Sensor Network Support for Real-time Indoor Localization of Four-rotor Flying Robots

Juergen Eckert, Falko Dressler and Reinhard German
Computer Networks and Communication Systems
Department of Computer Science 7, University of Erlangen, Germany
{juergen.eckert, dressler, german}@informatik.uni-erlangen.de

***Abstract*—We present a sensor network based indoor localization system that uses ultrasonic distance measurements for real-time localization of flying four-rotor robots. Such quadrocopters are on-board sensor controlled systems. They are very sensitive to lateral drifts, which cannot be compensated by mounted sensors. In our work, we provide a framework for time-of-flight based localization systems relying on ultrasonic sensors. It is optimized for use in sensor nodes with low computational power and limited memory. Nevertheless, it offers scalability and high accuracy even with erroneous measurements. We implemented the system in our lab using ultrasound sensor that are light enough to be carried around by the flying object. Using this real-time localization system, a position controller can be implemented to maintain a given position or course.**

I. Introduction

Flying four-rotor robots are similar to helicopters. In contrast to mono-rotor systems, these so-called quadrocopters usually provide more sensors and more robust controllers. A combination of gyrometers and acceleration sensors is used to determine its current state. Based on these measurements, a digital controller continuously adjusts the orientation of the platform. In such a way devices can easily be piloted by other digital systems such as a sensor network. By only controlling the pitch and the roll angles, the current position cannot be obtained. The quadrocopter always hovers on top of an air cushion. Thus, any minimal measurement error or any airflow may cause a drift to a random direction. The system remains highly in-stable w.r.t. position maintenance. Angle corrections must be permanently applied in addition to the board instruments to keep the flying robot in position.

Figure 1 shows the scenario. A quadrocopter is relying on an external positioning system to continuously update its system parameters. In general, there are many cases in which applications benefit from getting more accurate positioning information. A discussion of preferences for systems using active or passive mobile devices can be found in [1]. If privacy is an issue, passive localization systems should be preferred. For example, the infrastructure of the Cricket system [2] has no knowledge about the current position of any mobile device. However, this system architecture also has several disadvantages. The accuracy suffers if the mobile device moves during a series of (at least three) measurements. In some cases, e.g. using ultrasound, this is a strong limitation because a set of measurements can take up to several hundred milliseconds.

Therefore, we investigated appropriate real-time localization techniques and came up with a new solution that perfectly meets the needs in this application domain. We implemented a system based on ultrasonic distance measurements that is lightweight and can be carried by our quardrocopter. In summary, we not only provide a framework for our chosen scenario but also for other cases of real-time indoor localization. More detailed information can be found in [3].

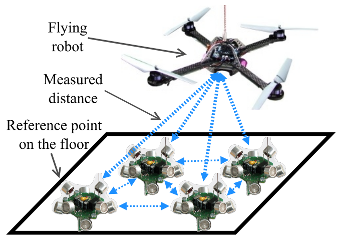

Fig. 1. Four-rotor flying robot hovers over reference points

II. Mathematical Procedure

This section covers the procedure of computing position information out of gathered distance measurements. We rely on ultrasonic distance estimation for time-of-flight (TOF) based lateration.

A. Preliminarities

We assume to start with a set of n tuples T_i, each consisting of a distance d_i to a reference point with a known position and the coordinates of this point $\overrightarrow{x_i}$:

$$T_i = (d_i, \overrightarrow{x_i}) : \overrightarrow{x_i} = (x_i, y_i, z_i)^T, i \in [1, n] \tag{1}$$

The trilateration problem can be solved for the unknown position $\overrightarrow{x} = (x, y, z)^T$ in different ways. Theoretically, the problem can be solved by a closed mathematical expression as shown in Equation 2. However, in practice, it is impossible to solve those n equations at once due to error-prone measurements.

$$(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 = d_i^2;\ i \in [1, n] \tag{2}$$

Several iterative optimization algorithms exist for the problem. For example, Foy [4] uses a Taylor-series estimation. At least for 2-dimensional problems, the method converges to a good solution within a few iterations. Another common approach is the use of an extended Kalman filter [5]. Abroy and co-workers [6] present such a non-iterative solution, however, with tremendous restrictions in terms of scalability and variability. Exactly three reference points, precisely oriented to each other are required: the coordinates have to be $\overrightarrow{x_1} = (0,0,0)^T$, $\overrightarrow{x_2} = (x_2,0,0)^T$, and $\overrightarrow{x_3} = (x_3,y_3,0)^T$. In order to apply this system to a general case, a coordinate transformation (offset and rotation) would be needed. Because this requires non-negligible computational effort, this method cannot be applied in many scenarios.

