Sensor fusion of IMU and GPS for geofencing on an industrial control system for safe mowing in road areas

Stefan Dumberger[1], Raimund Edlinger[1], Philipp Bauer[1], Michael Zauner[1], Andreas Nüchter[2]

***Abstract*— The paper presents the implementation of geofencing on a mobile industrial controller for the automation of a mobile carrier vehicle for agricultural use. This system is used to prevent collision with well-known obstacles as well as avoid the accidental run-away of the vehicle under autonomous operation. It describes the mathematical principle used, as well as the additional features needed for a practical implementation on an industrial controller. As it is necessary to ensure the operation under real-world conditions, the paper also handles the data acquisition, pre-filtering and sensor fusion for system. Finally the correct behavior of the system is evaluated with multiple test-cases as well as experimental field tests.**

I. INTRODUCTION

This publication is part of a larger research project, where the goal is autonomous mulching of highway embankments using the agricultural tool carrier platform Reform Metron P48RC, depicted in Fig. 1. The focus of this paper in particular is the usage of geofencing by defining a limited area of operation and by extend preventing the robot to work in sections not intended to be mulched, collide with well known obstacles (e.g.: trees, pipes, trees, ditches, ...) or in the worst case drive onto the motorway.

The term geofencing, referring to the definition taken from Koch [2, p 11], describes a service for monitoring of a virtual boundary related to a physical area and raising a signal once a relevant object ether enters or leaves this area. While most geofencing systems use a geographical border defined via WGS84 coordinates, the concept itself can be used in any coordinate system where both the perimeter as well as the current position itself can be expressed relative to a common point of reference (e.g. a local map of a building).

The main complexity in our case is the reliability on this system to prevent potentially dangerous situations while being aware that the area of operation will contain a variety of objects, like trees or bridges, which will temporary interfere or completely block the reception of GNSS signals. Therefore an additional sensor fusion to bridge temporary outages was implemented.

II. RELATED WORK

Geofencing has a wide array of application on different scales in modern technologies. Fundamentally it can be divided into two main categories: The monitoring of entering a specific area and the check that a object or mobile platform is not leaving a specific area.

Fig. 1: Carrier platform with attachment and automation kit

The first use case is mainly used in logistics, where a region is formed around the destination of a delivery. Once the truck enters this region, a signal is sent to the logistics center, which can better plan the immanent arrival [7]. Another application is the collection of tolls. Once a vehicle gets into the vicinity of a tollbooth (approx. 200m), the system automatically requests the corresponding data from the provider. This results in the transaction already being handled once the vehicle reaches the tollbooth and the gate can open without any delay. This approach provides a high density of up to 300 vehicles per hour while providing individual billing conditions for different companies [3].

For the second category a variety of use cases can be found in various fields. For example [1] describes a method to use geofencing in agriculture for the monitoring of animal herds. In an logistical context the check for exiting a specific area can be used to examine if a vehicle is on its correct route or help in the case of theft of a vehicle [5].

When combining both the entering and leaving of a specific area, new applications arise for example when handling dangerous areas, emergencies or natural disasters. in these scenarios the systems described in [8] and [6] help in distributing information quickly, gathering information about the situation as well as self-organization for affected persons.

However in most of these examples the systems expect the equipment to have access to a GNSS signal all the time. When this assumption cannot be guaranteed the system can only provide convenience features, which may not work

[1]Authors are with University of Applied Sciences Upper Austria, 4600 Wels, Austria *forename.sirname*@fh-wels.at

[2]Andreas Nüchter is with Faculty Informatics VII - Robotics and Telematics, Julius-Maximilians-University Würzburg, 97070 Würzburg, Germany andreas.nuechter@uni-wuerzburg.de

all the time. Whenever a geofencing application must work under all circumstances, the system cannot rely on GNSS alone and in most cases additional infrastructure sensors to guard the perimeter are required (See [3] and [1]).

III. ALGORITHMIC APPROACH

The core principle of geofencing is to check whether or not a POI (point of interest) is within a predefined region. In our case the region is given via a list of boundary points creating a closed polygon.

