Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 9734–9739

ICS-Zooids - An Experimental Testbed for Cooperative Control Strategies

Nirmal Rathakrishnan * **Patrick Göttsch** * **Herbert Werner** *

* *Institute of Control Systems, Hamburg University of Technology, Germany*

nirmal.rathakrishnan@tuhh.de patrick.goettsch@tuhh.de

h.werner@tuhh.de

Abstract: In this paper, we present an experimental test bench to implement various cooperative control strategies for multi-agent systems, and illustrate its use with experimental results for a source-seeking problem, where a group of small wheeled robots termed as *Zooids* should locate a source of a given spatial scalar field. This algorithm is implemented as a validation to demonstrate the capabilities of the test bench. We propose to achieve this by utilising an internal target-based position controller, under the assumptions of convexity of the scalar, continuous/discrete field and availability of local measurements of the field, so that agents can calculate its gradient and its Hessian. We then show in experiments, that using estimated gradients and Hessians (with data communicated from neighbours) in the presence of noisy measurements of the field strength provides satisfactory results for convex fields, under various algorithms such as Steepest Descent, Gauss-Newton, Levenberg Marquardt. These algorithms are analysed, and experimental results are discussed.

Keywords: Multi-agent systems, Embedded robotics, Autonomous robotic systems, Networked robotic system modeling and control, Networked embedded control systems, Control under communication constraints, Control under computation constraints, Formation control, Source-seeking, Experimental-platform

1. INTRODUCTION

Recently, a number of various robotic test benches to realise swarm algorithms have been reported, such as Rezeck et al. (2017); Pickem et al. (2015); Rubenstein et al. (2012); McLurkin et al. (2013). The interest in such experimental setups has increased, to realise real-life environments in a test bench. This allows to develop and test algorithms that can be applied with little adaptation to real life problems. Consider e.g. a situation of oil leakage, recreation of such oil spill or toxic leak is quite challenging with the existing swarm setups. Our requirements for such a multi-agent platform are: Usability of a high number of tiny agents, recreation of a reproducible source field in a setup with non-ideal communication where position and source data have to be obtained on the robot instead of being communicated.

The GritsBot Pickem et al. (2015) developed for this purpose serves to be an appropriate solution, yet doesn't have the advantages of position based tracking system and needs additional sensors to sense external source field concentration, to implement source seeking algorithms. This gives us the motivation to chose the Zooids framework developed by Shape lab as a Swarm user interface to represent visual data Le Goc et al. (2016). This uses a projector based tracking system Lee et al. (2005), for the zooids to know their position coordinates. We propose to make software changes to adapt this platform to perform various control algorithms such as source seeking, formation control and obstacle avoidance. The Zooids hardware is unaltered in the adaptation process. We show that altering the projection sequence 2.3 leads to a higher data density 1 and allows it to add additional data packets with information such as noise corrupted source field data (Fig. 4) and obstacle field data. The added advantage is that with this specific projector based tracking system, we could send data in a robust and synchronised way to all the agents in the work-space.

We present an experimental platform for multi-agent control problems, such as source seeking, formation control, obstacle avoidance, all under hardware limitations in communication (like package dropout, bandwidth) and computational power. We propose and demonstrate improvements in the data projection scheme, achieved by software changes to the DLP-structured light projector, peer-to-peer communication and a debug channel via software changes in the Zooid firmware. We show experimental results on this platform by utilising the source seeking algorithm proposed in Datar et al. (2020). This paper presents and discusses results for steepest descent vs Gauss Newton; in addition we show results for Levenbverg-Marquardt vs steepest descent vs Gauss Newton algorithms, tested under different scenarios.

The rest of the paper is organised as follows. Section II presents the complete experimental setup, followed by a brief explanation of the hardware and software architecture. Section III briefly reviews the theory behind the source seeking algorithms. Section IV gives an analysis of different algorithms used, also is a validation for the capabilities of such a test bed. Section V gives the conclusion and possible extensions.

