INTELLI 2023

The Twelfth International Conference on Intelligent Systems and Applications

ISBN: 978-1-68558-064-3

March 13th - 17th, 2023

Barcelona, Spain

INTELLI 2023 Editors

Gil Gonçalves, University of Porto, Portugal

INTELLI 2023

Forward

The Twelfth International Conference on Intelligent Systems and Applications (INTELLI 2023), held between March 13th and March 17th, 2023, continued a series of events on advances towards fundamental, as well as practical and experimental aspects of intelligent systems and applications.

The information surrounding us is not only overwhelming, but also subject to limitations of systems and applications, including specialized devices. The diversity of systems and the spectrum of situations make it almost impossible for an end-user to handle the complexity of the challenges. Embedding intelligence in systems and applications seems to be a reasonable way to move some complex tasks from user duty. However, this approach requires fundamental changes in designing the systems and applications, in designing their interfaces and requires using specific cognitive and collaborative mechanisms. Intelligence become a key paradigm and its specific use takes various forms according to the technology or the domain a system or an application belongs to.

We take here the opportunity to warmly thank all the members of the INTELLI 2023 technical program committee, as well as all the reviewers. The creation of such a high-quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and effort to contribute to INTELLI 2023. We truly believe that, thanks to all these efforts, the final conference program consisted of top-quality contributions. We also thank the members of the INTELLI 2023 organizing committee for their help in handling the logistics of this event.

We hope that INTELLI 2023 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of intelligent systems and applications.

INTELLI 2023 Chairs

INTELLI 2023 Steering Committee

Carsten Behn, Schmalkalden University of Applied Sciences, Germany
Stefano Berretti, University of Firenze, Italy
Leo van Moergestel, HU University of Applied Sciences Utrecht, Netherlands
Gil Gonçalves, University of Porto, Portugal
Marcin Paprzycki, Systems Research Institute, Polish Academy of Sciences – Warszawa, Poland

INTELLI 2023 Publicity Chairs

José Miguel Jiménez, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

INTELLI 2023
Committee

INTELLI 2023 Steering Committee

Carsten Behn, Schmalkalden University of Applied Sciences, Germany
Stefano Berretti, University of Firenze, Italy
Leo van Moergestel, HU University of Applied Sciences Utrecht, Netherlands
Gil Gonçalves, University of Porto, Portugal
Marcin Paprzycki, Systems Research Institute, Polish Academy of Sciences – Warszawa, Poland

INTELLI 2023 Publicity Chairs

José Miguel Jiménez, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

INTELLI 2023 Technical Program Committee

Azizi Ab Aziz, Universiti Utara Malaysia, Malaysia
Lounis Adouane, Université de Technologie de Compiègne, France
Leo Aguilera, 33 Technologies LLC, USA
Ari Aharari, SOJO University, Japan
Bilal Ahmad, University of Warwick, UK
Zaher Al Aghbari, University of Sharjah, UAE
Miltos Alamaniotis, University of Texas at San Antonio, USA
Antonios Alexos, University of California Irvine, USA
Sarra Alqahtani, Wake Forest University, USA
Rachid Anane, Coventry University, UK
Zahra Ebadi Ansaroudi, University of Salerno, Italy
Olugbenga Moses Anubi, Florida State University, USA
Benjamin Aziz, University of Portsmouth, UK
Arvind Bansal, Kent State University, USA
Suzanne Barber, The University of Texas at Austin, USA
Dariusz Barbucha, Gdynia Maritime University, Poland
Carmelo Bastos-Filho, University of Pernambuco, Brazil
Rafael Batres, Tecnologico de Monterrey, Mexico
Carsten Behn, Schmalkalden University of Applied Sciences, Germany
Fayçal Bensaali, Qatar University, Qatar
Lyes Benyoucef, Aix-Marseille University, France
Giuseppe Berio, IRISA | Université de Bretagne Sud, France
Stefano Berretti, University of Firenze, Italy
Jonathan Berrisch, University of Duisburg-Essen, Germany
Mahdis Bisheban, National Research Council Canada (NRC), Canada
José Miguel Blanco, Masaryk University, Brno, Czech Republic
Francisco Bonin Font, University of the Balearic Islands, Spain
Lucas Botoni de Souza, Federal University of Technology - Paraná, Brazil
Frederic Bousefsaf, LCOMS | Université de Lorraine, France