However the simple check for the point to be inside the polygon is not enough in a practical use-case considering measurement uncertainty and the dynamics of the robotic systems. The border polygon needs to be scaled inwards to generate a buffer zone. In addition the current distance to the border can be interesting for adapting the maximum speed of the vehicle.

A. point-in-polygon algorithm

To decide whether or not a POI is inside the polygon, the Jordan curve theorem for polygons [4] is used. It generates a ray beginning at the POI and counts the intersections against all polygon edges $\overrightarrow{P_nP_{n+1}}$. Whenever the total count of intersections is odd, the POI is inside the polygon. To check for intersection the following equation can be used, using $C = P_n$, $D = P_{n+1}$, $A = POI$ and the arbitrary point $B = \begin{pmatrix} POI_x \\ POI_y + 1000 \end{pmatrix}$.

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{(B_y - A_y)(D_x - C_x) - (B_x - A_x)(D_y - C_y)} \cdot \begin{pmatrix} -(D_y - C_y) & (D_x - C_x) \\ -(B_y - A_y) & (B_x - A_x) \end{pmatrix} \begin{pmatrix} C_x - A_x \\ C_y - A_y \end{pmatrix} \tag{1}$$

An intersection can be found whenever $\alpha, \beta \in [0,1]$.

B. polygon scaling

The generation of the scaled down polygon needs to be handled with care, as this procedure can generate overlaps, as can be seen in Fig. 2a. If not removed, these overlaps will break the point-in-polygon algorithm for the newly created polygon. Therefore the following steps need to be taken:

1) transforming points inwards: In the first step four temporary points need to be generated for each polygon corner B using the previous corner A and next corner C. When ordering the corners in a counterclockwise manner, these points can be calculated using

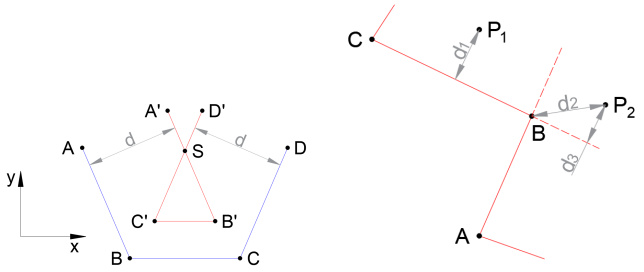
$$\begin{pmatrix} P_x \\ P_y \end{pmatrix} = \begin{pmatrix} X_x \\ X_y \end{pmatrix} + d \cdot \begin{pmatrix} \overrightarrow{e}^{\perp}_{XY\,y} \\ -\overrightarrow{e}^{\perp}_{XY\,x} \end{pmatrix} \text{ with } \begin{cases} X = A, Y = B \text{ for } P_1 \\ X = B, Y = C \text{ for } P_3 \end{cases}$$
$$\begin{pmatrix} P_x \\ P_y \end{pmatrix} = \begin{pmatrix} Y_x \\ Y_y \end{pmatrix} + d \cdot \begin{pmatrix} \overrightarrow{e}^{\perp}_{XY\,y} \\ -\overrightarrow{e}^{\perp}_{XY\,x} \end{pmatrix} \text{ with } \begin{cases} X = A, Y = B \text{ for } P_2 \\ X = B, Y = C \text{ for } P_4 \end{cases} \tag{2}$$

Afterwards a new corner B' can be calculated using the intersection between the two straight lines g_1 and g_2, defined as

$$g_1(x) = \frac{P_{2y} - P_{1y}}{P_{2x} - P_{1x} \cdot x + \frac{P_{2x} \cdot P_{1y} - P_{1x} \cdot P_{2y}}{P_{2x} - P_{1x}}}$$
$$g_2(x) = \frac{P_{4y} - P_{3y}}{P_{4x} - P_{3x} \cdot x + \frac{P_{4x} \cdot P_{3y} - P_{3x} \cdot P_{4y}}{P_{4x} - P_{3x}}} \tag{3}$$

and Cramer's rule to calculate the x- and y- component.

