Demo: Small PLaNS Towards Mars: Exploiting Ultra-wideband for Self-localizing Rover Navigation

Nikola Janicijevic[1,2], Pablo Corbalán[1], Timofei Istomin[1], Gian Pietro Picco[1], Enrico Varriale[3]
[1] University of Trento, Italy gianpietro.picco@unitn.it
[2] University of Novi Sad, Serbia
[3] Thales Alenia Space, Italy enrico.varriale@thalesaleniaspace.com

Abstract

We describe a small-scale demonstration of the PLaNS system, which exploits ultra-wideband communication and ranging to support autonomous rover navigation in planetary exploration and other harsh environments.

1 Motivation and Scenario

The PLaNS project, based on a patent [1] by Thales Alenia Space, aims to support planetary exploration mission by providing an autonomous and self-deployed infrastructure enabling the localization and navigation of a mobile rover. Several types of battery-operated devices with energy-harvesting capabilities are envisioned: a lander, a rover, and a set of anchors. All devices include an ultra-wideband (UWB) transceiver to support *both* the communication and localization requirements of the project.

During descent onto the Mars surface, the lander ejects the anchors over a 1–5 km^2 area. The anchors then discover their neighbors and form a multi-hop network topology rooted at the lander. The anchors must also perform ranging (distance estimation) with their neighbors periodically, sending the measured distances to the lander. This, in turn, builds a relative coordinate system, computes an estimate of all anchor coordinates, and disseminates them to assist the navigation of the rover. The rover continuously discovers its surrounding neighbors and performs ranging with them, computing its own location on-board or offloading the computation task to the lander via long-range communication.

This demo presents a simplified version of the PLaNS scenario, showcasing *(i)* the building of the relative coordinate system and estimation of the anchor coordinates, and *(ii)* the navigation and tracking of the rover.

2 UWB and the DW1000

UWB radios based on impulse radio transmit a time-hopping sequence of very narrow pulses ($<$2 ns) that translate into a large bandwidth ($>$500 MHz) in the frequency domain [2]. The resulting large bandwidth leads to a very high time resolution enabling UWB radios to precisely estimate the time-of-arrival (ToA) of a signal and also measure a fine-grained version of the channel impulse response (CIR), which helps distinguish the signal leading path from multipath components. By precisely timestamping the transmission and reception of a packet, UWB radios can estimate the distance between two devices based on a ranging exchange.

The IEEE 802.15.4-2011 [3] includes an UWB PHY layer based on impulse radio. The DecaWave DW1000 is a standard-compliant UWB transceiver that provides centimeter-level ranging and reduces the power consumption by an order of magnitude w.r.t. previous UWB transceivers. This characteristic, along with the small footprint and the intrinsic ability of UWB to perform both communication and ranging, are a good match to the tight requirements of space exploration.

3 Demonstration System Description

We illustrate the features of the various devices in the demo, and offer brief implementation details.

Devices. The UWB platform is the DecaWave EVB1000 evaluation board [4] featuring an STM32F105 ARM Cortex M3 microcontroller and the DW1000 transceiver. We use the external PCB antenna provided with the evaluation kit.

The anchors are fixed UWB boards whose coordinates are known. The lander includes a Raspberry Pi (RPi) with an UWB node attached. The RPi collects ranging information from the lander via the serial port and provides a Web server that computes the relative coordinate system based on the anchor-to-anchor ranging estimates. The rover is a mobile robot [5] controlled by a RPi, to which an UWB node is attached. This RPi provides an interface to easily control the robot remotely (e.g., via smartphone); it also collects the ranging estimates from anchors and reports them via WiFi to the lander, which computes the rover location. This WiFi link mimics the long-range communication link envisioned in PLaNS between the lander and the rover.

Implementation. We rely on our port of Contiki OS for the EVB1000 platform [6] that provides *(i)* the core network primitives from the Rime stack [7], and *(ii)* popular ranging mechanisms such as single-sided (SS-TWR) and double-sided two-way ranging (DS-TWR) [3]. The Web server is implemented in Python and computes the rover position using a non-linear least squares solver.

4 Demonstration Highlights

We introduce the two separate phases of our demo, *initialization* and *navigation*, a Web application showcasing them to the demo audience, and provide measurements characterizing the performance of our localization mechanisms.

Initialization. This phase builds the relative coordinate system. The lander first discovers its neighbors by sending a broadcast message and collecting the responses sent by neighboring anchors with a random delay. This process is repeated until the lander has at least 3 neighbors, the minimum required to build a 3D coordinate system. The lander then schedules a ranging round per active node, including itself. In each round, a given node performs ranging using DS-TWR with all its neighbors in a round-robin fashion, reporting the distance measurements to the lander. This process is repeated until the lander obtains at least (by default) 15 samples for each anchor-to-anchor distance. These samples are exploited by the lander to build an adjacency matrix $\mathbf{A}$ whose elements a_{ij} correspond to the average distance from anchor i to anchor j. $\mathbf{A}$ is used to compute the coordinate system via trilateration, after which the UWB node on the lander becomes a simple anchor supporting the rover navigation.