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

A Novel Diagnosis Concept for Verification of Neural Networks on the Target Hardware

Ilkay Wunderlich
Technical University of Dresden
Institute of Computer Engineering
Dresden, Germany
email: ilkay.wunderlich@tu-dresden.de

Jens Braunes
PLS Programmierbare Logik & Systeme GmbH
Lauta, Germany
email: jens.braunes@pls-mc.com

Rainer G. Spallek
Technical University of Dresden
Institute of Computer Engineering
Dresden, Germany
email: rainer.spallek@tu-dresden.de

***Abstract*—Neural networks are increasingly being used in embedded applications. It is important to check whether the trained network fulfills its tasks not only in simulations, but also on real target hardware. This is of particular importance in safety-critical applications such as plant control or autonomous driving. For this reason, a diagnostic concept for Artificial Intelligence (AI)-based systems based on the Universal Debug Engine from PLS Programmierbare Logik & Systeme GmbH was developed at the Technical University of Dresden. This concept enables developers to verify the hardware implementation of the neural network. After the concept was developed, it was successfully tested on several example applications of which two are presented in the paper.**

***Index Terms*—neural networks, hardware, diagnosis, debugging.**

I. Introduction

Methods from the field of Artificial Intelligence (AI) are still advancing and have also produced impressive applications in connection with the Covid-19 pandemic. For example, one application of these methods, developed by researchers at the University of Central Florida, is able to use computed tomography images to identify whether COVID-19 pulmonary pneumonia is present [1]. Studies show that this AI based algorithm, with a detection rate of 90%, has similar accuracy as doctors who specialize in lung disease. However, in the area of health care, the AI methods typically run on systems with large computational power and are supervised by an expert, such as a doctor.

But in addition to this impressive use case, AI applications are also being used more and more frequently in the embedded area. Especially in today's automotive systems, techniques from the discipline of AI are increasingly being used. In this field, neural networks are a tool that should be highlighted, which are particularly well suited for detection and classification tasks in the field of computer vision. One common applications for networks is the fusion of sensor data as described by X. Zeng et al. in [2] with focus on sensor data in automotive systems. Another use case is the detection and classification of traffic participants for visual environment recognition. For this use case, plenty of public traffic data sets like the Berkeley Deep Drive Data Set [3] or City Scapes Data Set [4] exist. In such safety critical applications which mostly run without any supervision, it is of great importance to verify and test the implementation on the target hardware.

In [5] by J. Zhang et al., a large review of testing and verification methods for neural networks is given. However, the majority of the presented publications aim on software verification without the inclusion of the target hardware.

A more hardware focused method, which tests the hardware implementation of the neural network, is described by S. Huang et al. [7]. As described in the publication title, each layer is individually checked for correctness using an extra transition module for a specific Field Programmable Gate Array (FPGA)-based hardware accelerator.

In this research, the focus is set on a diagnosis concept, which allows developers to verify the neuronal network on a broad range of different embedded hardware solutions. Instead of the implementation of an extra module on the target hardware, the Universal Debug Engine® (UDE) from PLS Programmierbare Logik & Systeme GmbH (PLS) is used [8]. UDE is a debugger solution for various microcontroller families such as AURIX, TriCore, Power Architecture, Cortex, Arm, STM32 and more. The physical interface between the hardware and the host computer with UDE is realized with the PLS Universal Access Device (UAD).

The proposed method aims exclusively at the verification of the neural network hardware. False results of the neural network due to poor training, under-sizing of the neural network, or incomplete case coverage in the training data are not part of the diagnosis. Such issues must be clarified before porting the network on the hardware. For example, in [6] by A. Kirchknopf et al., methods to investigate detection results are given.

