Journal of Mechatronics and Robotics

Original Research Paper

A Vision-Based Approach to Real-Time Trust Evaluation within Multiagent Systems

[1]Alan Kruger and [2]Sun Yi

[1]Department of Computer and Electrical Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC, USA
[2]Department of Mechanical Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC, USA

Article history
Received: 20-07-2024
Revised: 17-08-2024
Accepted: 24-08-2024

Corresponding Author:
Alan Kruger
Department of Computer and Electrical Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC, USA
Email: agkruger@aggies.ncat.edu

Abstract: As robotic autonomy increases, single-agent systems are evolving into more complex multiagent systems that require more factors for successful implementation. In exchange for this complexity, applications such as autonomous delivery services and reconnaissance swarms become possible when they are infeasible with only one agent. By incorporating Trust into the multiagent system, the complexity of the agent-to-agent relationships can be managed more reliably by accounting for agent defectiveness or abnormality. This research explores ways of obtaining trust values for other agents using their own sensors and implementing them using a novel trust model called OmniLightTrust. The implementation is also deployed on a real multiagent system tasked with mapping an unknown region collaboratively with realistic computational limitations. Results show the relationship trust has with certain perception techniques, mission performance, and time delay. The perception pipeline was found to be the greatest bottleneck to this research as an inline dependency to each iteration of the trust algorithm. Operating in real-time and minimizing time delay are instrumental to getting acceptable performance from the trusted output. Consequently, this research focused a lot on how to solve this bottleneck by surveying the different perception technique pathways one can choose and how they synergize with the trust algorithm.

Keywords: Trust, Multiagent Systems, Robotics, Networked Robots, ROS, Communication

Introduction

A Multiagent System (MAS) has a lot of applications in the Internet of Things and robotics. In theory, Basheer *et al*. (2015) say a MAS can break down a complex task into a parallel set of simple tasks for multiple agents to collaboratively complete. Gautam and Mohan (2012) lay specific example applications ranging from package delivery to search-and-rescue robot teams. This study utilizes robotic-based agents such as UAVs and Unmanned Ground Vehicles (UGVs) for collaborative completion of tasks. Unfortunately, MASs are tough to implement for many reasons.

Natural and Contrived MAS Problems

Robotic agents are typically equipped with sensory, computing, power, and mechanical hardware. Due to wear and tear from prolonged usage, all these components can either malfunction or fail altogether, according to Kurniawati *et al*. (2011). Additionally, these components can suffer physical damage in the field, where they are at the mercy of the elements. Software bugs are another form of imperfections that can naturally be expected when developing robotic platforms. Unfortunately, there are also contrived threats that should be taken into account. Das and Islam (2012); Chen *et al*. (2018) state that adversaries wishing to simply disrupt or even benefit from the MAS may attempt a cyber-attack. Examples include agent modifications, trojan attacks, eavesdropping, or even Man-in-the-Middle (MITM) attacks, according to Liu *et al*. (2018); Arnaldy and Perdana (2019).

Trust

To see which agents are safe and worthwhile collaborating with, Chun and Guihan (2009) talk about MASs having begun to add trust models into their design. Trust values are obtained by observing agent interactions and performance using a trust evaluation model, according to Kurniawati *et al*. (2011); Rishwaraj *et al*. (2017). Chun and Guihan (2009) mentioned that the higher the trust value, the more the agent demonstrates ideal behaviors within the MAS. Agent interactions are traditionally judged

DOI: 10.3844/jmrsp.2025.1.12

based on data exchanges between agents. However, extracting useful information from data exchanges is not always practical. Users must consider the bandwidth and computational power needed to correctly interpret other agents in certain applications.

Proposed Alternative

This study introduces a way to make real-time trust evaluation possible using an MAS relying on lower memory and computing power devices. A practical trust-based MAS was achieved using low-bandwidth, live sensor-derived interpretations of other agents. The authors made this possible by creating OmniLightTrust (OLT), a novel trust model optimized for efficiently analyzing multiple parallel behaviors. The benefits and weaknesses of this approach will be expanded upon throughout the following sections.

Literature Review

Chae *et al*. (2016) created a color code to address getting a high detection range and flexible recognition angle in marker technology. The authors found that these markers can be placed on the robots as a means of detection and simultaneous unique identification. Increasing the number of agents became a limitation as it required more complex patterns to retain uniqueness. This made using the work of Chae *et al*. (2016) only viable in contained, short-range environments.

Wei *et al*. (2018) created a real-time sensor fusion algorithm that combines a monocular camera and LiDAR together to detect certain objects. The fusion consisted of an image-based Convolutional Neural Network (CNN) in parallel with a point cloud-based support vector machine. The greatest limitation of this research was the performance. The output detections on board the NVIDIA Jetson TX2 came in at 5 Hz.

In RoboTrust by Mikulski *et al*. (2012), the output of its observation, or acceptance function, is binary for simplicity. A favorable observation is a 1, and an unfavorable observation is a 0. Mikulski *et al*. (2012) also noted each agent considers the current observation and a constant number of old observations tuned by the practitioner. The maximum likelihood that the event is favorable is calculated for each observation in an agent's memory. The minimum of the maximum likelihoods within the set of observations is the output trust. This study was most limited by the use of one single behavior to observe.

Rishwaraj *et al*. (2017) attempted to create a more efficient trust estimation model using objective completion. The binary agreement of detected targets dictates favorability. Their proposal uses temporal difference learning, which takes the observation value, the previous trust value, and a tunable learning rate to determine Trust. Agents who get too low of a reputation in the network are identified as useless agents. Unfortunately, this research is very theoretical and untested outside of simulation.

Zhou *et al*. (2015) created a stereotypical deep model called Context-Aware Stereotypical Trust (CAST). Liu *et al*. (2009) point out that stereotypes are said to be behaviors correlated with certain Trust. CAST from Zhou *et al*. (2015) observes agent quality of service and extracts up to 7 stereotypes pertaining to the application. Additionally, CAST evaluates the Trust of the agent once trained. While this was very robust, Rishwaraj *et al*. (2017) critiqued that any behavior outside of the seven stereotypes would not be covered.

Das and Islam (2012) focused much more on the cybersecurity aspect of Trust by using secured trust to detect malicious oscillatory behaviors. Some of the tools Rishwaraj *et al*. (2017) used to achieve this include satisfaction, similarity, and trust trends. Overall, Trust is calculated using the expected Trust and a decay model that states Trust decreases as agents become absent over time, according to Das and Islam (2012). Not only did this study include a trust model, but it also included a load-balancing scheme to avoid having a single agent doing all the work. Rishwaraj *et al*. (2017) argue that this study is very powerful, but it is also taxing on the system to compute so many complex parameters. It also is limited in terms of practical performance, which is something the authors intend to improve in the future.