Navigation. The goal of this phase is to track the rover position. First, the rover performs discovery in the same way as the lander. After discovering one or more neighbors, the rover interleaves discovery with ranging, following a round-robin approach similar to the initialization phase. Once the rover has discovered at least 4 anchors, the minimum required to enable its positioning, it stops discovery and only performs ranging with the current neighbors. If some of these fail during ranging, the rover resumes discovery until it has again 4 neighbors. Aboard the rover, the RPi collects the measured distances and sends them via WiFi to the lander Web server, which computes the rover location.

Web Application. The Web server interacts with two clients to visualize (e.g., on a browser) the behavior of our demo.

The first client focuses on the initialization phase. After the server has received the matrix $\mathbf{A}$, it computes the anchor coordinates and sends them, along with $\mathbf{A}$, to the client. This information is visualized together with the error w.r.t. the known coordinates of the anchors, which is also rendered graphically as a displacement on a map.

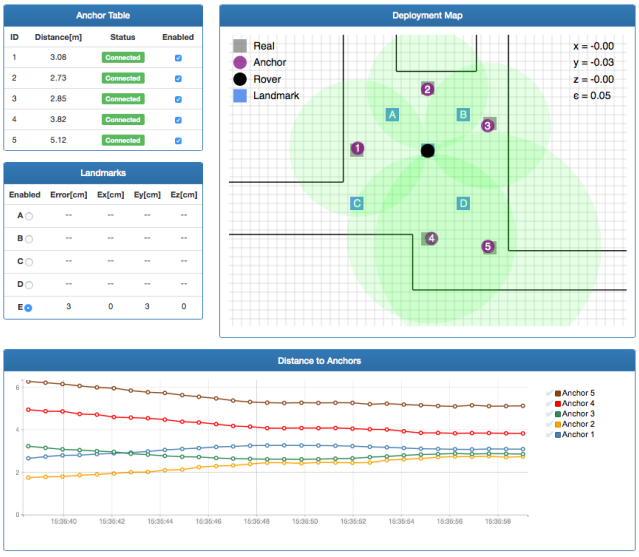

Figure 1: Web client application: navigation pane.

The second client, shown in Fig. 1, focuses on the navigation phase. The server updates this client after every message received from the rover. The client visualizes two tables, one for anchors and one for landmarks. The former shows the rover neighborhood and the distance estimates w.r.t. each anchor. Further, it includes a checkbox for each anchor, toggling its use in the localization solver, useful to assess the impact of each anchor on the location estimation. The landmark table enables us to easily assess the localization error of our system. When the rover is close to a landmark, the corresponding row in the table is enabled, showing the positioning error of the rover w.r.t. the (known) landmark coordinates.

The client also includes a map showing graphically the estimated and real anchor positions, the available landmarks, and the current location of the rover. The coordinates of the rover are updated in real time, along with the error estimate ε introduced by using the estimated anchor positions instead of the the real, known ones. The map also shows the measured anchor-to-rover distance as a green circle centered on the anchor, effectively visualizing the information used for trilateration. When the rover is within 50cm to an anchor this circle becomes yellow, mimicking the real-world alarm situation in which a rover could damage a nearby anchor. Finally, the client also plots the last 30 distances per anchor.

Evaluation. We ran experiments in a 7×7 m^2 area of an office building, therefore affected by multipath. We use 5 anchors that are placed on the floor, coherently with the PLaNS deployment scenario. One of the anchors serves as lander; all devices are in range of each other.

We analyze the average error (ε) and standard deviation (σ). First, we collected 148 adjacency matrices to compare the anchors estimated coordinates with the known ones. Our system achieves an anchor positioning error $\varepsilon = 15$cm with $\sigma = 19$cm. Then, we analyzed the accuracy of rover tracking in 5 landmarks (squares in Fig. 1) using the estimated anchor coordinates. The ranging error is $\varepsilon = 14$cm with $\sigma = 11$cm. This results in a rover positioning error of $\varepsilon = 12$cm with $\sigma = 13$cm; the 50^{th}, 75^{th}, and 99^{th} percentiles errors are 10, 15, and 62cm, respectively. Using the estimated anchor coordinates instead of ground truth ones introduces an increase in ε and σ of only 1cm and 6cm.

5 References

[1] F. Gottifredi and E. Varriale. Navigation system for exploring and/or monitoring unknown and/or difficult environments, 2013. US Patent 8,473,118.

[2] M. Z. Win and R. A. Scholtz. Impulse radio: How it works. *IEEE Communications letters*, 2(2):36–38, 1998.

[3] IEEE Std 802.15.4-2011. *IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2011.

[4] DecaWave. DecaWave ScenSor EVB1000 Evaluation Board, 2013.

[5] https://www.piborg.org/diddyborg.

[6] P. Corbalán, T. Istomin, and G. P. Picco. Poster: Enabling Contiki on Ultra-wideband Radios. In *Proc. of EWSN*, 2018.

[7] A. Dunkels, F. Österlind, and Z. He. An Adaptive Communication Architecture for Wireless Sensor Networks. In *Proc. of Sensys*, 2007.