2) removing overlaps: As stated beforehand, this reduction of the polygon can lead to overlaps. To remove these phenomenons a section of four neighbouring corners A'. B', C' and D' is evaluated for each corner. Whenever the two vectors $\overrightarrow{A'B'}$ and $\overrightarrow{C'D'}$ intersect, using equation (1), the two inner points B' and C' are removed and replaced by the intersection point S.



(a) scaled polygon overlap (b) distance problem

Fig. 2: visualization of mathematical problems encountered in the algorithm

C. shortest distance to polygon

When determining the distance between the POI and the polygon, two scenarios can occur: The nearest polygon feature is ether a edge or a corner. Our approach assumes the nearest feature to be an edge in the first place and checks if this assumption was correct afterwards.

1) distance to edge: To calculate the minimal distance between the POI and an polygon edge, the edge is interpreted as a straight line defined by the two neighbouring corners. Then the normal distance between this line and the POI is calculated. However, as can be seen on the example of P_2 in Fig. 2b, this calculated distance can lay outside the confined section between the two polygon corners. To filter these outliers a rotated bounding box is used.

2) rotated bounding box: We define a rotated bounding box as a square around two points A and B. The length is equal to the distance between the two points, the orientation is the same as the vector $\overrightarrow{AB}$ and the width is defined two times the arbitrary distance d. With these constraints the four corners of the box can be calculated as follows:

$$\begin{pmatrix} p_{0x} \\ p_{0y} \end{pmatrix} = \begin{pmatrix} A_x \\ A_y \end{pmatrix} + d \cdot \begin{pmatrix} \overrightarrow{e}^{\perp}_{AB\,y} \\ -\overrightarrow{e}^{\perp}_{AB\,x} \end{pmatrix} \tag{4}$$

$$\begin{pmatrix} p_{1x} \\ p_{1y} \end{pmatrix} = \begin{pmatrix} A_x \\ A_y \end{pmatrix} + d \cdot \begin{pmatrix} -\overrightarrow{e}^{\perp}_{AB\,y} \\ \overrightarrow{e}^{\perp}_{AB\,x} \end{pmatrix} \tag{5}$$

$$\begin{pmatrix} p_{2x} \\ p_{2y} \end{pmatrix} = \begin{pmatrix} B_x \\ B_y \end{pmatrix} + d \cdot \begin{pmatrix} \overrightarrow{e}^{\perp}_{ABy} \\ -\overrightarrow{e}^{\perp}_{ABx} \end{pmatrix} \quad (6)$$

$$\begin{pmatrix} p_{3x} \\ p_{3y} \end{pmatrix} = \begin{pmatrix} B_x \\ B_y \end{pmatrix} + d \cdot \begin{pmatrix} -\overrightarrow{e}^{\perp}_{ABy} \\ \overrightarrow{e}^{\perp}_{ABx} \end{pmatrix} \quad (7)$$

Using the normal distance calculated in the last step as d, this value is valid minimal distance as long as the POI is inside the polygon defined by P_1. P_2, P_3 and P_4.

3) distance to corner: If no suitable candidate is found using this method on all corners of the polygon, the nearest feature to the POI is by definition a corner. This distance can now be calculated using the pythagorean theorem between the POI and every polygon corner. The lowest result is automatically the globally smallest distance.

IV. Implementation

The POI algorithm was implemented on an industrial controller communicating with an external GNSS module. For the analysis of accuracy and repeatability, the measurement system has been prototyped for the first outdoor tests.

A. Hardware Architecture

The hardware implementation for this project is shown in Fig. 4 and split into two dedicated hardware modules to maximize flexibility.

1) Sensor board: Raw sensor data acquisition is done on its own PCB to allow the independent data access over network from multiple computation units. In addition to the sensors, the board contains multiple voltage regulators and an ATMEL ATxmega32A4 microcontroller, which reads out all sensor values in an fixed frequency and provides the measurements over the network.