Cheng *et al*. (2021) found that a MAS with a trust model could be used to improve safety in intersections and cooperative adaptive cruise control scenarios. Resistance to adversarial agents was another compelling aspect of this research. The work was specialized for its automotive application. More generalized work can be done for various types of agents in MAS, considering their different dynamics and characteristics.

The use of a Bayesian model to predict trust values of given agents in a multiagent system is very effective, as presented by Hallyburton and Pajic (2024). However, the approach relies on multiple sources of agent data, which may require substantial time to converge to something meaningful for actual implementation.

Materials

This research can be replicated using any edge device, camera, LiDAR, and agent robot model. All hardware used in this research was chosen because of availability and specifications. The niche components used in this experiment include AR codes and retroreflective tape. AR codes can be freely printed and retroreflective tape can be found in arts and crafts stores. Any open-source software used in this experiment can also be used freely. However, the state machine software, trust algorithm, and models developed in this research will need to be redeveloped by anyone trying to replicate this. Derivatives may cause worse or improved results.

DOI: 10.3844/jmrsp.2025.1.12

Methods

This study does not focus purely on either Trust or image processing techniques. The focus of this research is how these two topics can be combined to ultimately create a solution that is beneficial and worth exploring further. As a result of this novelty, the literature provides insights from either subject but rarely a combination. Shakshuki and Reid (2015) warn that the major clash between these two topics is the detection of robotic agents in a homogeneous system and how they can maintain a memory of Trust.

Case Study

A single observation method has its own strengths and weaknesses. A case study exploring and comparing multiple observation methods inspired by the literature is presented to obtain a broader perspective. The three cases are designated as AR tracker, sensor fuser, and CNN Detector, respectively, for simplicity. These methods aim to collect and calculate behavioral data such as attitude and velocity for example. This fundamental data can then be used to interpret more complex behaviors about the agent's performance.

In AR tracker, agents are equipped with a stereo camera and a unique AR data matrix code representing a value between 1 and 4. The initial advantage of this method is the simple gathering of relative position, velocity, and Euler angles using the AR track alvar package. This package from Ros Wiki Org (2016) additionally supports up to 24 different agents, which far exceeds the numbers of the other cases. Simply printing out an AR code, attaching it to an agent, and running the AR track alvar package describes all the complexity.

In Sensor Fuser, the agents are equipped with a camera, a LiDAR, a headlight, and color-coded retroreflective tape. This case supports only four agents, each representing a dominant color of retroreflective tape. Agents 1 through 4 are designated to be red, green, blue, and yellow, respectively. Adding LiDAR not only allows for detection in most lighting conditions but also finds reflective surfaces as outliers, according to Hata and Wolf (2014).

The robots receive visual information using Robotic Operating System (ROS) middleware. Camera data is processed using OpenCV libraries. LiDAR data is processed using Point Cloud Library. For more information, see Quigley *et al*. (2009); Bradski (2000); Rusu and Cousins (2011), respectively. Once there is a consensus between the LiDAR and the camera, the agent is recognized as the correct color ID and is localized.

In a CNN detector, agents are simply equipped with a camera. COCO SSD MobileNetV2 from Huang *et al*. (2017) is used to train a homemade data set of robot images found in the MAS to detect them in real time. In theory, the biggest advantage of using this method is the scalability it has with proper training. This can possibly achieve great results without needing to pile on more sensors or bulky modifications on the agents in some cases.

Trust Evaluation Model

Without an implementable perception-based trust model in the literature, OmniLightTrust (OLT) was born. Most of the algorithms in the literature and OLT share similarities. All of these models need an acceptance function and ultimately output a trust value between 0 and 1. Table (1) and the pseudocode in Fig. (2) present the lower-level functionality.

Table 1: Useful terminology and mathematical rules for understanding OLT

Variable/Term	Meaning	Condition
B	# of behaviors	$B \geq 1$; $B = P + N$;
P	# of positive behaviors	$P \geq 0$
N	# of negative behaviors	$N \geq 0$
Observation data	Perceivable agent states	Measurable in sensor's range
Behavior	Coupled states to trust	Feasible computation
Neutral Trust (NT)	The default trust value if there is no reward or punishing reinforcement	If $N == 0$, $NT = 0$; If $P == 0$, $NT = 1$; Else, $0 < NT < 1$;
Acceptance Function (AF)	Current behavior is favorable or unfavorable	If $Obs == bad$, $AF(Obs) = 0$; Else, $AF(Obs) = 1$;
Favorability Threshold (FT)	Maps favorability to appropriate delta value	If $AF(Obs) == 0$, ΔT = Decrement; Else, ΔT = Increment;
Increment/Decrement (ΔT)	The small value that adjusts trust depending on the output of acceptance function	Increment >0; Decrement <0; Increment < |Decrement|;
Optimistic Limit (OL)	The ceiling value of a behavior	$OL > PL$; $-1 < OL < 1$;
Pessimistic Limit (PL)	The floor value of a behavior	$PL < OL$; $-1 < PL < 1$;
Positive Weights (W_{+i})	The weight of each positive behavior	$W_{+i} = OL_{+i} - PL_{+i}$ for $i = 0, P-1$.
Negative Weight (W_{-})	The weight of all negative behaviors	$W_{-} = OL_{-} - PL_{-}$

DOI: 10.3844/jmrsp.2025.1.12

Table 1: Continue

Behavior Extremes	Limits weight W within the bounds of OL and PL to bind overall trust value	$\sum_{i=0}^{P-1} W_{+i} + W_{-} = 1$ $NT + \sum_{i=0}^{P-1} OL_{+i} + OL_{-} = 1$ $NT + \sum_{i=0}^{P-1} PL_{+i} + PL_{-} = 0$
Behavior Trust (BT_i)	The current value of each behavior	$BT_i + = \Delta T$ for $i = 0,\ldots B-1$. $PL_i \leq BT_i \leq OL_i$;
Reward Value (RV)	The sum of all positive behavior trust values	$RV = \sum_{i=0}^{P-1} BT_{+i}$
Punishment Value (PV)	Min of all negative behavior trust values	$PV = MIN\ (BT_{-i})$ for $i = 0,\ldots N-1$.
Overall, Trust (OT)	Quantification of productivity of an agent	$OT = RV + PV + NT$
Collaboration Threshold (ClT)	Collaboration begins when the agents trust or mutual Trust surpasses this value	If OT > ClT, collaborate; Else, do not collaborate; ClT > CnT
Connection Threshold (CNT)	Agents below this level of Trust are too risky to keep in the network and removed	If OT < CnT, disconnect; Else, do not disconnect; CnT < ClT
Performance Trust	Observes higher level tasks	Optionally independent from OT