2. EXPERIMENTAL PLATFORM

2.1 Zooid Hardware

Each Zooid is 26 mm in diameter, 21 mm in height and 12 g in weight. The Zooids are driven by 2 DC micro motors,

10.1016/j.ifacol.2020.12.2640

non-colinearly placed but compensated by software to provide co-linearity. It can achieve a maximum speed of 74 cm/s. A 48 MHz ARM micro-controller manages the computation and wireless communication with the master computer and the other Zooids using a 2.4 GHz nRF24L01+ radio chip. The circuit board is integrated with 2 photo-diodes placed diagonally, as a part of the projector based tracking system. It is powered by a 100 mAh LiPo battery. Further details about the hardware can be found in Le Goc et al. (2016).

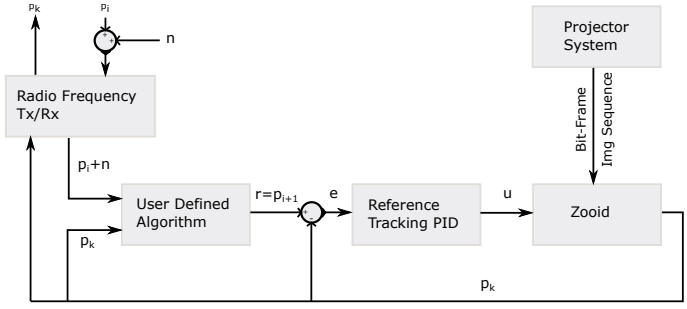

Fig. 1. Block diagram of the Zooid control loop.

2.2 System Overview

A block diagram of the proposed Zooid system is given in Fig. 1. The projection system displays a sequence of images on the workspace. These images are read by the Zooid using its photodiodes. The measurements are treated as a gray-code sequence and used to infer at least their respective location in the workspace, but based on the setup of the experiment, it can also contain further auxiliary data like, obstacle position or source field data, P_k. This location and auxiliary data is also broadcasted over RF to other Zooids and for debug purposes. A user defined algorithm like source-seeking (default for the rest of the paper), formation control, obstacle avoidance or any combination gets the current Zooid-data P_k and also the neighbouring Zooid datas P_i, which can be received over a RF channel. We also expect P_i to be corrupted with noise n during transmission over RF. One can infer that the projector system induces less noise in the control-loop. Both P_k and $P_i + n$ are fed into the user defined algorithm, which provides a new target P_{k+1}, that is tracked by the reference tracking controller.

2.3 Projector Based Tracing System

The projector hardware used is an ultra high frame rate (3000 Hz) DLP Structured Light Projector (LightCrafter), by Texas Instruments Inc. The projector based tracking system as in Lee et al. (2005) is used in this project. They proposed to transmit position information via the projector, the robots sense the data and calculate from the projected data sequence its actual position and orientation. The existing data transmission sequence comprises of a start bit, single data packet (20 bits) and a termination sequence (21 bits), illustrated in the Fig. 2.

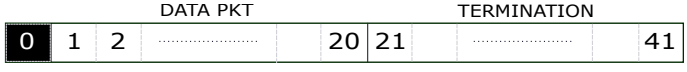

Fig. 2. Existing Data Sequence

This sequence is able to achieve a data density of 47.61%. Data density is the total amount of actual data carried in a projected data sequence, neglecting the termination sequence, start and stop bits.

$$\text{DataDensity} = \frac{n-(E+S)}{n}, \tag{1}$$

Where, n is the total number of bits, E is number of bits in the termination sequence and S is the number of start bits.

Every bit-frame image is projected for 333 μs, the time taken by the Zooid sensor to detect an high or low level. The total time taken for a message transmission is $13.65\,\text{ms} = 41\,\text{bits} \cdot 333\,\mu\text{s}$ equals an update frequency of 73 Hz, where only the termination sequence consumes $6.99\,ms = 21\,\text{bits} \cdot 333\,\mu\text{s}$. This is a serious drawback to be considered in the projection sequence, which costs the Zooid software an idle wait time of 6.99 ms in every data sequence. It also occupies additional bandwidth, which is in a setup with multiple robots an limited resource. Also adding more auxiliary data bits to the existing sequence is difficult as the number of bits in termination sequence increases proportional to the number of added auxiliary data bits, with the previously described consequences.