The inertial measurement unit (IMU) on the board is a TDK InvenSense MPU9250. It can measure linear accelerations up to $\pm 16g$, angular velovity up to ± 2.000 degrees per second, and a magnetic field strength up to $\pm 4900 \mu T$.

Finally the board uses a u-blox NEO-M8N module for global positioning. This module is technically able to use the free services provided by GPS, GLONASS, Galileo and BEIDOU. However in the scope of this project only GPS and GLONASS were enabled to test whether the system could also work on low-cost hardware.

2) Industrial controller: All further data processing and computation is done on a X90CP174.24-00[1]. This industrial controller is based on an 650 MHz ARM processor with 256 MB SDRAM and was chosen due to its IP 67 rating and optional support for a dedicated and certified safety CPU module.

All code is written in ANSI C and uses the B&R Automation real-time operating system to communicate between the individual software modules and also interact with other software running on the controller at the same time (e.g. robot controller, data logging, ...).

[1]https://www.br- automation.com/de/produkte/steuerungssysteme/x90-mobile- steuerungssystem/x90-mobile-steuerung/x90cp17424-00/

3) Communication: To transmit data between the sensor board and the X90 controller can be achieved via two different modes: The first method uses a TCP connection between both partners. This allows the detection of a new connection, synchronization and automatic retransmission of lost or corrupted packets. On the other the second method uses UDP broadcasts. While this mode cannot correct any transmission errors, it is possible to provide the data to multiple participants simultaneously and not affect the overall network load.

B. Software Architecture

While the algorithm described in section III describes the core approach of geofencing, it cannot be used directly and additional preprocessing of the data needs to happen beforehand. The overall software architecture is split into multiple modules and their interaction depicted in Fig. 3.

C. IMU parser

The IMU parser module converts the incoming data stream from the sensor board into individual data packets. In Addition this module is responsible to filter corrupted packets and obvious measurement errors. Handling these errors and outliers in the very front of the processing pipeline helps to produce a overall smoother position estimation.

D. AVD module

The AVD (Acceleration - Velocity - Distance) module calculates speed and distance travelled based on the IMU data. As this process depends heavily on integration of measurements, of all modules it is the most susceptible to sporadic sensor errors. A special consideration has to be taken during the initialization of this module. As the whole software uses geo-referenced coordinates, the default value of (0,0) would let the system start somewhere in the Atlantic Ocean and would result in a gradual adjustment over approx. half an hour until the position is usable. To prevent this behavior, the AVD module is reset whenever a positional error of more then 500m is detected and re-initialized with the current measured GPS position.

E. Kalman filter

This module combines the measurement from the GPS sensor with the output from the AVD module to determine the current position of the robot and by extension to generate a POI for the geofencing point-in-polygon algorithm. The combination of both independent data sources enables the filtering of the random walk inherently present in every GNSS system, the compensation of temporary drift near building or large objects and the bridging of short events without GNSS information like driving under a bridge.

F. CoordTransform

The CoordTransform module translates all GPS positions from Cartesian coordinates into the UTM reference frame. This transformation is necessary as working in the nonlienear cartesian space would violate some mathematical assumptions of independence and by extension would break or at least impair the results of both AVD and kalman filter.