B. Position calculation

One common feature of all indoor location systems attracted our attention. Given that all reference points are mounted to the ceiling, the wall, or the floor, they all have one coordinate in common. Let us denote this as the z coordinate. We exploit this information for a closed position calculation.

First, a distribution of all tuples T_i into m subsets S_j to pairs of three different points must be done. The precise subset generation method will be explained later in Section II-C. For the moment, we assume we have m subsets that fulfill the condition that all z coordinates within a subset S_j are equal. Furthermore, it must be defined *a priori* whether the object to be localized is above the selected c_j, i.e. $z \geq c_j$, or below, i.e. $z \leq c_j$. Then, we can compute m possible coordinates for the unknown object out of the m subsets. Using a set of three single equations from (2) and taking the characteristics of each subset S_j into account, we can form a linear equation system:

$$A\overrightarrow{x} = \overrightarrow{b} : A \in \mathbb{R}^{2\times 2}, \overrightarrow{x} \in \mathbb{R}^2, \overrightarrow{b} \in \mathbb{R}^2 \qquad (3)$$

This 2-dimensional problem can be solved easily be applying Gaussian elimination. For the computation of the x and y coordinates, only simple arithmetic operations are needed such as addition, subtraction, and multiplication. Those are very basic (and fast) operations, available on low cost micro-controllers. The z coordinate can be generated in two ways. The easiest way is simply to measure it, which is straightforward using an ultrasound system. Alternatively, the already computed values can be inserted in Equation 2, which, however, requires a square root function for the used micro-controller.

C. Subset generation

In theory, one subset S_j, which contains three tuples T_i, would be sufficient for position estimation. However, taking measurement errors into account, more subsets are required.

Casas and co-workers [7] investigated all kinds of ultrasonic measurement errors. They came up with an average rate of measurement failure of $P_{mf} = 30\,\%$. A position estimation can only be successful if at least one correct subset S_j is used for evaluation, where a correct subset corresponds to one that contains only accurate measurements. P_m denotes the probability that none of the chosen subsets is correct. The required number of subsets can be calculated as follows [7]:

$$m = \frac{log(P_m)}{log(1-(1-P_{mf})^3)} \qquad (4)$$

Thus, for example, 11 subsets are required if we accept a failure probability of $P_m = 1\,\%$. Furthermore, the authors suggest that Monte Carlo techniques should be applied to randomly pick m subsets. However, more information about the subsets could help to improve the selection. In general, subsets with geometric shapes that minimize the error rate of the position calculation should be preferred (e.g., regular or well-formed triangles). Thus, the basic idea is to generate and, subsequently, to qualify a subset. Afterwards, it can be placed in a sorted list. Finally, the first m elements in this list are then used for the position calculations.

We decided to use a weighted combination of the average measured distances and the covered ground of the three points would be suitable. Both values are important for a well-formed but (mostly) non-regular tetrahedron (3 reference points plus the unknown point). The base area of the figure is a triangle. Usually, this can not be computed very fast because square root or trigonometric functions would be needed. Therefore, we used the cross product $\overrightarrow{n} = \overrightarrow{a} \times \overrightarrow{b}$ (with $\overrightarrow{a} = \overrightarrow{x_2} - \overrightarrow{x_1}$ and $\overrightarrow{b} = \overrightarrow{x_3} - \overrightarrow{x_1}$). Its length directly corresponds to the covered area. The base area is parallel to the x–y plane, so the cross product only contains a z component. This length can therefore be computed very fast.