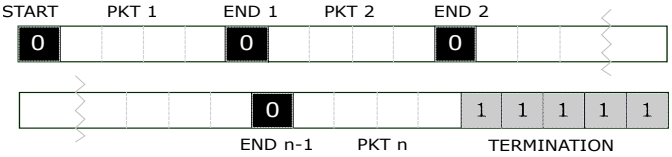

Fig. 3. Proposed Data Sequence

Proposed Projection Scheme A new data sequence is proposed which allows addition of auxiliary data bits as data packets. The data to be transmitted is first segmented as data packets of 4 bits each. Each packet is then packed with the sequence illustrated in the Fig. 3. Each data packet is prefixed with a 0-bit (Black), to distinguish them from the other packets. The termination sequence is 5 bits (1-White). The data sequence used in the experiments in section 4 is structured as shown in table 1. This gives the ability to add more data such as source and obstacle field concentration, by segmenting it to smaller data packets, thus increasing the data density without increasing the number of bits in termination seq. Also the risk of error will remain the same as the old data sequence. Increasing data density in-turn transmits more data within the same time frame of the old data scheme, thus maintaining 73 Hz refresh rate for one complete message.

Table 1. Data Packet Contents

Packets	No.Of Bits	Content
1 to 5	20	x, y coordinates
6	4	source information
7	MSB-1	parity bit

Source field concentration The designed continues source field concentration is discretized as scalar level curves, used in source seeking algorithms for finding local minima or maxima. This discrete field information is transmitted via projector as bit-frame images. Each bit frame image would comprises one bit data in the data sequence. One can also simulate various source field interpretations like static source field or (noisy) time varying source field (using projector's dynamic mode). This provides the motivation to use this method to replicate real life environment in the test bench and to be able to perform various experiments with the same reproducible source field.

Bit-Frame Construction The data to be transmitted via projector is segmented as bit-frame matrix of (1140x912) pixels and converted to earlier proposed data sequence in Fig. 3. This is accomplished by the procedures provided in Lee et al. (2005) and Kocev (2018).

Initially the information to be transmitted is created as data map, then scaled based on the size of the data packet. For instance consider the case of *scalar field*, 4 bits are used giving a resolution of 2^4 (0 to 15). After being scaled the image is converted to a matrix of size (1140x912 pixels), which is then decomposed to 4 binary bit-frame matrices. These bit-frame matrices are then converted to Gray codes. The final outcome from this conversion is a Gray code version of the bit-frame matrix. Final bit frame images of source data are shown in Fig. 4. One can use a closed form expression to represent continuous static source field as in eq. 2 or use the bit transmission procedure from the projector. The map in Fig. 4 represents a noisy corrupted source field, replicating a realistic source such as CO_2 or any toxic leak. These kind of source fields are hard to interpret as mathematical equation, hence the bit-frame projection method is preferred. When the Zooid is in coordinate (0,0) it would read gray code value for the source field potential as 0001 and when in centre of source field, it reads 1110.

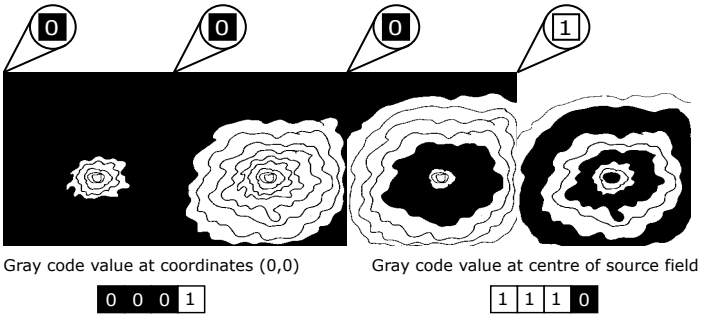

Fig. 4. Noise corrupted source field bit-frame construction

Comparison This new data sequence has proved to be a more efficient method with less wait time in the Zooid software compared to old data sequence. A projection at 3000 Hz, a period of 333μs for each bit frame hence 13.96 ms (42$\times$ 333 μs) for a complete message to be transmitted with a data density of 47.61%, is replaced with the new sequence (40$\times$333 μs), takes 13.32 ms for a complete message with a data density of 70%. The Zooid hardware is efficient enough to support a refresh rate of 1μs but the delay in task completion and data transmission requires to have a slower sample time (1/refresh rate). This scheme has a significant impact on transmission delays and the sample time of control algorithms that are implemented in Zooid Software. Table 2 gives a comparison of the data density between the old scheme and the new scheme.