Table 2: Comparison of OLT proposed in this study to other methods

Study	Communication data	Perception data	Mission objectives	Simple	Multiple stereotypes	Experiment
Mikulski *et al*. (2012)	Y	N	N	Y	Possible	Simulation
Rishwaraj *et al*. (2017)	Y	N	Y	Y	N	Simulation
Zhou *et al*. (2015)	Y	N	N	-	Y	Simulation
Das and Islam (2012)	Y	N	N	N	N	Simulation
OLT	N	Y	-	Y	Y	Implemented

Looking at the visual representation of OLT in Fig. (1), the practitioner has the freedom to pick many visual behaviors to observe. Behaviors are defined by quantifiable parameters that the sensors can extract, such as roll and position. It is worth noting that OLT can work with nonvisual inputs, too, so it is universal. These behaviors each have their own feedback loop best shown in Fig. (1). The current trust value of that specific behavior is increased or decreased with positive or negative observations, respectively. The weight system is unconventional because it is not a gain-based system. Instead, it uses neutral Trust as a starting point and allows each behavior to increase or decrease it appropriately. Each behavior has a certain range of improvement or detriment to trust depending on its worth to the overall Trust of an agent. This is just one iteration; Fig. (2) shows the time component better as this is just a single frame of data collection. Trust approaching 0 signifies a danger to the MAS. Trust approaching 1 signifies productivity and safety in the MAS. Akintunde *et al*. (2024) define the Trust that is gathered from past events as retrospective Trust in the literature or typical Trust. However, Akintunde *et al*. (2024) define the predicted Trust to be prospective Trust or actual Trust of an agent. Simply using gain coefficients while limiting overall Trust to 0 and 1 would lead to less critical, yet more common, behaviors skewing Trust. This new weight and management system for agent stereotypes, as well as the use of visual observations as the input, contribute to the novelty of this study. Table (2) shows the most relevant studies and how they compare to OLT in terms of novelty.

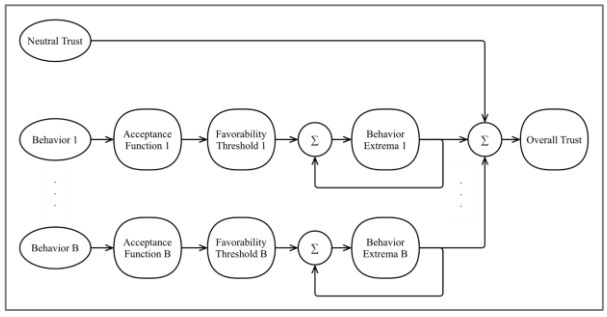

Fig. 1: Low-level symbolic view of the OLT system

Implementation

The MAS consists of two Robotis TurtleBot3 Waffle agents, each using an NVIDIA Jetson TX2 as the on-board edge device. Each TX2 uses the middleware ROS Kinetic, running on Ubuntu 16.04 LTS. Those lacking the hardware needed to reproduce this study with the same code can utilize Gazebo as Dekkata *et al*. (2023) did, which is a physics simulator that integrates seamlessly with ROS. The experiment is done in a collaborative manner without the aid of GPS through network sharing of x and y boundary coordinates. Each agent makes its own decentralized map using the method in Wiki. Ros. Org. (2016) with respect to its own origin, regardless of any collaboration. An open-source SLAM algorithm is provided in Wiki. Ros. Org. (2016).

DOI: 10.3844/jmrsp.2025.1.12

```
CONSTANT B, NT, OL [1: B], PL [1: B], increment [1: B], decrement [1: B];
BT [1: B] ← 0;
OT ← 0;
while collecting data, do
    OT ← NT;
    for B behaviors, do
        observation ← observationCallback ();
        favorability ← accFunction(observation);
        if favorability is True, then
            if BT_i + increment_i ≤ OL_i then
                Increment BT_i;
            else
                BT_i ← OL_i;
            end if
        else
            if BT_i + decrement_i ≥ PL_i then
                Decrement BT_i;
            else
                BT_i ← PL_i;
            end if
        end if
        OT ← OT + BT_i
    end for
    return OT;
end while
```

Fig. 2: OmniLightTrust (OLT) pseudocode

Any robot can be used in this case as long as it is suitable for the application and given the resources. Omnidirectional ground robots were selected due to their higher payloads for holding a 3D LiDAR. Those wanting to implement this study on a MAS of autonomous UAVs can opt for lighter sensors than this study showcased. To consider the number of robots allowed in the system, the user needs to consider the system's bandwidth. In this study, there is a maximum number of robots that can be processed in a single frame before time delays are introduced in Eq. (1):

$$t_{total} = n\,(t_{\text{agent } processing}) + t_{detection\ system} \quad (1)$$

where, n is the number of robots in the system, $t_{agent\ processing}$ denotes the period of time each positive detection takes, and $t_{detection\ system}$ is the time it takes for the system to refresh. This can be used to determine the total amount of delay, t_{total}, the system can afford per frame with n agents when converted to seconds. Unlike what Toda and Kubota (2013) performed, this study is decentralized, so there is no master agent or map. Toda and Kubota (2013) focused on the development of a multiresolution map. This study used an existing SLAM algorithm and focused on how agents trust each other while creating these maps. They send each other vectors of 3D points representing map data as well as Boolean indicators of which agents have been removed. These transmissions take the form of TCP messages within ROS while they keep their trust values private.

This example MAS implementation makes use of 4 user-defined behaviors to demonstrate its effectiveness. Presence and stability denote positive behaviors. Crash and oscillation denote the negative behaviors. A high-level representation of the trust algorithm in this implementation can be seen in Fig. (3). Presence judges an agent based on its status of being in range of the user at certain time intervals following the decay model of Das and Islam (2012). Presence is of lower priority in this application because the agents seldom separate for long periods, which merited it a weight of 0.1. Stability considers the roll or pitch of the agent to judge whether the agent is properly upright. The tuning of the weights was chosen to be pessimistic for this example, so stability gets a weight of 0.3. This allows the punishing behaviors to share a majority weight of 0.6. Crash considers velocity as well as roll or pitch to judge whether the agent is stuck and unable to move. Oscillation considers velocity and relative position to observe whether the agent is moving in a malicious manner. Figure (4) shows how the behaviors and weights balance out visually.

