Collision Prediction using UWB and Inertial Sensing: Experimental Evaluation

Aarti Singh
Electrical and Systems Engineering
Washington University in St. Louis, USA
aartisingh@wustl.edu

Neal Patwari
McKelvey School of Engineering
Washington University in St. Louis, USA
npatwari@wustl.edu

***Abstract*—Real-time proximity and collision detection via radio frequency (RF) distance measurements has application in smart helmets, drones, autonomous vehicles, and social distancing. In this paper we evaluate ACED, a range-based, infrastructure-free, distributed algorithm that utilizes inter-node range data and intra-node acceleration data to estimate the recent relative positions of each node and to predict impending collisions between any pair of nodes. The framework is tested and validated using experimental data from a testbed of mobile nodes which use ultra-wideband (UWB) ranging and inertial sensing. ACED is shown to outperform two state-of-the-art methods.**

***Index Terms*—Collision prediction, multidimensional scaling, autonomous vehicles**

I. Introduction

Autonomous and real-time collision prediction and collision avoidance is crucial in a world filled with multiple mobile entities operating in the close vicinity of each other. Collisions, which do happen [1], are both life-threatening and expensive. GPS and lidar are insufficient to reliably predict the collision of small objects moving quickly towards each other, e.g., multiple drones, or a smart helmet and a baseball. In many cases, it will be possible to add a radio frequency (RF) tag to nodes, vehicles, or objects that need to monitor to avoid collisions. However, RF tag localization systems are insufficient to predict collisions because they do not predict future positions, and they require a fixed, known-location infrastructure to calculate global map of nodes' locations, which may not be present, inconvenient, or expensive to deploy for the application. We argue that, fundamentally, collision prediction from range measurements should be distributed, local, and relative. Fortunately, collision between two objects does not require the coordinates of each object in a global coordinate system because the collision between two objects is a matter of their relative kinematics, such as their relative position, velocity, and acceleration. In this paper, we present a method to address this gap by enabling mobile agents of any size or speed to predict impending collisions without relying on a centralized infrastructure or a global reference.

A popular approach to obtain relative positions is multidimensional scaling (MDS). In MDS, 'dissimilarities' between each pair of objects are mapped into a low dimensional relative coordinates such that the distances between nodes are preserved as much as possible. MDS has been utilized in the localization research extensively [2], however, MDS is a centralized algorithm as it requires all dissimilarities to be known by one processing unit. For N nodes, classical MDS has a computational complexity of $O(N^3)$. A distributed method of estimating location is proposed in [3], is implemented with known-location infrastructure nodes. Based on the same work, another approach is presented to obtain a relative map of objects in motion, which although does not require known-location infrastructure, but is centralized in its implementation [4]. Another challenge with using MDS to generate relative kinematics over a time period is that, since there is no fixed frame of reference, the generated map can undergo random translation, rotation, and flip. Therefore, without infrastructure, successive applications of MDS over time provides incorrect kinematics. A modification of classical MDS such that a common frame of reference is maintained for position and higher order kinematics (velocity and acceleration) is implemented in [5]. The relative kinematics are estimated using higher order derivatives of squared distance measurements. However, this method is highly sensitive to noise in range measurements. In order to predict collision from RF range measurements we need relative kinematics estimators which are tolerant to noisy measurements. One extension of this work is implemented to produce a kinematics based collision prediction model, but it is limited to only linear motion [6]. We present a robust, decentralized, infrastructure-less algorithm 'Autonomous Collision Estimation for Dynamic Motion' (ACED), that produces quality relative kinematics of moving objects by using noisy inter-node range measurements and intra-node acceleration data. With the estimated kinematics, this distributed scheme predicts the future trajectory and any impending collision without requiring known-location devices. In addition, our solution can be complementary to other widely adopted technologies for collision prediction. These technologies involving either active sensors such as lidar [7] or radar [8], or passive sensors [9] such as cameras, are dependent on size, shape, reflective properties, luminescence of, and distance between the objects involved in the collision. The algorithm presented in this paper is free of such limitations involving physical properties of the objects.