Table 2. Comparison between old & new scheme

Scheme	No.of Start Bits	No.of Data Packets	Packet Size	Termination Seq Size	Data Density
Old	1	1	20	21	47.61%
New	7	7	4	5	70.00%

2.4 Zooid Software

The Zooid software is programmed in C and acts as a platform for implementation of control algorithms. In Le Goc et al. (2016) the software was designed for the Zooid to act as a slave for the commands being sent by the user via RF transceiver from a centralised command centre, to illustrate swarm robotics in a completely different scenario. This functionality is no more used since the source seeking algorithms we are developing and testing are decentralised algorithms and requires the Zooid to have its own intelligence. This requires to alter the existing software to a new version with various functionalities discussed in this chapter.

Scheduler The earlier version of the Zooid firmware consisted of a simple infinite loop, non-preemptive, task based scheduler. When it comes to time critical tasks such as computing direction vector for obstacle avoidance algorithms, this basic scheduler has a risk of failing to achieve the refresh rate of 73 Hz. Hence, we propose to have a preemptive, periodic Scheduler. The Zooid functions are subjected to tasks (blocks of code) and added as tasks to the scheduler linked list, along with specified period for the task to repeat. They repeat periodically until the task expires. Once the allotted time is expired for a specific task, next task is started unconditionally. The initial version was built with a infinite while loop, where the tasks do not give up the control if they have not completed their execution (no preemption). This causes a problem if a task takes longer than usual like in computing the hessian, as the data set varies every instance. The preceding tasks have to wait until the current task is complete. With the new scheduling algorithm, if the task has not completed within the requested time, then old estimation values are used and the scheduler preempts the current task with the preceding task.

2.5 Communication stack

The Zooid has a built in RF module (Nordic Semiconductors nRF24L01+) to communicate with other Zooids and the base command centre. We use the command centre platform as a debug channel to know the whereabouts of the Zooid and also to provide start and stop commands for the Zooids. We have also achieved peer to peer communication (Broadcast mechanism-UDP).

The Zooid software consists of a RF message transmission framework, that converts the message to be sent as 32 byte data with message type and Zooid ID. The maximum payload is 30 bytes. Every message sent is less than 30 bytes so it is composed in a single data packet. Zooids can transmit at a rate of 50 Hz (with acknowledgement), without affecting other functionalities, provided that the microprocessor used is single threaded. This has given the flexibility to provide a continuous feedback to the debug channel. We propose to transmit Zooid location and source field data that is read by the Zooid's sensors, from the projection system, as primary components of the feedback message. The rest of the bits could be used to debug other internal variables such as PID gains in order to tune the controller, etc. We have used the RF communication protocol with 2.4 GHz, data-rate of 2 Mbps and with address width of 3 bits.

Peer-to-Peer Communication The need for peer-to-peer communication arises when there is a requirement that every Zooid

must know the whereabouts of the other Zooids. The implementation procedure is similar to that of the one used in Paulsen (2018). The Zooid software can use the source data received from the projector, then it has to be transmitted along with the location information to the other agents, only the position data is transmitted and the source data is given by an internal lookup functions (2) and (3). This leaves us with 7 bits less than the other version and is used in the experiments. We have used the UDP protocol to transmit these messages, where there is no acknowledgement for the received data. We used this method since the packet loss is acceptable given the time taken for TCP/IP protocol is higher than the UDP as stated in Paulsen (2018).

Consequences of Mesh Communication The success rate of this broadcast mechanism comes with few constraints that increase the possibility of packet loss with a direct impact. Packet loss is a serious problem in systems with no special estimation technique to interpolate the lost packet. The factors that cause this effect are data rate (the rate at which the message is broadcasted), size of data packets, number of Zooids and, interference in the bandwidth. From Le Goc et al. (2016) we have to sacrifice the data rate if we have to send dense data packets or have more Zooids in the swarm.

2.6 Zooid Motion Control

The Zooid motion control consists of two coupled controllers, the angle and position controller PIDs. The target is provided to the control algorithms as pixel locations, both controllers are used to reach the target. Once the target is set, the angle controller aligns the Zooid angle to the target's angle, then proceeds forward with the position controller, while keeping alignment to target angle.