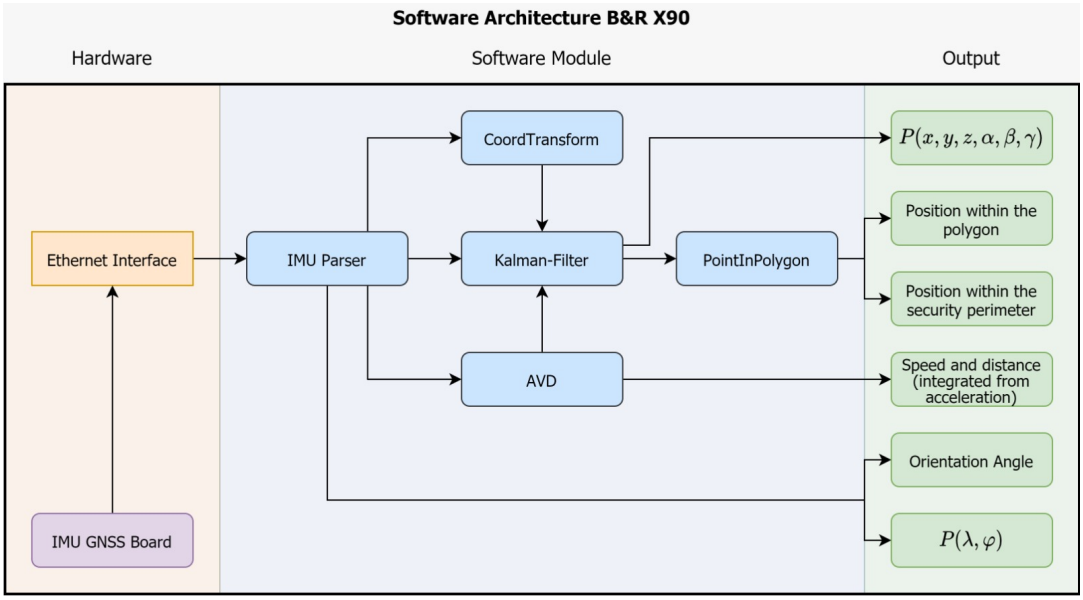


Fig. 3: Software architecture on the X90 controller

V. EXPERIMENTAL RESULTS

In this section, both lab and real-world test results are discussed. The detailed evaluation of all hardware and software components would exceed the scope of this paper, therefore we focus mainly on the newly developed point-in-polygon module.

A. *Evaluation of algorithm and implementation*

For the evaluation of the algorithm a black box testing approach was chosen. The X90 Controller is fed with synthetic data from a Matlab script and the results are checked automatically against the Matlab implementation of the algorithm and manually by visualizing the results as a plot, where the polygon is drawn in black, the POI in red, an the scaled polygon in green. In addition the nearest polygon feature in each plot is highlighted in purple.

For debugging purposes the values chosen for both polygon points and the POI do not use reasonable UTM coordinates, but an arbitrary reference frame around the point (250,250). However relative distances between points correspond to the scale expected in real-world scenarios. Examples of relevant test cases are listed below.

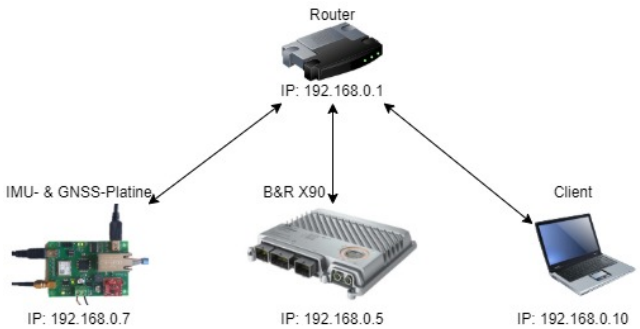


Fig. 4: Hardware and network configuration

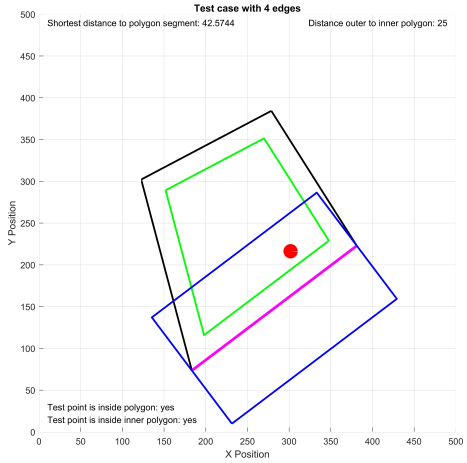


Fig. 5: Test against a polygon with four edges.

1) Simple polygon: The first test case checks the basic operation of the algorithm. In this case the number of polygon edges was limited to four. Fig. 5 shows the correct behaviour: The POI was evaluated both inside the original as well as the scaled polygon and the correct nearest feature was recognized, marked by the blue rotated bounding box.