II. Problem Statement

We take a network of N unknown-location nodes in a D-dimensional space. Over a period of time, node i collects

pairwise range measurements between itself and its neighbors $\mathcal{N}_i$. We denote the range measurement between node i and $j \in \mathcal{N}_i$ at time t as $\delta_{i,j}^t$. A real-world scenario is considered in which nodes move with arbitrary motion. The problems we explore include:

1) estimation of the coordinates $\boldsymbol{x}_i^t$ for $i = 1, \ldots, N$ and $t = 1, \ldots, T$, where $\boldsymbol{x}_i \in \mathcal{R}^D$ given pairwise range measurements $\{\delta_{i,j}^t\}$ and individual acceleration measurements, $\boldsymbol{\alpha}_i^t$, both taken over the time window $t = 1, \ldots, T$;
2) predicting an impending collision between i and any neighbor node in $\mathcal{N}_i$ at a time soon after $t = T$.

We assume that the primary goal of the system is to predict collisions, but in the case of an impending collision, the recent positions may be useful for the system reaction, for example, to know which direction to swerve.

III. Proposed Algorithm

To achieve the goals we articulate in the Introduction, in this section, we formulate a new cost function based on errors between measured and calculated ranges and acceleration over time for all nodes. Our insight is that distributed tracking can be formulated in a distributed, low computational complexity manner by using a majorization framework. In this framework, each device estimates its recent positions locally by minimizing its local cost function and broadcasting its newest position estimates to its neighbors. Each node successively refines its recent position estimates based on its range and acceleration measurements in addition to the most recent received position estimates from its neighbors. The majorization approach ensures non-increasing local cost functions. Since these local costs contribute additively to the global cost, thus, the global cost function will be non-increasing. Further, the local optimization step is low complexity because it is based on finding the minimum of a quadratic (majorizing) expression. Finally, the distributed optimization is guaranteed to converge, because the majorization approach guarantees each round's global cost is non-increasing. Our algorithm follows similarly to the *scaling by majorizing a complicated function* (SMACOF) [10] approach, but expands SMACOF to enable simultaneous estimation of multiple recent positions, and to enable use of acceleration measurements in the cost function.

A. Proposed Cost Function

As described in Section II, our problem is to estimate node positions $X = \{\boldsymbol{x}_i^t\}_{i,t}$ to match measured ranges $\{\delta_{i,j}^t\}$ and node acceleration measurements $\{\boldsymbol{\alpha}_i\}$. Our cost function S penalizes any coordinates that increase the squared error. We divide S into components for each node, $S(X) = \sum_i S_i(X)$, where the local cost function $S_i(X)$ is:

$$\sum_{t=1}^{T} \left[\sum_{j \in \mathcal{N}_i} w_{i,j}^t \left[\delta_{i,j}^t - d_{i,j}(X) \right]^2 + r_i \left[\boldsymbol{\alpha}_i^t - \boldsymbol{a}_i(X) \right]^2 \right],$$

where the first term represents the error between measured distances $\delta_{i,j}^t$ and the actual distances based on location coordinates $d_{i,j}(X^t)$, which are calculated as,

$$d_{i,j}(X) = \|\boldsymbol{x}_i^t - \boldsymbol{x}_j^t\| = \sqrt{(\boldsymbol{x}_i^t - \boldsymbol{x}_j^t)^T (\boldsymbol{x}_i^t - \boldsymbol{x}_j^t)}. \quad (1)$$

Whenever a node's velocity changes, its non-zero acceleration is measured by its accelerometer. We incorporate this extra information in the latter part of the sum, representing the error between the measured acceleration $\boldsymbol{\alpha}_i^t$ and acceleration $\boldsymbol{a}_i^t$ which calculated from the coordinate path travelled as:

$$\boldsymbol{a}_i(X) = \left(\boldsymbol{x}_i^{t+1} - \boldsymbol{x}_i^t\right) - (\boldsymbol{x}_i^t - \boldsymbol{x}_i^{t-1}). \quad (2)$$

Our approach finds $\hat{X} = \text{argmin}_X S(X)$, in a distributed manner, to provide location estimates $\{\hat{\boldsymbol{x}}_i^t\}$.