2.7 Experimental Platform Setup

The Zooid test bench platform consists of Zooid Hardware, DLP-LightCrafter by TI, Basler Camera, TV Screen, Physical stand, Debug Channel. The Basler camera is mounted perpendicular to the screen to optically track the Zooids location to validate motion control of the Zooids. The TV screen is used for visualising the source field as an image, since the projected bit-frame data of the source field is not visible to human eye due to the high projection frequency. All these devices are held in position by the physical stand. This portable physical stand has a compartment to hold a PC underneath, and also has border linings to prevent the Zooids from accidentally falling off the apparatus. The debug channel was implemented to read the RF messages and debug the internal variables of the algorithm since the Zooid hardware does not provide any other means of debug portal.

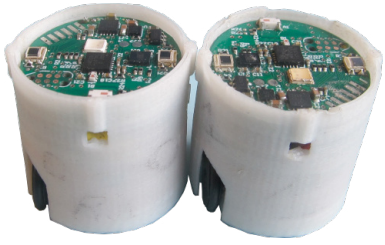

Fig. 5. Zooids used in experiments

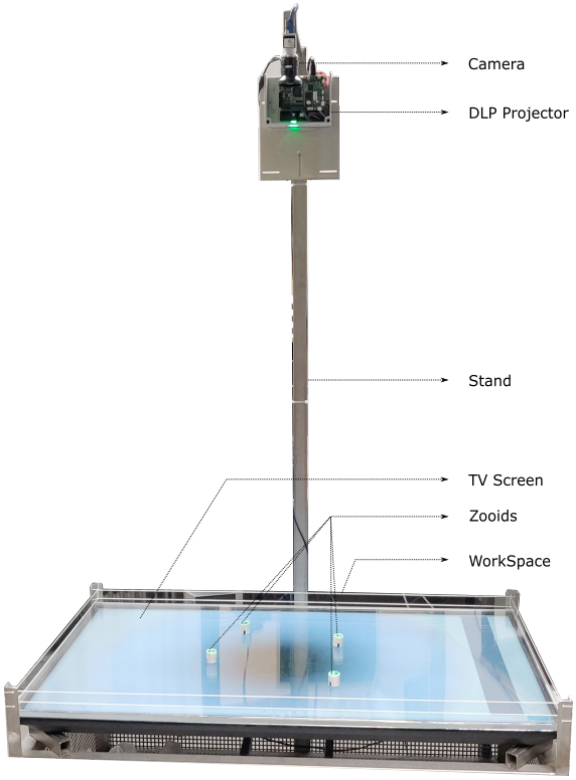

Fig. 6. Test Bench Front View

3. SOURCE-SEEKING

In this section we compare different source seeking strategies, and illustrate that Levenberg Marquardt achieves better convergence than Steepest Descent and Gauss Newton algorithms. The formulas used for computing these algorithms can be found in algorithm 1.

3.1 Notation

Let i denote the number of an agent, p_i the 2x1 position vector consisting of x and y components (x,y) and let (x_c,y_c) be the source field origin, $\psi(p_i)$ be the source field (scalar) corresponding to the position p_i, let η be zero mean, uncorrelated, white noise. $\mathcal{N}_i$ denotes the set of neighbours of agent i, g and H are the gradient and Hessian to be estimated using Eberly (1999). For a convex source field, α, β and ρ are positive constants, μ is a small positive number (e.g :$\mu = 0.01$).

3.2 Algorithm

The general procedure to compute the next reference point $r = p_{k+1}$ is summarised in Algorithm 1. The procedure to determine β is referred from Lourakis (2005), applicable only for Levenberg Marquardts algorithm. For Steepest Descent and Gauss Newton, β is used as a tuning knob. The computation of algorithm runs at maximum of 20 Hz, where the communication is capable to be implemented in 50 Hz. The received data are stored in a ring buffer used as a dictionary to store past values up to ten values p_i, including own data p_k. The dictionary data used to estimate the gradient and hessian using procedures in Eberly (1999) and new target is fed back to the reference tracking controller. For Gradient Descent we could also compute g with local measurement data p_k, by fitting a

plane, where it is also possible to calculate the same with neighbouring agent data p_i and own data p_k, by fitting a paraboloid.