2) Polygon scaling: In the next test case the main focus was, whether the implementation could handle a variable number of polygon edges and if the calculation of the scaled down polygon worked as expected. Therefore polygons with both convex and concave corners were generated. The result in Fig. 6 shows the correct scaling and expected behaviour.

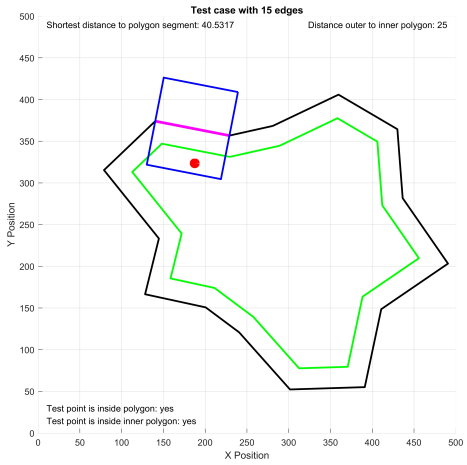


Fig. 6: Example of a scaling test case.

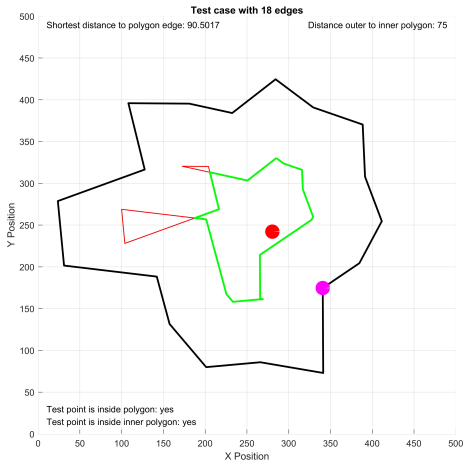


Fig. 7: Overlap test case

3) Overlaps and distance to corner: The final use case depicted in Fig. 7 depicts the evaluation of two functions. On the one hand the plot shows the nearest detected polygon feature to be a corner which is correct. On the other hand the distance for the scaled polygon was increased to force overlaps to occur. The algorithm filtered everything as expected, indicated by the sections marked in red being removed.

B. Field tests

To test the system under real-world conditions, all components were mounted on the mobile cattier platform depicted in Fig. 1 as part of the automation kit. A local and freely accessible meadow was chosen as a test location due to the combination of open field and some trees which may interfere with the GNSS signal. The perimeter polygon was created using coordinated taken from Google Maps and consists of the eight points listed in Table I.

TABLE I: Coordinates of the test area

Point	Coordinates in degrees		Coordinates in UTM (U33)	
	North	*East*	*North*	*East*
P0	48.22060	14.10073	5341210.375	433205.289
P1	48.22060	14.10053	5341210.549	433190.534
P2	48.22097	14.10053	5341251.674	443190.915
P3	48.22097	14.09995	5341252.178	443147.836
P4	48.22158	14.099450	5341320.414	433111.493
P5	48.22177	14.09972	5341327.472	433173.232
P6	48.22165	14.10028	5341312.353	433230.254
P7	48.22152	14.10105	5341312.353	433230.254