Note that $S_i(X)$ is local to i^{th} node since it only depends on the measurements available at i^{th} node and positions of its neighbour nodes. Minimizing $S_i(X)$ with respect to $\{\boldsymbol{x}_i^t\}_t$ results in new position estimates for node i. Implementing our approach at each node constructs the backbone of this distributed method. We use majorization at node i to guarantee non-increasing local cost.

Our method is described in Algorithm 1 in [11]. The algorithm is iterative and must be given initial position estimates. Generally, each time the algorithm is run, it is initialized using the coordinates of positions from the previous round's estimates, $\boldsymbol{x}_i^t$ for $i = 0, \ldots, N-1$. The first time a neighbor j appears to node i, it must provide its own locations, $\{\boldsymbol{x}_j^t\}_{t=0,\ldots,N}$. Here, we use classical MDS to generate any coordinates $\{\boldsymbol{x}_j^t\}$ for which there are no prior round estimates.

B. Regression Based Collision Prediction Algorithm

Using the relative locations estimated from previous stage, we predict locations into the near future. Regression analysis is widely used for prediction and forecasting, as it reveals the causal relationships between a dependent variable and one or a collection of independent variables. We choose quadratic regression since trajectories are quadratic in constant acceleration, and polynomial regression generally works well for non-linear interpolation problems. In our case, we predict the future trajectory of the node, which has a non-linear relationship with time due to the dynamic nature of the motion. The output of the polynomial regression in such a scenario can also be interpreted as higher order kinematics. Given T data points $(t, \boldsymbol{x}_i^t)$, where the independent variable t is a time instance and $\boldsymbol{x}_i^t$ is the corresponding location of node i at times $t \in \{1, \ldots, T\}$, we fit a 2^{nd} degree polynomial to approximate node i's location for real-valued t,

$$\hat{\boldsymbol{x}}_i(t) = \boldsymbol{p}_2 + \boldsymbol{p}_1 t + \boldsymbol{p}_0 t^2, \quad (3)$$

where, $\boldsymbol{p}_0$, $\boldsymbol{p}_1$, and $\boldsymbol{p}_2$ are the polynomial coefficients,which we estimate using the least squares approximation giving the estimated coordinates $\hat{X}$. Using the coefficients, the algorithm extrapolates future relative locations of each node, thereby, giving future inter-object distances of each pair of nodes. We are interested in the *near future*, i.e., the time-frame that is

equal to or less than the reaction time of the node, which is application dependent. If the minimum inter-node distance threshold between two nodes is crossed within this near future, a collision is predicted.

IV. Experiments

A. Hardware

We conduct a series of experiments with mobile nodes to predict collisions between any pair of nodes. Each node follows the architecture as described in [12]. Each such node is attached to a iRobot Create that moves the node as programmed. Lastly, a Raspberry Pi3 processor is attached on top of each node, which both lets us program the node movement, and measures the acceleration of each node via a BN0055 IMU sensor [13].

B. Multi-node Ranging Protocol

Each node measures ranges between itself and all other nodes, and no anchors are present in the system. An efficient way to measure all $\binom{N}{2}$ ranges between the N nodes is to use the efficient multi-node ranging protocol in [6], which requires only N message exchanges per cycle to get all the ranges.

C. Setup

We set $N = 4$ floor nodes to move as depicted in Figure 1 in a 6m $\times$ 6m area. Each mobile node (top right in Figure 1) undergoes acceleration as detailed in Table I, constantly between its starting position and its stopping position in Test I and II. Test III has node 3 in motion at constant speed, and due to its motion in a circle, the magnitude of its acceleration is 0.125 m/s^2. We collect UWB ranges between every pair of nodes at a rate of 18 ranges per second. The acceleration of each node is measured via the IMU sensors and collected by their attached Raspberry Pi at a rate of 100 samples per second. We route the ranges and acceleration data collected by each node to a central processing unit for offline algorithm testing and result generation. Note that this offline implementation is just for convenience during tests; our distributed algorithm can be implemented in firmware at each node, and will be our future work. Each Raspberry Pi is NTP time synchronized, such that the timestamps for ranges and acceleration can be matched to produce 18 range-acceleration pairs per second. Lastly, to record the ground truth coordinates of each node during each experiment, we use a 16-camera OptiTrack motion capture system, which enables millimeter accuracy [14]. The results are explained in Section V.