Algorithm 1 Runs on each agent i at 20 Hz

Measure:
- i Measure p_i
- ii Measure $\psi(p_i) = \psi(p_i) + \eta$

Communicate:
- i Broadcast $(p_i, \psi(p_i))$
- ii Receive $(p_j, \psi(p_j))\ \forall j \in \mathcal{N}_i$

Estimate:
- i Update dictionary:
  $\forall j \in \mathcal{N}_i$ and $j = i$, **if** $||p_j - \hat{p}_j|| > d$, **then**:
  add $(p_j, \psi(p_j))$ to dictionary
  $\hat{p}_j = p_j$
  if$||Algorithm = SteepestDescent||$ **then**:
  Estimate [ii] gradient g only **else**:
  Estimate both [ii] gradient g and [iii] Hessian H
- ii Estimate the gradient g by fitting a plane with local measurements
- iii Estimate the Hessian H by fitting a paraboloid to the dictionary data.

Compute: Choose one of the three options below based on the experiment. Use β as tuning knob.
- i Steepest Descent: $p_{k+1} = p_k + \beta g$
- ii Gauss Newton: $p_{k+1} = p_k + \beta H^{-1} g$
- iii Levenberg Marquardt: $p_{k+1} = p_k + \beta g + (1-\beta) H^{-1} g)$
  if$||Algorithm = LevenbergMarquadts||$ **then**:
  Procedure to choose β:itemize
- Initially fix $\beta = 1$
- Compute p_{k+1}
- **if** $p_{k+1} \geq p_k$, **then**: $\beta = \rho \cdot \mu$, **else**: $\beta = \frac{\rho}{\mu}$

Pass: p_{k+1} to the target tracking controller
Repeat: Until source centre is reached

4. EXPERIMENTAL RESULTS

The experiments reported here use two Zooids in two different scenarios:

(1) The underlying source field is scalar, quadratic, elliptical ($K_1 \neq K_2$, eq. 2) and corrupted with noise by 5%
(2) The underlying source field is scalar, exponential, elliptical ($Q_2 \neq Q_3$, eq. 3) and not corrupted by noise.

Each scenario uses two different deployment positions [Type 1 :(x_1,y_1) = (750,50); (x_2,y_2) = (60,60), Type 2: (x_1,y_1) = (60,45); (x_2,y_2) = (100,40)]. These positions are represented in pixels. The workspace pixel range is (900x500). We used fixed starting positions for all the experiments. The Zooids send the status update over the debug channel, which gives the locations of each Zooid. The Zooid also receive the location of the nearby agents via the RF message and estimate Hessians and Gradients. The estimation of Hessians at times leads to nearly singular matrix, which is quite challenging in the Zooid environment to solve. This is due to the variation in the data set. Hence a reduction factor is multiplied to the data set, loosing the accuracy in data but ensuring the solution.

Quadratic Source Field

$$\psi(p_i) = K_1(x - x_c)^2 + K_2(y - y_c)^2 \tag{2}$$

Exponential Source Field

$$\psi(p_i) = Q_1 \exp\left(Q_2(x - x_c)^2 + Q_3(y - y_c)^2\right) \tag{3}$$

Error Plot To investigate convergence to the source, the error to the centre of each Zooid is calculated as:

$$v = \sum_{i=1}^{n} \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} \tag{4}$$

Where, K_1, K_2, Q_1, Q_2 and Q_3 are the parameters used to alter the source field dimensions and strength.

Error Plot Analysis : One can observe that close-by starting positions lead to longer convergence time for Hessian and Levenberg Marquardt methods. Whereas in a quadratic field corrupted with 5% uncorrelated, Gaussian, zero mean white noise one can see that Levenberg Marquadts provides better performance as the Gauss Newton in Fig. 8 and Steepest Descent. Looking into Fig. 10 one can see that Levenberg Marquardt provides a better performance than the other two algorithms in an exponential field, but when the agents are placed near each other, the estimation is poor.When the agents are placed far from each other and in opposite axis to the source field, a complete image (rich data set) of the source field is obtained.