As each agent encounters other agents, it takes time to allow trust values to adjust with respect to the behavior of the new agent. If Trust surpasses the collaboration threshold shown in Fig. (4), the trustworthy agent shares its live data with the observer until that threshold is no longer met.

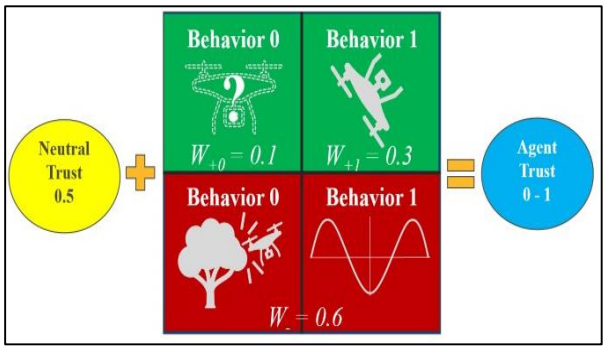

Fig. 3: High-level representation of experimental behaviors

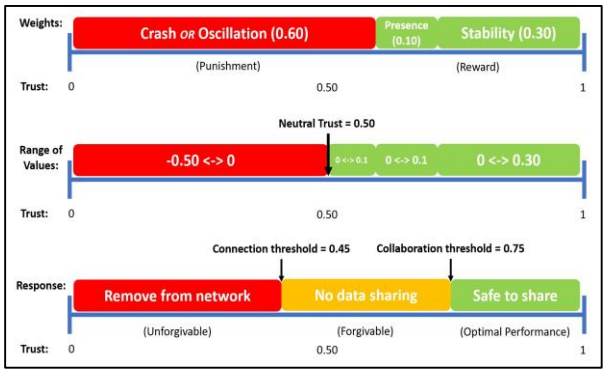

Fig. 4: MAS response to trust

DOI: 10.3844/jmrsp.2025.1.12

The agents can more efficiently create decentralized maps by collaborating with agents that behave as the MAS was designed. Any agent that dips below the connection threshold is disconnected from the entire network. Not only does that agent lose its mapping capabilities, but all agents become aware and avoid collaboration with the abnormal agent. Two other consequences for dipping below the connection threshold were considered. The first consequence added the absolute shutdown of the agent to avoid continued physical movement. The second consequence could involve other agents ignoring the detection of the disconnected agent to reduce unnecessary computations. Considering that the agent most likely displayed suspicious behavior according to its tuning, it will not be forgivable and can theoretically be replaced by a new agent throughout the mission. Some cybersecurity attacks can really abuse these consequences, but that will be discussed in the Discussion. The consequences depend on what the MAS designer values and views as worth the added complexity.

The first experiment to test the vision-based approach and OLT is the collaborative and autonomous mapping of a room. A manually controlled drone is used as a leader agent but does not have an on-board system to evaluate Trust. The leader, agent 4, is autonomously followed by agent 3. Lastly, agent two autonomously patrols the perimeter and is tasked with ensuring that agent 3 is following agent 4 as a subtask. While the agents navigate throughout their mission, agent three will eventually get cyber-attacked after a trigger and imitate defectivity in its control system. As it is behaving abnormally, agent three will propagate erroneous SLAM data to the other agents. This demonstration will be used as a way to show the effectiveness of incorporating a trust model into the system. The rainbow-colored dots in Fig. (5) denote live laser data, while the white dots denote accumulated SLAM data using the open-source software package from Wiki. Ros. Org. (2016).

The second experiment tackles the idea of excessive computational delay being introduced to the first experiment and how it affects Trust. A lot of sensor-based agent detection algorithms are not able to perform well in real-time, so this is an important factor to explore in the field of Trust. To find out, the same experiment is played back, but the trust evaluation model gets an input throttled to 1 Hz instead of the expected 15 Hz.

The significance of these experiments revolves around discovering the worth of Trust. The other trust models from the literature were not very practical in a real-life mission. Most were just too computationally expensive and would consequently throttle primary functions. Other approaches were just not robust for real-time applications. If the incorporation of a trust model is simply going to create a bottleneck and degrade the performance of the system, then the trust model should be omitted. This study demonstrates a system that needs real-time decision-making and discovers an approach to trust evaluation that improves the MAS.

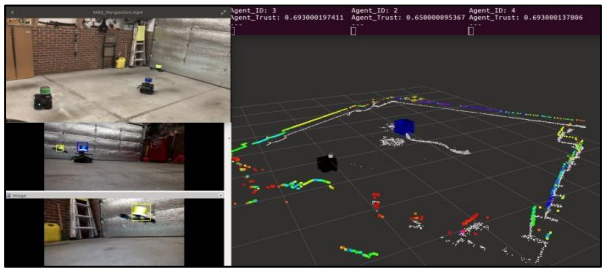

Fig. 5: Visualization of the SLAM MAS in action

Results

Case Study

A MAS benefits from a simplistic design with minimal hardware. Fewer components lead to fewer points of error. Some MASs can use hundreds of agents, and each component needs to be multiplied by the number of agents, which can skyrocket the budget. Another consideration is the size and mass of the equipped hardware. A more robust robot will be needed for heavier equipment, which will increase cost. Specifically, of interest to the vision-based approach, the effective range, Field-of-View (FOV), and resolution of sensors are crucial to gathering accurate and reliable information about other agents. As the range and FOV decrease, the time agents spend without any other agent witnessing their behavior increases. As sensor resolution increases, the agent will need more computational power to process the data in real time. Table (3) shows how this case study stacks up in regard to hardware. All color-coded tables will imply that green is advantageous, yellow is neutral, and red is disadvantageous.

The CPU and RAM usage of each method can impact the kind of embedded processor used onboard each agent. The results were all taken using the TX2 for reference. See Hardware for Every Situation (2024) for more information. Other devices considered for this study are shown in Table (4). The TX2 was the most expensive, most powerful, and largest option. This limited the robots available to carry it, but the capability allowed for the more flexible exploration for this case study. Despite the strengths and weaknesses of each method in the case study of visual behavior capture, each case affects Trust in a different way.

The results of the case study are shown in Table (4-5). All three cases contribute a major advantage, but there are some major deficiencies found in each one. It is worth noting that cheaper or more powerful hardware can be interchanged or added to rectify some of each case's flaws. This depends on the practitioner's performance, budget, and convenience.

AR Tracker provided the best number of supported agents, diverse parameter detection, and a simple setup. AR Tracker used C++ and a Stereo Labs ZED 3D camera for rectified image support. The lighting needed to be optimal, and any movements captured by the camera or made by the agent needed to be minimal.