	Test I	Test II	Test III
Stationary Nodes	1	2	3
Mobile Nodes	3	2	1
Acceleration (m/s^2)	0.125, 0.09, 0.06	0.125, .06	0.125

TABLE I
Node Setup in Three Tests

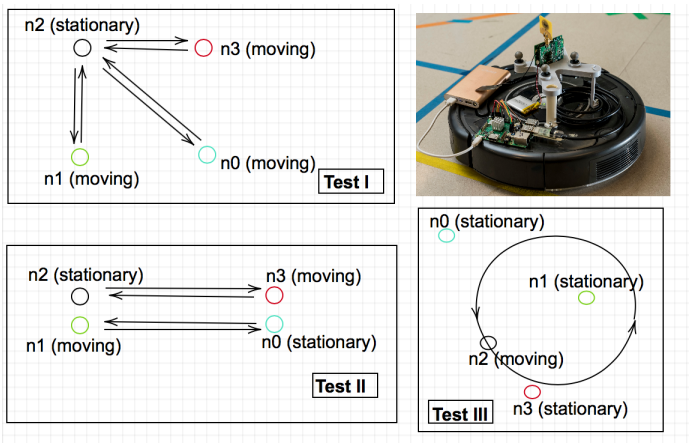

Fig. 1. Overview of the motions conducted by each node in three tests, and (Top Right) photo of hardware setup.

V. Results

A. Location Estimation

We first demonstrate the quality of location estimates generated from the ACED algorithm. For any of the 4 nodes, ACED estimates the trajectory that was followed over time. We use a window of $T = 20$ samples, thus each time the algorithm is run, we estimate $\{\boldsymbol{x}_i^t\}$ for $t = 1, \ldots 20$ and $i = 1, 2, 3, 4$. In a sliding window manner, ACED repeats by dropping the oldest time and adding one new time, and re-running the estimation for the next window of time. As a typical example, we plot the T location estimates for node 2 as '$\times$' and the ground truth locations as '$\circ$' in Figure 2. We also plot the location estimates from another state-of-the-art method, friend-based autonomous collision prediction and tracking (FACT) [6]. FACT assumes a constant velocity, and hence is unable to track the curved trajectory of node 2 from Test III at all. Furthermore, ACED is capable of predicting future positions based on (3), which are plotted for node 2 in the Figure 2 against the ground truth, where we define 'near future' as within 0.02 sec into the future.

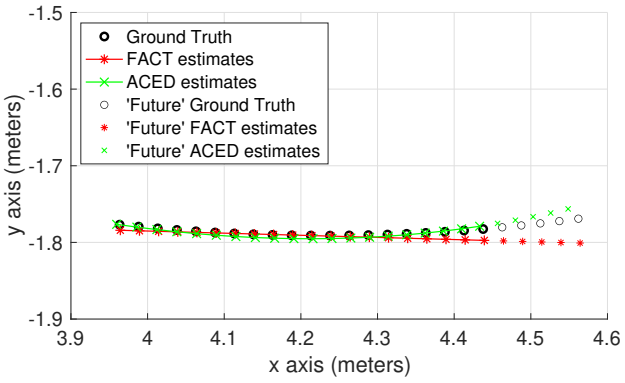

Fig. 2. Location estimates by ACED and FACT for the node moving with acceleration per window. ACED's predicted 'future' location estimates, are also plotted against ground truth.