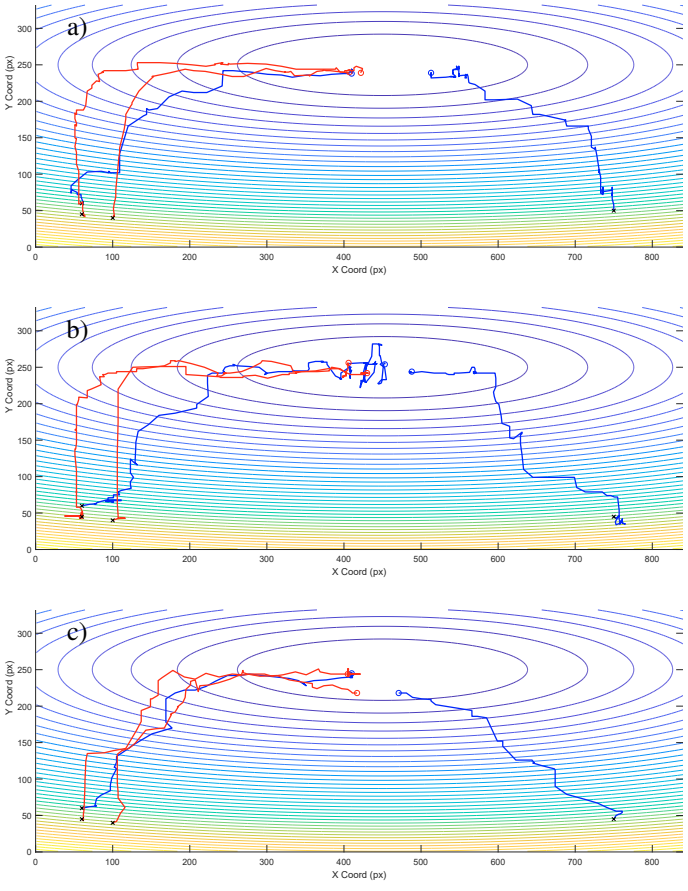

Fig. 7. Scenario 1: Elliptical noisy source field for both starting positions with a) Steepest Descent, b) Gauss Newton and c) Levenberg Marquadt

5. CONCLUSION AND OUTLOOK

An experimental platform for cooperative control scenarios for multi-agent systems has been established successfully, and has been tested with Zooids for source seeking applications under hardware limitations like limited computational power. A novel data projection scheme was implemented and turned out to improve performance. Source seeking experiments were