DOI: 10.3844/jmrsp.2025.1.12

Table 3: Comparison of hardware used in this study

Description	AR tracker	Sensor fuser	CNN detector
External sensors	Stereo labs ZED 3D camera	CSI camera, Velodyne LiDAR puck	CSI camera
Additional price	$450	$4,000	$0
Total mass (g)	135	830	-
Size (mm)	175×30×33	103.3×103.3×71.7	-
FOV (°)	90 (H) × 60 (V) × 100 (D)	360 (H) × 48.8 (V) combined	62.2 (H) × 48.8 (V)
Sensor resolution	1920×1080 p	960×540 p, H: 0.1◦–0.4◦, V: 2◦	480×270 p
Bandwidth (MB/s)	~93.3	~29.3 combined	~5.8

Table 4: Comparison of edge computing hardware

Edge device price	CPU CLK freq	RAM	GPU support	Mass	Size (mm)
NVIDIA Jetson TX2 $400	4-core: 2 GHz	8 GB	CUDA	467 g	170×170×49.8
NVIDIA Jetson Nano $100	4-core: 1.4 GHz	4 GB	CUDA	250 g	100×80×29
Raspberry Pi 4 $75	4-core: 1.5 GHz	8 GB	OpenCL	46 g	88×58×19.5

Table 5: Vision-based observation methods

Description	AR Tracker	Sensor Fuser	CNN Detector
Agent distinguishability	AR codes (24 agents)	Color coding (4 agents)	Feature extraction (? agents)
Effective range	~1.5 m	~5 m	~4.5 m
Visualizable data	XYZ coordinates (m), Euler angles (rad)	XYZ coordinates (m), euler angles (rad)	Relative pixel coordinates
Conditions	Great lighting only	All lighting, overexposure	Great lighting only
Sensor delay	0.062 s	0.013 s combined	0.010 s
Preprocessing delays	0.133 s	0.017 s	-
Agent detection delay	N (0.001 s)	N (0.016 s)	N (0.120 s)
Agent update frequency	~7.5 Hz	~15 Hz	~13 Hz
Trust calculation	0.0000045 s	0.0000045 s	0.0000045 s
Total delay (s)	0.195+0.001 n	0.030+0.016 n	0.010+0.120 n
CPU usage (with viz)	~95% (1 core)	~35% (1 core)	~60% (4 cores)
RAM usage (with viz)	0.7 GB (33%)	0.3 GB (25%)	4.2 GB (83%)

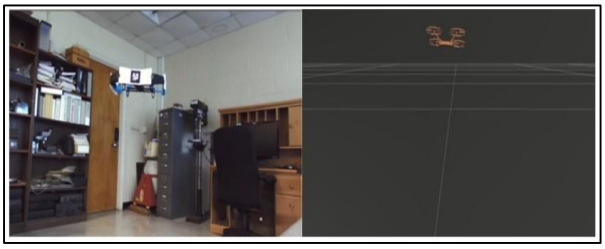

Fig. 6: AR Tracker interpreting flight behavior of quadcopter

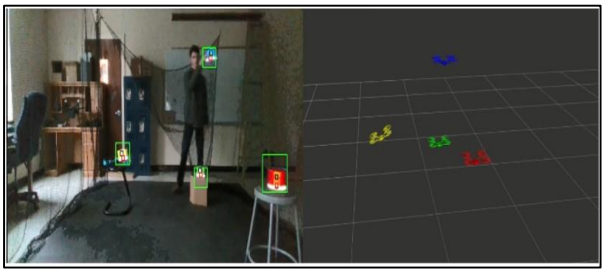

Fig. 7: Sensor fuser analyzing MAS in a dark environment

The same issues in Chae *et al*. (2016) regarding range, size, and lighting applied here. Notably, being outside the 1.5 m effective range not only resulted in missed detections but also incorrect detections. In addition, the paper AR codes are 2D and fragile. The side of the agent with the AR code must face the other agent's cameras to be detected, as shown in Fig. (6). As a result of these drawbacks, gathering reliable information about an agent is highly compromised. Consequences include the facilitation of hiding malicious behaviors within the same vicinity and false assignation of trust updates. The main takeaways from this method are to avoid 2D identifiers and focus on maximizing detection accuracy.

Sensor fuser boasted the best effective range, the most robustness to external conditions, and the lowest computational expense. The Velodyne LiDAR puck introduces 3D coordinate data with a root mean square error of 3 cm. For more information, see Velodyne LiDAR PUCK™ (2015). This level of accuracy is adequate for measuring certain agent parameters reliably, such as velocity and relative position, as shown in Fig. (7). The LiDAR also contributed to the increase in effective

DOI: 10.3844/jmrsp.2025.1.12

range and allowed the camera to use a lower resolution for higher efficiency. The 3-2D conversion is done in C++ and makes up the backbone of the accuracy of Sensor Fuser. The experimental and expected signals had an autocorrelation of 0.9986 and 0.9945 for x and y, respectively. The cylindrical tags are lightweight enough for a small drone to carry and allow for 360° yaw coverage in both light and darkness. Decreasing the diameter of the cylinder decreases the detection range and would be less burdensome, than the AR tracker discussed.

The setup is not only difficult but also expensive and heavy. The number of agents that can be supported is the other drawback that limits its viability to only small MASs. However, Sensor Fuser is still able to better detect agents with more range, sampling frequency, and realistic lighting expectations. Trust values, therefore, have the flexibility to better represent the Trust of a wider variety of applications and requirements. The main takeaway from this method is to keep in mind the kind of MAS desired before committing.

CNN Detector best-showcased simplicity and good effective range in exchange for the high computational load, as shown in Fig. (8). This is the only case using Python and a good GPU. Thanks to the NVIDIA Pascal GPU, the time delay is still very acceptable. Additionally, the low resolution of the camera and lack of external sensors also contributed to the achieved update frequency. Without an adequate update frequency, applications that depend on precise behaviors will not be happy. Additionally, this case lacks the capabilities that the other two demonstrated, but that can be added. CNN Detector does not escape the need for modification to discriminate agents. The data set required to train an agent to calculate information such as Euler angles while being able to discriminate robots would be a huge challenge. Combining this challenge with the extra resolution and needed depth data results in an embedded system that may not achieve a real-time frequency. Without enough measurable parameters, the agent's trust value will not optimally represent an agent. The main takeaway from this method is to ensure that the parameters of interest are simple enough to train for high accuracy. Additionally, users would need to be prepared to acquire an appropriate data set.

This case study is meant to provide insight into the challenges of acquiring robust observation data using on-board real-time computing for Trust. For those using big MASs in a lab setting with little effort, AR technology is great. Those wanting more robust detection should consider fusing a camera with compatible sensors. Lastly, those wanting to get the most data with the least sensors possible should consider using machine learning approaches.