The rest of the paper is structured as follows. In Section 2, the "life" is described as an illustration of the different phases of a neural network. Section 3 deals with the possible sources of possible errors on hardware. In Section 4, the diagnosis concept to detect these errors is described theoretically whereas Section 5 demonstrates the diagnosis concept on two hardware examples. Section 6 concludes the work and alludes to additional features as well as the future work.

 ISBN: ISBNFILL

II. The "Life" of a Neural Network

The "life" of a neural network generally consists of the following three phases:

A. Training Phase

With large amounts of data, all trainable weight and bias parameters of the network are adjusted by gradient-based training algorithms. At the same time, the accuracy of the network is determined using validation data. The individual elements of the training and validation data, consisting of pairs of input values and the associated output values, correspond to the structures of the input and output layers. Due to the high computing intensity, high-performance Graphics Processing Units (GPUs) or special cloud services are usually used to train neural networks. Common frameworks for training neural networks are Tensorflow [9] and Pytorch [10].

The output of the Training Phase is called the Golden Model, which contains structure, trained parameters and other meta information of the neural network. A concrete example for the Golden Model could be a neural network which is trained with Tensorflow and exported as a Tensorflow Keras model file: `network.h5`.

B. Adaptation Phase

After the training phase, neural networks can be accelerated using optimization steps such as batchnorm fusion or pruning. With suitable quantization methods, the arithmetic operations are also transformed from floating point to integer formats. The adjustments reduce the arithmetic complexity and allow neural networks to run on embedded processors with acceptable power consumption and latency. A commonly known tool for optimization and quantization is Tensorflow Lite which is build in Tensorflow [13]. However, there are also many custom methods for adapting neural networks as described by I. Wunderlich, B. Koch and S. Schönfeld in "An Overview of Arithmetic Adaptations for Inference of Convolutional Neural Networks on Re-configurable Hardware" [11].

The result of the Adaptation Phase is called the Silver Model and represents the adapted and quantized version of the Golden Model. In general, there are quantization losses due to the optimization and adaptation steps, but these ideally lead to minor deviations between the outputs of the Golden and Silver Models. Nevertheless, it is advisable to carry out further tests with test data to ensure that the deviations are within an acceptable range.

After successfully generating the Silver Model, it can be ported to the target device. In the example from above, the Silver Model could be a Tensorflow Lite model file: `network.tfl`.

C. Inference Phase

The inference phase describes the use or application of the fully trained and adapted neural network. Typically, unknown data is interpreted and evaluated accordingly.

In the example, the network is running on a STMicroelectronics Micro Controller Unit (MCU). In Figure 1, the general interaction as well as the example between the three phases, the Golden and Silver Model are schematically visualized.

III. Possible Sources of Error

The use of neural networks offers several advantages, especially in safety-critical applications in which large amounts of data must be evaluated and processed within a short period of time. But only if it works as desired on the real hardware. Ultimately, this can only be completely clarified with a detailed hardware diagnosis, because, unfortunately, there is a whole range of potential sources of error. They can essentially be divided into the following categories:

- Conversion Errors:
  When converting in the adaptation phase, incorrect quantization can lead to arithmetic overflows and underflows, thus reduce the quality of the predictions immensely.
- Porting Errors:
  After the adaptation, errors such as exceeding memory limitations, incorrect programming of the interfaces or similar can occur when porting the quantized model.
- Implementation Errors:
  When implementing neural networks, there are many sources of error related to arithmetic, flow control, and data management. With frameworks such as STMicroelectronics' X-CUBE-AI, MCU manufacturers are already providing tested and functional code [12]. However, similar sources of error are conceivable again when adapting or expanding. It is, therefore, essential to ensure that neural networks are correct, especially in the case of safety-critical applications.

In order to enable the fastest, most efficient possible checking and verification of all those factors in the future, a new diagnostic concept for AI-based systems was developed at the Technical University Dresden in cooperation with PLS.

IV. The Diagnosis Concept

The diagnosis concept is realized as a diagnosis loop, which iteratively compares the results of the hardware with the results of the Golden or Silver Models. In Figure 2, the diagnosis loop is schematically shown. The diagnosis concept is implemented as a Python project on a host computer which is connected to the target hardware via the UAD. The individual processes (rectangular blocks in Figure 2) are described below:

A. Data Extraction

In the first step, the input matrix x_{HW} and corresponding output matrix y_{HW} are read via the Python automation interface of the software debugger UDE and hardware tool UAD from PLS.