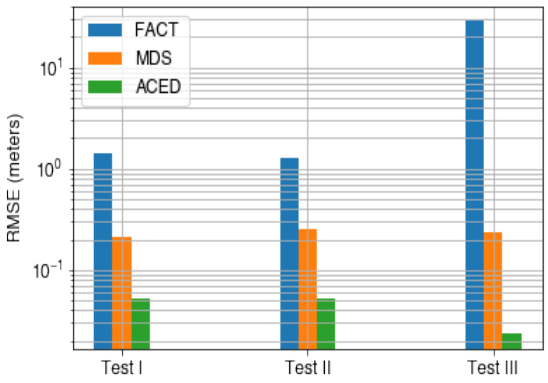

Fig. 3. Comparison of ACED vs. the modified MDS [5] and FACT [6] for each test, showing RMSE averaged across all nodes.

We use the following as our metric of error for the output of one window of any estimator:

$$RMSE = \left[\frac{1}{N} \sum_{i=1}^{N} \left\| x_i^{T/2} - \hat{x}_i^{T/2} \right\|^2 \right]^{1/2}, \quad (4)$$

and we report the average RMSE across all sliding windows across the entire test. Figure 3 plots this average RMSE of location estimated for all the nodes during each test by three methods, ACED, FACT[6] and modified MDS[5]. From [5], we used its centralized algorithm to find a globally optimum solution for relative position and velocity; however, it is known to perform sub-optimally in noise [5], [6].

Our results show that the FACT method of [6] diverges over time when motion is not linear. As described, each new run of the algorithm uses as initialization the trajectory estimates from the prior window. In Test III with a circular track for node 2, as FACT estimates a linear trajectory, its initialization from the prior window is poor. Over the course of Test III, its estimates at some point are unable to converge to the global optimum, after which it loses track of the coordinates and is unable to recover, leading to a very high average RMSE across Test III. ACED provides better tracking in dynamic motion, and doesn't have this convergence problem in our experiments. ACED also compares well to the centralized modified MDS method of [5], demonstrating a lower RMSE by 5-10x.

B. Interpolated Distance-based Collision Prediction

In order to avoid collision, a node must predict a collision before it happens. In ACED, if the distance $\hat{d}_{i,j}^t$ between nodes i and j in the near future t will fall below a threshold, d_{thd}, this counts as a future collision. Using the relative location estimates from ACED, we extrapolate pairwise distances into the near future. In this experiment, we define the near future as within $\tau = 0.02$ s.

We set the distance threshold d_{thd} to allow a trade-off between false alarms and missed detections. Letting r be the radius of one autonomous object, we would set $d_{thd} = 2r + \epsilon_d$ for some $\epsilon_d \geq 0$. By increasing ϵ_d, we would increase the probability of detection of a collision P_D while also increasing the probability of false alarm P_{FA}. A user could set the threshold based on the desired trade-off between the two. Figure 4 shows the ROC curve, i.e., the relationship between P_D and P_{FA}, compiled with data from across all three tests. ACED is able to provide higher P_D for the same P_{FA} when compared with FACT [6]. Note that even when FACT location estimates diverge, it manages to keep an accurate relative position and velocity for two nodes that are very close, and thus collision predictions are good. However, ACED cuts P_{FA} approximately by a factor of 2 for a constant P_D compared to FACT. Since ACED provides more accurate kinematics whenever nodes are accelerating, it can extrapolate complicated trajectories better, thus providing accurate future inter-node distances and kinematics to predict collisions. The 2^{nd} degree regression coefficients are able to extrapolate the future locations while taking each node's acceleration into account, a trait not achievable by FACT. We also test against the pairwise method of [15], which does not perform nearly as well as FACT or ACED.

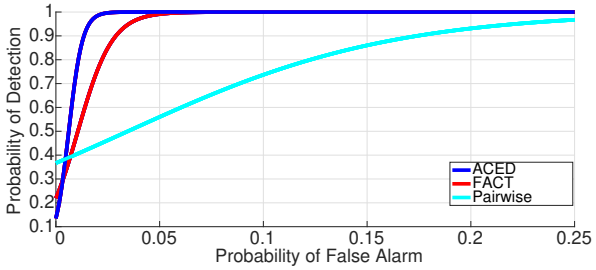

Fig. 4. ROC plot comparing ACED, FACT [6], and pairwise [15].