Trust Evaluation Model

OLT was written in C++ and takes 4.5 µs to update Trust per observation. This shows that OLT itself is almost negligible in comparison to the observation methods in terms of computation time. For the first 58 s of the experiment, the agents behave in a predictable manner, and trust increases as expected. Additionally, performance trust values fluctuate in parallel where agent 3 is assigned to follow and maintain a gap of 0.8 m away from agent 4. Once the defectiveness begins at 58.07 s in Fig. (9), it takes about 2.93 s to begin lowering Trust. It also took 5.93 seconds to lower it below the collaboration threshold to stop the flow of fraudulent data. Compared to the rise of Trust at the beginning of the experiment, the gradient of Trust is purposefully much higher while decreasing. This is because Mikulski *et al*. (2012) advised Trust to decrease faster than it increases. With a higher value assigned to the collaboration threshold, the sooner bad behavior can be blocked. The trade-off is that a high collaboration threshold also makes it more unlikely that collaboration will happen. Agent 3 also became disconnected from the network after about 13.53 seconds since beginning to show oscillatory behavior.

Although agent 3 was the culprit of abnormalities, Fig. (10) showed that it still trusted the drone and patrolling UGV while misbehaving. This was situational, and a defective agent could have mutually decreased Trust. Agent 3 took much longer to trust agent 2, and that is due to the vision-based approach. Considering that agent 3 was busy chasing agent 4, it was not encountering agent 2 as often as agent 2's FOV contained agent 3. This proves the importance of effective range and FOV in the observation methods.

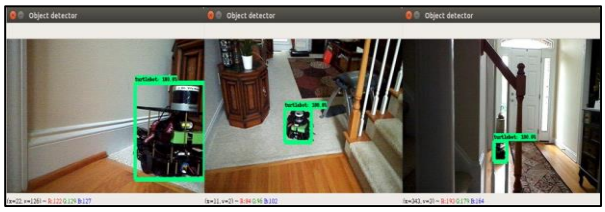

Fig. 8: CNN detector robustly detecting TurtleBot3

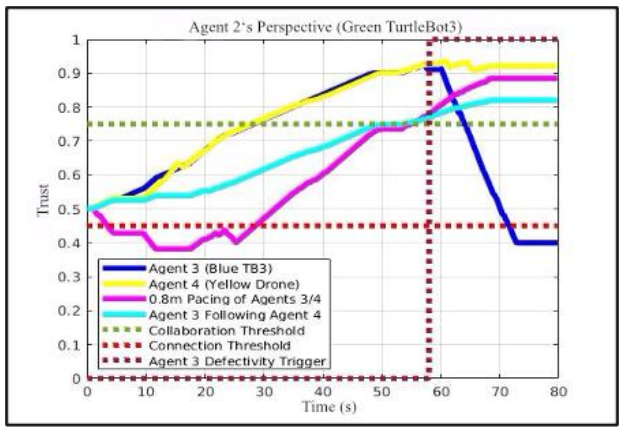

Fig. 9: Trust evaluations agent 2 gathered using OLT

DOI: 10.3844/jmrsp.2025.1.12

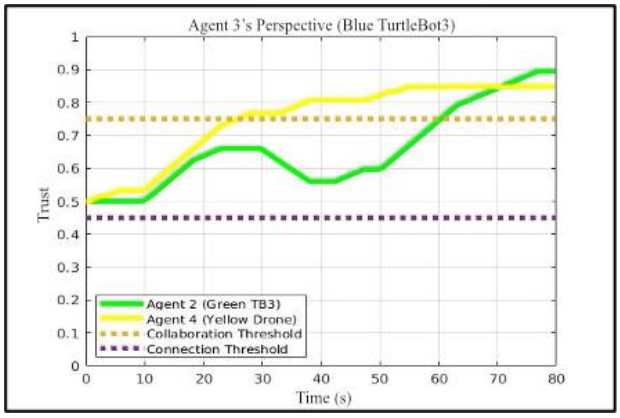

Fig. 10: Trust evaluations agent 3 gathered using OLT

After triggering the defectivity in agent 3, the SLAM became contaminated with poisonous and fraudulent data. However, it stopped affecting the network when agent 3's Trust was less than the collaboration threshold. Figure (11a) output contained 5.93 s of exposure to erroneous data for reference. Figure (11b) shows the same agent's SLAM output without using a trust model to attenuate the fraudulent data. It is essentially Fig. (11a) with about 22 s of exposure to faulty boundary scans instead of 5.93 s.

In a SLAM mission, corrupt data sent around the network can have infinite possibilities. Agents that use this data to directly navigate in real-time and do not question this data will be at risk of damaging themselves or others.

By paying upfront for the price of implementing a trust algorithm, agents become resistant and selective towards shared data that should be ignored. That is why the trust-based result in Fig. (11a) appears cleaner than in Fig. (11b). Another point about this comparison is the intricate differences between Fig. (11a and c). Consider an alternative where Trust is calculated by sending each other the map data each agent collects. Comparing and verifying the dense information is not only difficult, but one can see the two agents' SLAMs are not the same, even excluding the corrupt data. Applications like this are situational, and the designer needs to pick an appropriate input type. For collaborative mapping using Trust, the vision-based approach would be more effective and efficient than analyzing all the transmitted data.

In a situation where computations and communications cannot keep up with real-time applications, Figs. (12-13) show how this would affect trust from each perspective. The gradient of Trust is significantly reduced. As a result, the agent arrives at the appropriate trust value much slower in this case. Another effect is the loss of resolution in perceived agent data with only one update per second. Not only did this lead to fewer detections overall, but also more misinterpretations. Abrupt and sudden movements would be too fast to be registered at 1 Hz, and smooth movements appeared to be choppy. This degree of agent data resolution is unusable to calculate Trust appropriately in this example. Therefore, accounting for time delay is critical to the accuracy of perception-based trust algorithms.

