controllers with matlab and simulink," in *Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on*. IEEE, 2012, pp. 274–279.

[28] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing uav," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 340–345.

[29] P. Núñez, P. Drews Jr, R. Rocha, and J. Dias, "Data fusion calibration for a 3d laser range finder and a camera using inertial data." in *ECMR*, 2009, pp. 31–36.