D. Position estimation

Finally, the m possible positions (stored in $X(k+1) : X \in \mathbb{R}^{3\times m}$) have to be merged to one position $\overrightarrow{x}(k+1)$. The trivial approach would be the calculation the mean of all positions. However, outliers would significantly influence the result. Casas et al. [7] used an approach where a squared residual vector between all measured and all theoretical distances for each subset is computed. By taking the minimum median of the individual elements the influence of the outliers vanishes. Unfortunately, the computational effort for this method increases with the number of reference points and, therefore, is not very scalable.

We incorporated prior knowledge into the position estimator. Casas method [7] provides localization without any state information. However, already collected information could be exploited to gain better localization results. Thus, we split the estimation process into two steps in a similar way like an extended Kalman filter. In the first step, we predict the current position $\overrightarrow{x_p}(k+1)$ using a state vector:

$$\overrightarrow{x_p}(k+1) = \overrightarrow{x}(k) + \Delta\overrightarrow{x}(k,k-1) \cdot r \cdot \kappa(r) \qquad (5)$$

$$r = \frac{\Delta t(k+1,k)}{\Delta t(k,k-1)} \qquad (6)$$

For this vector, in each step we store the position and the localization time. The second step is slightly different from the original design of the filter. We generate the new position $\overrightarrow{x}(k+1)$ by selecting the nearest computed position to the predicted position out of the set $X(k+1)$.

The more time has elapsed since the last computation in relation to the last interval, the less reliable the prediction gets. The correction function $\kappa()$ in Equation 5 has been designed for compensating this effect. By using the ratio between the

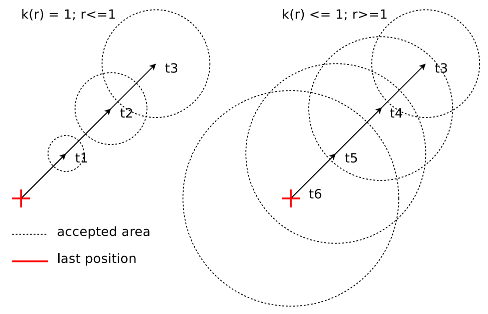

Fig. 2. Position prediction

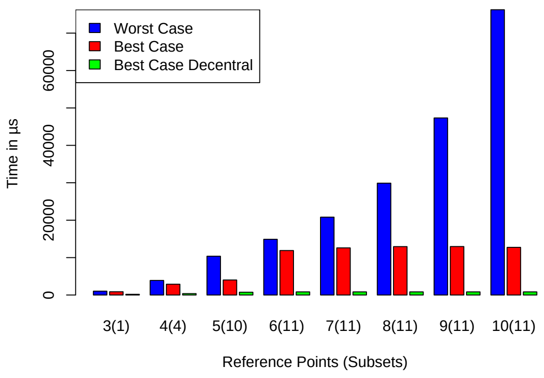

Fig. 3. Total localization time

localization attempts, this mechanism can be automated. Furthermore the absolute computation frequency is not relevant. $\kappa()$ is a function of r (Equation 6), which denotes the ratio of two time intervals. $\kappa()$ is a simple function that returns 1 for values between 0 and 1. For greater values, the output slowly decreases 0. Figure 2 illustrates the prediction vector and the growing space of the position acceptance. As shown on the left side, the prediction vector grows uninfluenced over time if the ratio r is smaller than 1 and, therefore, $\kappa()$ is 1. Thus, $\kappa()$ does not influence the prediction. The right side shows the situation if the ratio r increases beyond 1. This means that the last localization interval (i.e., the time between two accepted positions) was shorter than the elapsed time since the last position was accepted. Now, $\kappa()$ is being decreased because at this time a proper prediction based on the movement during the last interval can not be guaranteed.

III. Localization Performance

Scalability is one of the biggest issues in the context of sensor networks. In order to proof our localization algorithm works even on resource constricted embedded systems, we implemented the system and evaluated it in a lab scenario. In particular, we used the SunSpot sensor node platform [8] running JavaME as the host operating system. One of the key issues is the creation of the subsets. For reasons explained in Section II-C, we limited the number of subsets to 11. Independent of the number of reference points, an upper boundary for the classification can be given. The limitation of subsets implicitly restricts the position vector calculation time to an upper boundary, too. Thus, not every possible position needs to be calculated: Only positions from subsets that meet a certain threshold in the qualification are being considered.