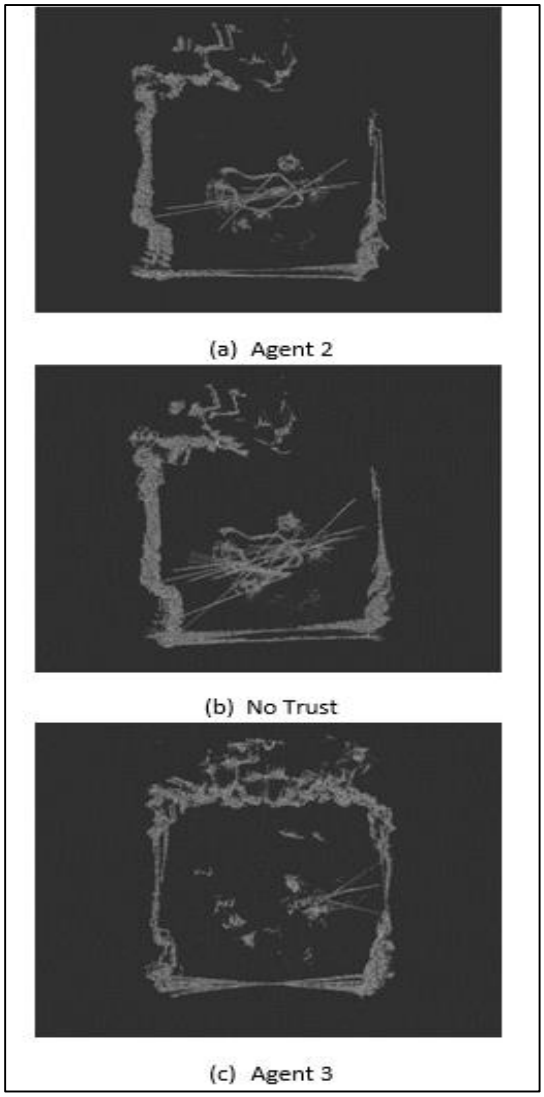

Fig. 11: SLAM mission results

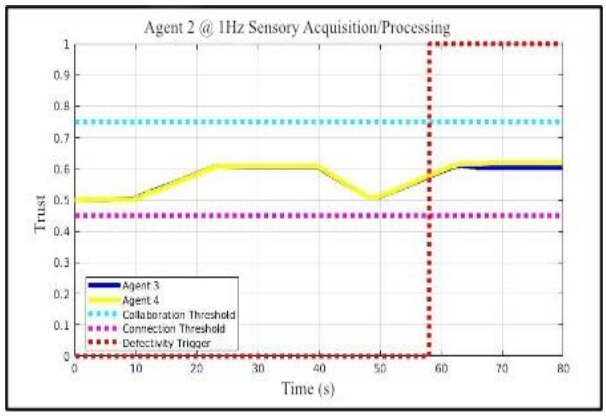

Fig. 12: Trust from agent 2 with excessive time delays in observations

DOI: 10.3844/jmrsp.2025.1.12

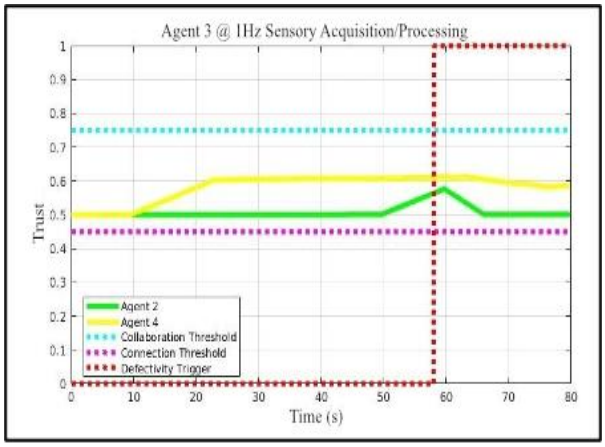

Fig. 13: Trust from agent 3 with excessive time delays in observations

Discussion

Table (2) presents the relevant work that is most comparable to OLT. Mikulski *et al.* (2012) showed the standard of Trust algorithms with RoboTrust and offered a great foundation for communicating simple signals among the network. OLT was designed to bolster RoboTrust's weakness of being vulnerable to adversarial spoofing by introducing interpretation to behaviors that cannot be hidden. Additionally, expanding to multiple behaviors like (Zhou *et al.*, 2015) gives a more holistic view unlike the single stereotype used in RoboTrust. Rishwaraj *et al.* (2017) focused more on objective completion; the experiment demonstrating OLT never showcased objective completion but does allow for visualizable objective completion to be incorporated. Das and Islam (2012) inspired a lot of OLT's reliance on democratizing Trust throughout the network and tendencies to be pessimistic about Trust. However, OLT did not incorporate any of the sophisticated decay models and load balancing schemes in an effort to maintain real-time performance. Consider that OLT was simple and robust enough to be implemented on a real multi-agent system compared to the rest who simulated everything on powerful PCs. Compared to all these works, OLT may have inherited many of their strengths, but it also acquired the assumption that success is naively visualizable as a glaring weakness. However, naivety is still an unsolved problem in Trust algorithms as a whole.

Conclusion

Trust not only impacts decision-making, but it also protects the MAS. Many issues within the MAS community still exist, such as natural agent deterioration and adversarial attacks. This research demonstrates the novelty of vision-based interaction analysis and how it offers many defenses to current robotic MAS issues as an alternative. Theoretically, this creates a range of acceptable actions an agent can perform at a given time without intruding on its freedom. This assumes an agent is around witness agents that can enforce these rules when broken. The big takeaway of this research is that trust evaluation is not meant to be complex and resource-demanding because a MAS has a simultaneous task at hand. OLT was designed to work with a variety of parallel inputs, including vision-based interactions.

This study is not perfect and does not provide an answer to some aspects. First of all, some MAS applications involve robots that never enter each other's sensory range. Another abuser of this system is the MITM attack. If the adversary catches the TCP message and modifies it before it reaches the recipient, the system will fail. This can happen without a difference in the physical behavior. Defects that do not show obvious abnormalities can also pose a huge threat to this study.

Future research using advanced cybersecurity techniques could more effectively tackle shortcomings like the sniffing parasite and even MITM attack problems. In the SLAM application explored in this study, the data is very large and unpredictable. Future research could use trust algorithms that combine different observation sources. Other future works can minimize the complexity of sensors needed to detect other agents to utilize more lightweight and cheaper UAVs and UGVs. With increasing security risks, using the agent's own senses to artificially understand its environment will increasingly become of interest. As a result, agents will need to be more distinguishable from the sensors. This can be done through unique manufacturing or new research on minimalistic agent identification modifications. OmniLightTrust is not flawless, but it has unique contributions that will eventually lead to MASs becoming more than just experiments.

Acknowledgment

This research is funded by the ONR (Award No. N00014-22-1-2724). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S.

Funding Information

The authors would like to thank the Office of Naval Research (ONR) for their financial support for this research.

Author's Contributions

Alan Kruger: Contributed to the manuscript, technical development, experimentation, and analysis of this research.

Sun Yi: Provided guidance, direction, review, and coordination for this research.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript, and that no ethical issues are involved.