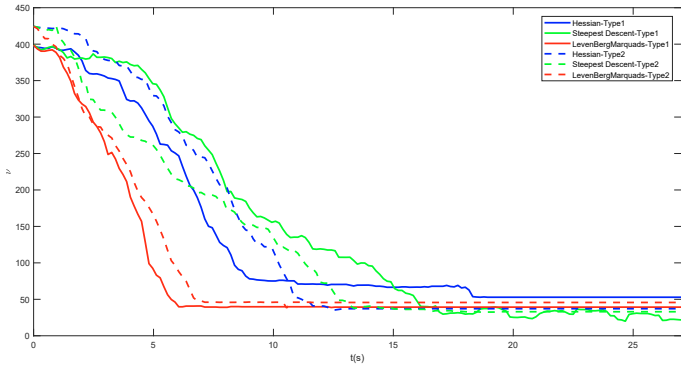

Fig. 8. Scenario 1: Error plot of all algorithms over time

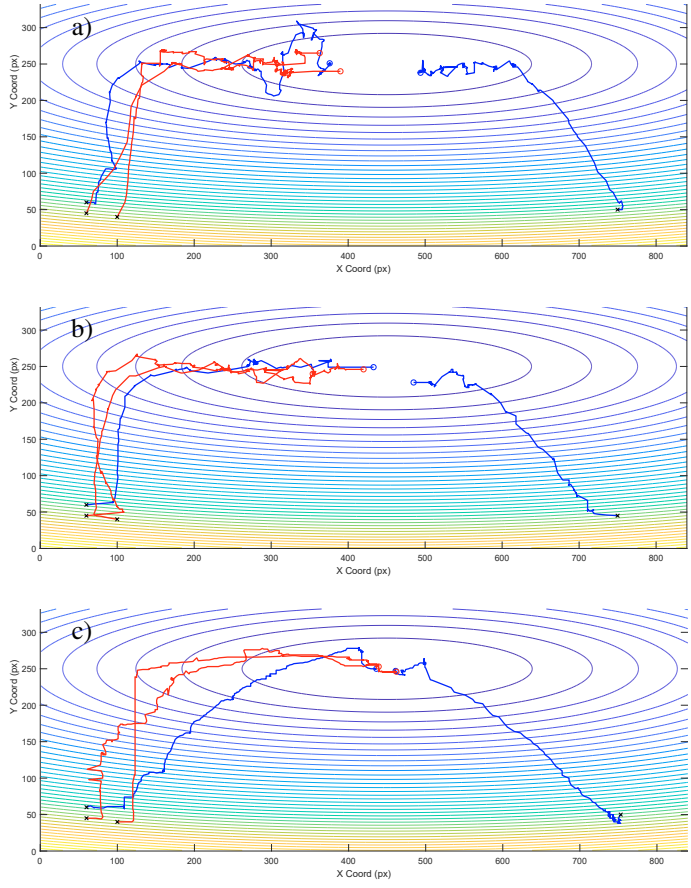

Fig. 9. Scenario 2: Exponential noiseless source field for both starting positions with a) Steepest Descent, b) Gauss Newton and c) Levenberg Marquadt

conducted as a first step towards implementing more complex and challenging multi-agent algorithms, and the results were discussed.

The Zooids hardware is a single threaded micro processor, with limited computational power, hence it restricts the implementation of complex algorithms. This provides the motivation to develop the next version of the hardware to have multiple core. With a powerful processor one can extend the source code with multiple algorithms to run efficiently on a board. More Zooids with updated hardware are currently being developed to test the algorithms with more agents in the swarm. Further developments are in progress to extend the area that is covered

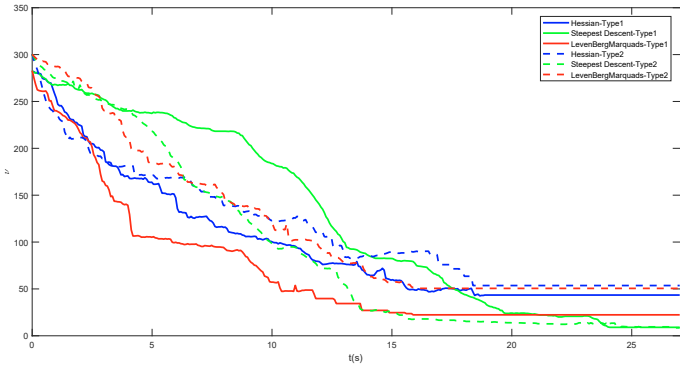

Fig. 10. Scenario 2: Error plot of all algorithms over time

by the projected positioning system. It is also planned to publish the code of this project as open source in future.

REFERENCES

Datar, A., Paulsen, H.P., and Werner, H. (2020). Flocking towards the source: Indoor experiments with quadrotors. In *European Control Conference*.

Eberly, D. (1999). Least squares fitting of data by linear or quadratic structures. *Geometric Tools*.

Kocev, K. (2018). *Development of a Mobile Platform for Swarm Robotics Experiments*. Ph.D. thesis, Technical University Hamburg.

Le Goc, M., Kim, L.H., Parsaei, A., Fekete, J.D., Dragicevic, P., and Follmer, S. (2016). Zooids: Building blocks for swarm user interfaces. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, 97–109. ACM.

Lee, J.C., Hudson, S.E., Summet, J.W., and Dietz, P.H. (2005). Moveable interactive projected displays using projector based tracking. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, 63–72. ACM.

Lourakis, M. (2005). A brief description of the levenberg-marquardt algorithm implemened by levmar. *A Brief Description of the Levenberg-Marquardt Algorithm Implemented by Levmar*, 4.

McLurkin, J., Lynch, A.J., Rixner, S., Barr, T.W., Chou, A., Foster, K., and Bilstein, S. (2013). A low-cost multi-robot system for research, teaching, and outreach. In *Distributed Autonomous Robotic Systems*, 597–609. Springer.

Paulsen, H.P. (2018). *Development of a software framework for implementation of Co-operative control algorithms on a swarm of small quadrotors*. Ph.D. thesis, Technical University Hamburg.

Pickem, D., Lee, M., and Egerstedt, M. (2015). The gritsbot in its natural habitat - a multi-robot testbed. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 4062–4067. doi:10.1109/ICRA.2015.7139767.

Rezeck, P.A., Azpurua, H., and Chaimowicz, L. (2017). Hero: An open platform for robotics research and education. In *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, 1–6. IEEE.

Rubenstein, M., Ahler, C., and Nagpal, R. (2012). Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*, 3293–3298. doi:10.1109/ICRA.2012.6224638.