The index $(.)_{\mathrm{HW}}$ indicates that the matrices come from the hardware. The dimensions of the matrices vary depending on the application. In computer vision tasks, the general input matrix x can have the following dimensions, as shown in (1):

$$\dim(x) = (W, H, C) \tag{1}$$

 ISBN: ISBNFILL

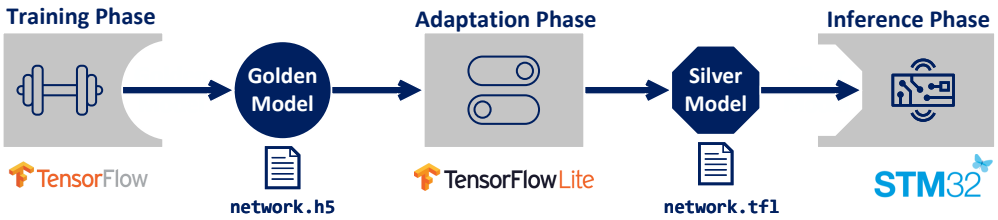


Figure 1. General and exemplary interaction between Training Phase, Adaptation Phase, Inference Phase, Golden Model and Silver Model.

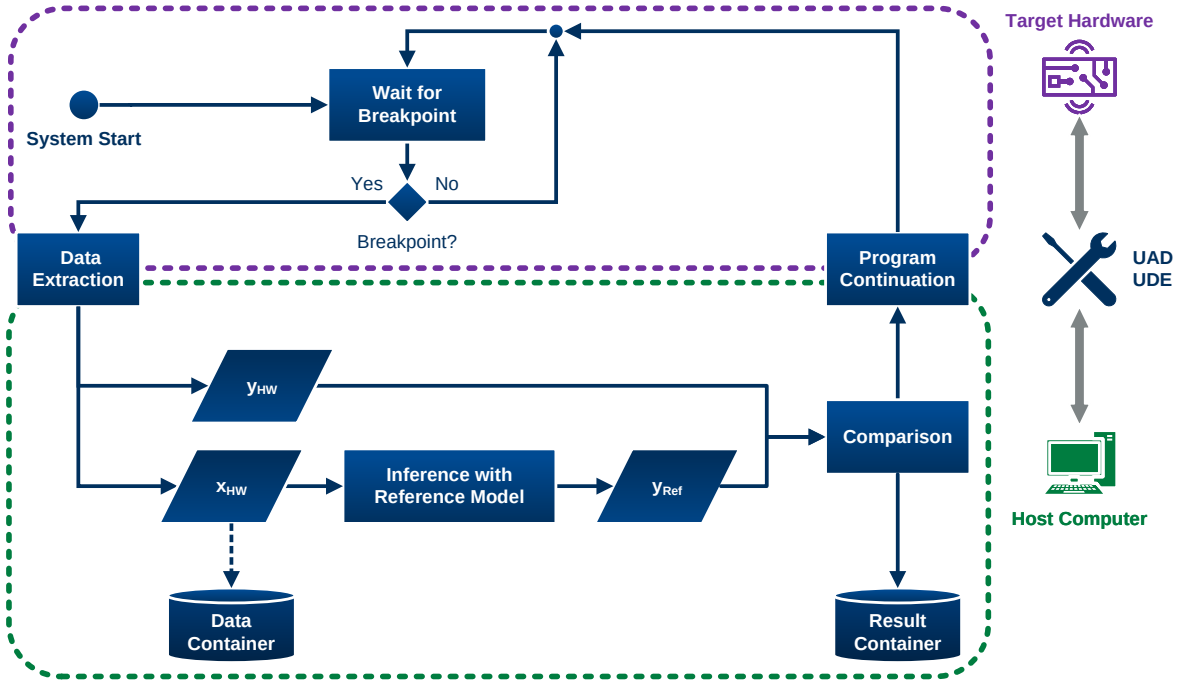


Figure 2. Schematic structure of the diagnostic loop for the continuous comparator-based analysis of the hardware and model outputs. Purple dashed frame: Processes belonging to the context of the target hardware. Green dashed frame: Processes belonging to the context of the host computer.

with W and H as the image width and height and C the color channel, which is generally $C = 3$ for Red Green Blue (RGB) images or $C = 1$ for grayscale images.