VI. Conclusion

This paper presents ACED, a new approach to estimate trajectories and predict collisions for systems involving mobile devices that are capable of measuring node acceleration and pairwise range measurements.The algorithm does not require a known-location infrastructure or a centralized computation. We test its performance in a network of four prototype mobile nodes mounted on ground robots in three tests. ACED predicts a node's trajectory with an order of magnitude lower RMSE, and collisions with a $> 2x$ lower false alarm probability, than three state-of-the-art infrastructure-free trajectory estimation and collision prediction methods.

Acknowledgment

This work is supported in part by the US National Science Foundation under Grant No. 1622741. We thank Alemayehu Solomon Abrar for sharing his code with us.

References

[1] D. Daneshvar, C. Nowinski, A. Mckee, and R. Cantu, "The epidemiology of sport-related concussion," *Clinics in sports medicine*, vol. 30, pp. 1–17, vii, 01 2011.

[2] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," in *Proc. 4th ACM Intl. Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '03, 2003, p. 201–212. [Online]. Available: https://doi.org/10.1145/778415.778439

[3] J. A. Costa, N. Patwari, and A. O. Hero III, "Distributed multidimensional scaling with adaptive weighting for node localization in sensor networks," *IEEE/ACM Transactions on Sensor Networks*, vol. 2, no. 1, pp. 39–64, Feb. 2006.

[4] B. Beck, R. Baxley, and J. Kim, "Real-time, anchor-free node tracking using ultrawideband range and odometry data," *Proceedings - IEEE International Conference on Ultra-Wideband*, pp. 286–291, 11 2014.

[5] R. T. Rajan, G. Leus, and A.-J. van der Veen, "Relative kinematics of an anchorless network," 2018.

[6] A. S. Abrar, A. Luong, G. Spencer, N. Genstein, N. Patwari, and M. Minor, "Collision prediction from uwb range measurements," *arXiv preprint arXiv:2010.04313*, 2020.

[7] P. Wei, L. Cagle, T. Reza, J. Ball, and J. Gafford, "Lidar and camera detection fusion in a real time industrial multi-sensor collision avoidance system," 2018.

[8] A. Viquerat, L. Blackhall, A. Reid, S. Sukkarieh, and G. Brooker, "Reactive collision avoidance for unmanned aerial vehicles using Doppler radar," in *Field and Service Robotics: Results of the 6th International Conference*, C. Laugier and R. Siegwart, Eds. Springer Berlin Heidelberg, 2008, pp. 245–254.

[9] R. Chellappa, Gang Qian, and Qinfen Zheng, "Vehicle detection and tracking using acoustic and video sensors," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, 2004, pp. iii–793.

[10] I. Borg and P. Groenen, "Modern multidimensional scaling: Theory and applications, volume 2 of statistics in social science and public policy," 1997.

[11] A. Singh and N. Patwari, "Range-based collision prediction for dynamic motion," in *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, 2021, pp. 1–6.

[12] A. Luong, P. Hillyard, A. S. Abrar, C. Che, A. Rowe, T. Schmid, and N. Patwari, "A stitch in time and frequency synchronization saves bandwidth," in *ACM/IEEE Intl. Conference on Information Processing in Sensor Networks (IPSN 2018)*, April 2018, pp. 96–107.

[13] K. Townsend, *Adafruit BNO055 Absolute Orientation Sensor*. [Online]. Available: https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/overview

[14] NaturalPoint. Motion capture systems - optitrack webpage. [Online]. Available: optitrack.com

[15] A. S. Abrar, N. Patwari, and J. Decavel-Bueff, "Demo abstract: Collision prediction from pairwise ranging," in *19th ACM/IEEE Intl. Conference on Information Processing in Sensor Networks (IPSN 2020)*, April 2020.