Figure 3 shows the total computation time. The worst case scenario (blue) is a combination of the techniques that are not bounded in computational time. All subsets are computed and the residual based position estimation [7] was applied to the best 11 subsets. In the best case scenario (red), only techniques with a bounded computational time are used. So an upper boundary for the localization algorithm can be given independent of the number of used reference points. This is important to fulfill the real-time specification. The best case decentral scenario (green) describes the absolute minimal computational time consumption for the initiator of the localization, if subset grouping and position vector calculations are distributed on the entire sensor network. Unfortunately, the overhead of the communication latency is far too big to benefit from it, at least using our available hardware.

IV. Test System

Figure 1 shows the latest version of our ultrasonic measurement system including the sensor nodes. Despite the classical master-slave topology, we decided for a hybrid measurement architecture. Whether a device is master (transmitter) or slave (receiver) is completely hardware independent and can be controlled on application level. The detection field of the system is designed to be a hemisphere. Thus, the reference points on the floor can not only detect the flying object but also each other (this architecture is depicted in Figure 1). This way, it is possible to span up the grid automatically by attaching the reference points on top of mobile robots. Another advantage of a flying active beacon, as mentioned before, is that by sensing the TOF of its own active chirp the altitude of the object can be computed without the help of the localization infrastructure.

For the measurements shown in Figure 4, we placed the four-rotor robot at an arbitrary but fixed position over the detection field (four reference points arranged in a square of $2\,\mathrm{m}$ of edge length). It can be seen that there are four centers of gravity. Each subspace is the region for the computed position of one of the four possible subsets. Within this space, the maximum variance is about $\pm 2\,\mathrm{cm}$. The estimated position is normally confined to one of those regions. But as soon as the used subset is missing, the estimated point jumps to another subspace. The temporary vanishing of a subset can have two main reasons. First, one of the measurements was wrong and, therefore, the position was too far away. Secondly, the wireless communication may be disrupted. The generation of the regions is based on systematic errors of the reference points' positions. In our tests, we ensured an accuracy of about $\pm 3\,\mathrm{cm}$. With increasing deployment accuracy of the reference points, the resulting regions merge into a single one.

V. Communication

For communication we used the on-board radio capabilities of the SunSpot. It contains an IEEE 802.15.4 compatible physical interface (Chipcon CC2420) for wireless communication.

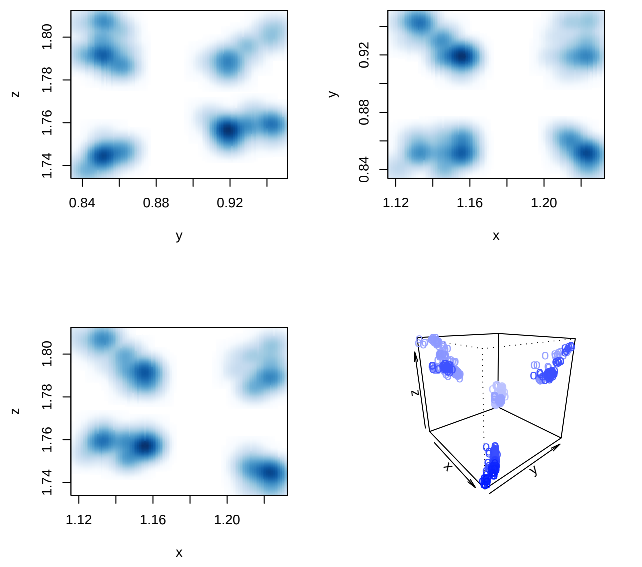

Fig. 4. Localization accuracy: the quadrocopter is fixed in a position, the measured x, y, z coordinates are plotted

Due to its low power specifications the emitted signal is not very strong and therefore the range is limited as well.