References

Akintunde, M., Yazdanpanah, V., Salehi Fathabadi, A., Cirstea, C., Dastani, M., & Moreau, L. (2024). Formal Specification of Actual Trust in Multiagent Systems. *HHAI 2024: Hybrid Human AI Systems for the Social Good*, *386*, 22–35. https://doi.org/10.3233/faia240179

Arnaldy, D., & Perdana, A. R. (2019). Implementation and Analysis of Penetration Techniques Using the Man-In-The-Middle Attack. *2019 2nd International Conference of Computer and Informatics Engineering (IC2IE)*, 188–192. https://doi.org/10.1109/ic2ie47452.2019.8940872

Basheer, G. S., Ahmad, M. S., Tang, A. Y. C., & Graf, S. (2015). Certainty, Trust and Evidence: Towards an Integrative Model of Confidence in Multiagent Systems. *Computers in Human Behavior*, *45*, 307–315. https://doi.org/10.1016/j.chb.2014.12.030

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, *120*, 122–125.

Chae, S., Seo, J., Yang, Y., & Han, T.-D. (2016). Color Code AR: Large Identifiable Color Code-Based Augmented Reality System. *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 002598–002602. https://doi.org/10.1109/smc.2016.7844630

Chen, Y., Su, L., & Xu, J. (2018). Distributed Statistical Machine Learning in Adversarial Settings. *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*, 96–96. https://doi.org/10.1145/3219617.3219655

Cheng, M., Yin, C., Zhang, J., Nazarian, S., Deshmukh, J.V., & Bogdan, P. (2021). A General Trust Framework for Multi-Agent Systems. *Adaptive Agents and Multi-Agent Systems.*

Chun, G., & Guihan, L. (2009). A Method of Trust Evaluation in Multiagent System. *2009 Third International Symposium on Intelligent Information Technology Application*, 123–125. https://doi.org/10.1109/iita.2009.315

Das, A., & Islam, M. M. (2012). SecuredTrust: A Dynamic Trust Computation Model for Secured Communication in Multiagent Systems. *IEEE Transactions on Dependable and Secure Computing*, *9*(2), 261–274. https://doi.org/10.1109/tdsc.2011.57

Dekkata, S. C., Yi, S., Muktadir, M. A., & Garfo, S. (2023). LiDAR-Based Obstacle Detection and Avoidance for Navigation and Control of an Unmanned Ground Robot Using Model Predictive Control. *Journal of Mechatronics and Robotics*, *7*(1), 27–41. https://doi.org/10.3844/jmrsp.2023.27.41

Gautam, A., & Mohan, S. (2012). A Review of Research in Multi-Robot Systems. *2012 IEEE 7th International Conference on Industrial and Information Systems (ICIIS)*, 1–5. https://doi.org/10.1109/iciinfs.2012.6304778

Hallyburton, R. S., & Pajic, M. (2024). Bayesian Methods for Trust in Collaborative Multiagent Autonomy. *CoRR*, *2403*, 16956. https://doi.org/10.48550/ARXIV.2403.16956

Hardware for Every Situation. (2024). *NVIDIA Developer*. https://developer.nvidia.com/embedded/develop/hardware

Hata, A., & Wolf, D. (2014). Road Marking Detection Using LIDAR Reflective Intensity Data and Its Application to Vehicle Localization. *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 584–589. https://doi.org/10.1109/itsc.2014.6957753

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., & Murphy, K. (2017). Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3296–3297. https://doi.org/10.1109/cvpr.2017.351

Kurniawati, H., Yanzhu, D., Hsu, D., & Wee Sun, L. (2011). Motion Planning Under Uncertainty for Robotic Tasks with Long Time Horizons. *The International Journal of Robotics Research*, *30*(3), 308–323. https://doi.org/10.1177/0278364910386986

Liu, X., Datta, A., Rzadca, K., & Lim, E.-P. (2009). StereoTrust: A Group Based Personalized Trust Model. *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 7–16. https://doi.org/10.1145/1645953.1645958

Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., & Zhang, X. (2018). Trojaning Attack on Neural Networks. *25th Annual Network and Distributed System Security Symposium (NDSS 2018)*, 1–15. https://doi.org/10.14722/ndss.2018.23291

Mikulski, D. G., Lewis, F. L., Gu, E. Y., & Hudas, G. R. (2012). Trust Method for Multiagent Consensus. *SPIE Defense, Security and Sensing*, 83870E-83870E. https://doi.org/10.1117/12.918927

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). Ros: An Open-Source Robot Operating System. *ICRA Workshop on Open-Source Software*, 1–6.

Rishwaraj, G., Ponnambalam, S. G., & Kiong, L. C. (2017). An Efficient Trust Estimation Model for Multiagent Systems Using Temporal Difference Learning. *Neural Computing and Applications*, *28*(S1), 461–474. https://doi.org/10.1007/s00521-016-2354-0

Ros Wiki Org. (2016). Wiki.ros.org. ar-track-alvar. *Ros Wiki*. http: //wiki.ros.org/ar_track_alvar

Rusu, R. B., & Cousins, S. (2011). 3D is Here: Point Cloud Library (PCL). *2011 IEEE International Conference on Robotics and Automation*, 1–4. https://doi.org/10.1109/icra.2011.5980567

Shakshuki, E., & Reid, M. (2015). Multiagent System Applications in Healthcare: Current Technology and Future Roadmap. *Procedia Computer Science*, *52*, 252–261. https://doi.org/10.1016/j.procs.2015.05.071

Toda, Y., & Kubota, N. (2013). Self-Localization Based on Multiresolution Map for Remote Control of Multiple Mobile Robots. *IEEE Transactions on Industrial Informatics*, *9*(3), 1772–1781. https://doi.org/10.1109/tii.2013.2261306

Velodyne LiDAR PUCK™. (2015). *AMTECHS Co., Ltd.*, https://www. amtechs.co.jp/product/VLP-16-Puck

Wei, P., Cagle, L., Reza, T., Ball, J., & Gafford, J. (2018). LiDAR and Camera Detection Fusion in a Real-Time Industrial Multi-Sensor Collision Avoidance System. *Electronics*, *7*(6), 84. https://doi.org/10.3390/electronics7060084

Wiki. Ros. Org. (2016). Ethzasl-Icp-Mapping. *ROS Wiki*. http:// wiki.ros.org/ethzasl-icp-mapping

Zhou, P., Gu, X., Zhang, J., & Fei, M. (2015). A Priori Trust Inference with Context-Aware Stereotypical Deep Learning. *Knowledge-Based Systems*, *88*, 97–106. https://doi.org/10.1016/j.knosys.2015.08.003