UWB Tracking Software Development

Abstract

An Ultra-Wideband (UWB) two-cluster Angle of Arrival (AOA) tracking prototype system is currently being developed and tested at NASA Johnson Space Center for space exploration applications. This talk discusses the software development efforts for this UWB two-cluster AOA tracking system. The role the software plays in this system is to take waveform data from two UWB radio receivers as an input, feed this input into an AOA tracking algorithm, and generate the target position as an output. The architecture of the software (Input/Output Interface and Algorithm Core) will be introduced in this talk. The development of this software has three phases. In Phase I, the software is mostly Matlab driven and calls C++ socket functions to provide the communication links to the radios. This is beneficial in the early stage when it is necessary to frequently test changes in the algorithm. Phase II of the development is to have the software mostly C++ driven and call a Matlab function for the AOA tracking algorithm. This is beneficial in order to send the tracking results to other systems and also to improve the tracking update rate of the system. The third phase is part of future work and is to have the software completely C++ driven with a graphics user interface. This software design enables the fine resolution tracking of the UWB two-cluster AOA tracking system.
UWB Tracking Software Development

Julia Gross

Dickey Arndt, Phong Ngo, Chau Phan, John Dusl, Jianjun (David) Ni

UWB Systems Group / EV4
Avionic Systems Division / EV
NASA Johnson Space Center
Introduction

An Ultra-Wideband (UWB) two-cluster Angle of Arrival (AOA) tracking prototype system is currently being developed and tested at NASA Johnson Space Center for space exploration applications.

This talk discusses the software development efforts for this UWB two-cluster AOA tracking system.
Introduction

- **UWB** is an emerging communications technology unlike traditional Continuous Wave (CW) narrow band systems. The technology was approved by FCC in February 2002.

- **Features**: High data rate, fine time resolution and low power spectral density.

- **UWB system is used for communications with added passive tracking**
Overview

• Useful for tracking rovers and astronauts for early Lunar/Mars exploration missions when navigation systems (such as GPS) are not available

• SCOUT vehicle as test-bed for Lunar/Mars rover prototype

• Excellent tracking performance with less than 1% error at ranges up to 2000’
Utilize commercially available UWB radios and customize for our specific applications.

- Radios are development kits and are not plug and play.
- Radios essentially transmit and receive a UWB signal.

To use the radios:
- Develop the Algorithm
- Develop the Software
- Develop the Hardware

Overview
Functionality of the UWB Tracking Software

Input: Waveform data from Rxs

Algorithm: Takes data from Rxs and calculates position of Tx

Output: Position of Tx

Software:
Functionality of the UWB Tracking Software

- Goal is to track the position of the rover
- A transmitter and an antenna are located on the rover
- Four antennas, two receivers, a hub, and a laptop at the base station
- Base station is divided into two clusters, each with two antennas and one receiver
- The role the software plays in this system is to take waveform data from two UWB radio receivers as an input, feed this input into an AOA tracking algorithm, and generate the target position as an output.
Architecture of UWB Tracking Software

Input: Waveform data from Rxs

Core: Algorithm takes data from Rxs and calculates position of Tx

Output: Position of Tx
The architecture of the software includes the Input/Output Interface and the Algorithm Core.

**Input:**
- Waveform data containing timing information received from four antennas

**Core:**
- AOA tracking algorithm
- Extracts timing information from waveform data and calculates position of transmitter

**Output:**
- Converts the position data into GPS coordinate system to track rover
- Plots and displays the position data to show accuracy of the system
Input:
Parallel Multi-Socket Programming

![Diagram showing two UDP sockets, Rx 1 and Rx 2, receiving and processing UDP packets.](Image)
Waveform data is divided into 25 segments at radio

Software has a separate UDP socket for each of the two radios

Software collects one segment from one socket from receiver 1 and then takes one segment over the other socket from receiver 2

This alternating process is repeated until a complete waveform is assembled from each radio

The two radio are not synchronized. This process allows the collection and assembly of the waveforms at approximately the same instant in time to ensure the accuracy of the system
Phase I of UWB Tracking Software Development

Start → Matlab: Algorithm → Mex → Array → Plot Position → End

C++:
- Receive Data
- Socket → Rx 1
- Socket → Rx 2

Form Array
- UDP Packet 1
- UDP Packet 25
Phase I of UWB Tracking Software Development

- The software is Matlab driven with a C++ portion
- The C++ part of the program is a Matlab Mex function which treats the C++ portion as if it were just a Matlab function
- Matlab uses C++ to provide the communication links to the radios via UDP sockets
- Beneficial in the early stage when it is necessary to frequently test changes in the algorithm, better to use Matlab
- Drawbacks are the efficiency of opening and closing the sockets on each loop in Matlab – Also, plotting in Matlab takes up time
Phase II of UWB Tracking Software Development

**Diagram:**
- **Rx 1 and Rx 2:** Receives data via UDP packets.
- **Socket:** Intermediary component for data transmission.
- **Start:** Initiates the process.
- **Receive Data:** Processing of data.
- **C++:** Receives data from sockets.
- **Matlab:** Processes and outputs data (X, Y).
- **DLL Algorithm:** Generates output from Matlab.
- **Send Vehicle its Position:** Outputs position to the vehicle.
- **Plot Position:** Visual representation of data.

**Text:**
- UDP Packet 1
- UDP Packet 25
- Form Array
- Array
- + Array
- = Array
- MatLab
- X, Y