After performing a measurement, the measured values are distributed on the sensor network. These data have to be transmitted back to the active beacon for location computation. Our first attempt was to dispatch a request and to have all sensor nodes with relevant data reply to this using the Aloha technique [9] (using broadcast). But in our environment shown above the probability of getting three or more measurements responses were under $65\,\%$. Relying on unicast packets is not feasible, because if a packet can not be delivered, the whole transmission unit can be blocked for more than $10\,\mathrm{s}$. The impact on the flying object of course would be dramatic. The system would not be real-time capable any more. Therefore we developed a custom agent user-level protocol based on broadcasts. The initial frame is broadcasted by the active beacon without any payload. One field of the frame must contain an identifier. That allows the agent to de- and emerge while hopping over the nodes and collecting the data. Three things can happen, if a node receives an agent frame. If the agent frame is already known, it is simply dropped. If it is new to the system, a departure time is generated depending on its options. Additionally, relevant measurements are added before the departure. If an agent arrives with the same identifier as one in the departure queue (duplicate agent), new information from the more recent frame is copied to the one in the queue. Then, the agent frame is dropped. The data collection is not only possible on the initiator side but also on the rest of the system, if desired. All measured information can be made accessible on every sensor node in range without extra radio load.

For the latest agent protocol the sum of probabilities for three and four reference points per cycle is more than $95\,\%$ (compared to $67\,\%$ for the simple broadcast case). Furthermore, the probability for four reference points has significantly increased from $30\,\%$ for the broadcast case to about $82\,\%$. This allows for a much more precise position estimation.

VI. Conclusion

We investigated the problem of continuous indoor localization for flying autonomous robots. In contrast to ground-based robots, any waiting until position measurements have been completed or taking advantage of additional support systems such as odometry are not possible in this case. Thus, a real-time localization is needed that must also take weight constraints into account.

Considering these requirements, we developed an algorithmic procedure that advances the state of the art in indoor localization by being able to perform real-time localization based on possibly error-prone distance measurements. The basic assumption is that one coordinate of the reference points needs to be equal. Without loss of generality, we set the z coordinates to a constant value. This allows a closed mathematical calculation that is even possible to be performed by low resource sensor nodes. If, however, a coordinate transformation needs to be executed, the localization algorithm suffers from the computational complexity of this transformation. We implemented and evaluated the algorithm in our lab. The results demonstrate the feasibility of the solution. We consider our ultrasound lateration technique a necessary step for completely autonomous operation of flying robots in indoor environments.

References

[1] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha, “Tracking Moving Devices with the Cricket Location System,” in *2nd International Conference on Mobile Systems, Applications, and Services (MobiSys 2004)*. Boston, MA: ACM, June 2002, pp. 190–202.

[2] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, “The Cricket Location-Support System,” in *6th ACM International Conference on Mobile Computing and Networking (ACM MobiCom 2000)*. Boston, MA: ACM, August 2000, pp. 32–43.

[3] J. Eckert, F. Dressler, and R. German, “An Indoor Localization Framework for Four-rotor Flying Robots Using Low-power Sensor Nodes,” University of Erlangen, Dept. of Computer Science 7, Technical Report 02/09, May 2009.

[4] W. Foy, “Position-Location Solutions by Taylor-Series Estimation,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-12, pp. 187–194, March 1976.

[5] L. Kleeman, “Optimal estimation of position and heading for mobile robots usingultrasonic beacons and dead-reckoning,” in *IEEE International Conference on Robotics and Automation*, vol. 3, Nice, May 1992, pp. 2582–2587.

[6] M. Abreu, R. Ceres, L. Calderon, M. Jimenez, and P. Gonzalez-de Santos, “Measuring the 3D-position of a walking vehicle using ultrasonic and electromagnetic waves,” *Sensors and Actuators: A. Physical*, vol. 75, pp. 131–138, June 1999.

[7] R. Casas, A. Marco, J. Guerrero, and J. Falco, “Robust Estimator for Non-Line-of-Sight Error Mitigation in Indoor Localization,” *EURASIP Journal on Applied Signal Processing*, vol. 2006, pp. 1–8, 2006.

[8] R. Smith, “SPOTWorld and the Sun SPOT,” in *6th International Conference on Information Processing in Sensor Networks*, Cambridge, April 2007, pp. 565–566.

[9] N. Abramson, “THE ALOHA SYSTEM: another alternative for computer communications,” in *AFIPS Joint Computer Conferences*. Houston: ACM, November 1970, pp. 281–285.