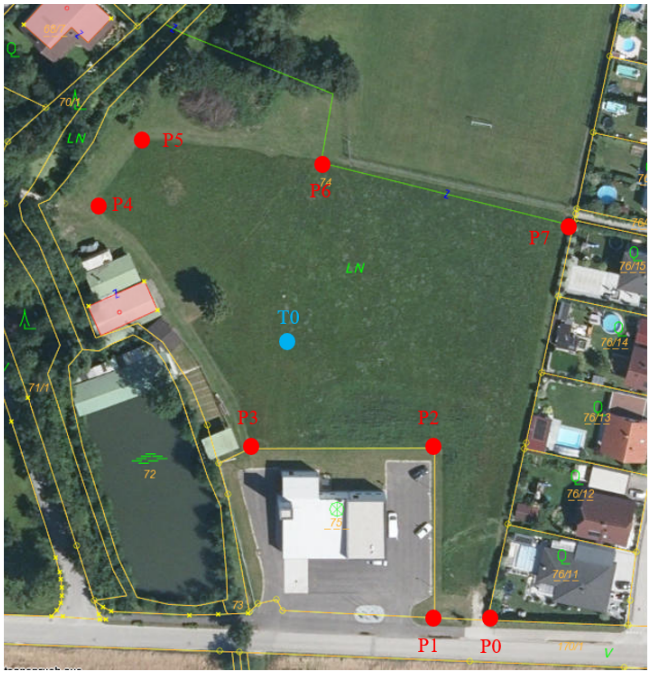


Fig. 8: Representation of the test area

The vehicle was placed on the start position T0 (see Fig. 8) and the autonomous operation was activated. However this resulted in an emergency stop due to the current position being outside the perimeter. The error could be traced back to the GNSS module measuring a position off by about 30 meter in reference to the actual location and therefore physically outside the perimeter. Forcing the correct GPS data by hand using a debugger, showed the geofencing to work as expected. However for a reliable and long term solution, the change to a DGPS system as positional reference will probably be necessary.

VI. SUMMARY AND OUTLOOK

In summary this work shows a software system for reliable geofencing running on an industrial grade controller provided the sensor measurements being correct. While not pursued at the moment, due to current lack of practical tests, the approach also shows potential for being moved to the safety CPU of the X90 controller to provide a functionally safe and certifiable module in the future.

ACKNOWLEDGMENT

The research of these results has been accomplished within the SMARTER - Slope Maintenance Automation using Real-Time Telecommunication and advanced Environment Recognition project. This work has been funded by the Austrian Research Promotion Agency (FFG) within the program "Mobility of the future" nr. 879646.

REFERENCES

[1] Q. M. Ilyas and M. Ahmad, "Smart Farming: An Enhanced Pursuit of Sustainable Remote Livestock Tracking and Geofencing Using IoT and GPRS," *Wireless Communications and Mobile Computing*, vol. 2020, pp. 1–12, Dec. 2020. [Online]. Available: https://www.hindawi.com/journals/wcmc/2020/6660733/

[2] F. Koch, K. Lakkaraju, and F. Meneguzzi, Eds., *Agent Technology for Intelligent Mobile Services and Smart Societies: Workshop on Collaborative Agents, Research and Development, CARE 2014, and Workshop on Agents, Virtual Societies and Analytics, AVSA 2014, Held as Part of AAMAS 2014, Paris, France, May 5-9, 2014. Revised Selected Papers*, 1st ed., ser. Communications in Computer and Information Science. Berlin, Heidelberg: Springer Berlin Heidelberg : Imprint: Springer, 2015, no. 498.

[3] S. K. Nagothu, "Automated toll collection system using gps and gprs," in *2016 International Conference on Communication and Signal Processing (ICCSP)*, 2016, pp. 0651–0653.

[4] L. Narens, "A nonstandard proof of the Jordan curve theorem," *Pacific Journal of Mathematics*, vol. 36, no. 1, pp. 219–229, Jan. 1971. [Online]. Available: http://msp.org/pjm/1971/36-1/p20.xhtml

[5] R. R. Oliveira, I. M. Cardoso, J. L. Barbosa, C. A. da Costa, and M. P. Prado, "An intelligent model for logistics management based on geofencing algorithms and RFID technology," *Expert Systems with Applications*, vol. 42, no. 15-16, pp. 6082–6097, Sept. 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417415002316

[6] R. Passarella, S. P. Raflesia, D. Lestarini, Taufiqurrahman, R. F. Malik, Sutarno, H. Ubaya, and A. Rifai, "Disaster mitigation management using geofencing in indonesia," in *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, 2017, pp. 1–4.

[7] F. Reclus and K. Drouard, "Geofencing for fleet amp; freight management," in *2009 9th International Conference on Intelligent Transport Systems Telecommunications, (ITST)*, 2009, pp. 353–356.

[8] A. Suyama and U. Inoue, "Using geofencing for a disaster information system," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, 2016, pp. 1–5.