A common novice example of neural networks is a classification network which assigns to the input matrix x the classes "cat" or "dog". In this case, the input matrix might have the dimensions $\dim(x) = (128, 128, 3)$ and the output is typically encoded by a two-element output matrix y with $\dim(y) = (2, 1)$ where the elements correspond to the class probability for "cat" or "dog".

B. Inference with Reference Model and Comparison

Depending on availability, either the Golden or Silver Model or both can be used as the reference model. The inference with the chosen reference model is performed on the host computer and yields the reference output matrix y_{Ref}.

Afterwards, a comparison between y_{HW} and y_{Ref} is performed. If the hardware implementation of the neural network is correctly implemented, the following relationships between the output matrices must apply in every single iteration of the diagnosis loop:

$$y_{\mathrm{HW}} = y_{\mathrm{Ref,\ Silver}} \tag{2}$$

$$y_{\mathrm{HW}} \approx y_{\mathrm{Ref,\ Golden}} \tag{3}$$

The relationship from equation (2) can be easily checked using a binary equivalence test for y_{HW} and $y_{\mathrm{Ref,\ Silver}}$, as shown in (4).

$$\mathrm{Eq}\left(y_{\mathrm{HW}}, y_{\mathrm{Ref,\ Silver}}\right) = \begin{cases} \text{True}, & y_{\mathrm{HW}} = y_{\mathrm{Ref,Silver}} \\ \text{False}, & y_{\mathrm{HW}} \neq y_{\mathrm{Ref,Silver}} \end{cases} \tag{4}$$

If they match exactly, "True" is returned, otherwise "False". For the second relationship from equation (3), the deviations to be expected are quantified by a difference metric, for example the Mean Squared Error (MSE), as shown in (5).

$$\mathrm{MSE}\left(y_{\mathrm{HW}}, y_{\mathrm{Ref,\ Golden}}\right) = \frac{1}{N} \sum_{\substack{u \in y_{\mathrm{HW}} \\ v \in y_{\mathrm{Ref,Golden}}}} (u - v)^2 \tag{5}$$

 ISBN: ISBNFILL

Other difference metrics, like the squared Euclidean [14] distance are suitable as well.

The results of the comparison are stored for each iteration of the diagnosis loop for later visualization and evaluation purposes. Optionally, the input matrices x_{HW} of each diagnosis iteration can be stored as well. This data can later be used for training or validation purposes.

C. Program Continuation and Wait for Breakpoint

All necessary interactions with the target hardware can be effectively implemented with the Python automation interface of UDE. The debugger software UDE not only supports most high-end micro controllers but also multi-core SoCs (System on Chips), which are well suited for AI applications. With the access devices of the UAD family, also fast and secure communication with the respective target system is ensured.

Additionally to the data extraction process, the program flow needs to be controlled by starting the system and setting a breakpoint where the input and output matrices can be read. After the inference and comparison processes are done, the program is resumed until the next breakpoint is reached.

In this context, setting the breakpoints is not trivial. It is essential to ensure that the read input and output matrices belong together. As a rule, this must be done manually and for each application individually.

V. The Practice Test

In order to clarify whether the diagnostic strategy described can generally be implemented in practice, extensive tests were carried out with conventional development boards for different AI applications. Two of the applications are described in detail below.

A. Acoustic Scene Classification

STMicroelectronics' Sensor Tile Kit (STEVAL-STLCS01V1) is an all-round sensor system [15]. It is equipped with the following components, among others:

- STM32L476JGY Low Power MCU with Arm Cortex-M4 Floating Point Unit
- BlueNRG-MS-Bluetooth-Prozessor
- Various sensers: microphone, barometer, thermometer, accelerometer, etc.

In Figure 3, the hardware setup consisting of STM32 Sensor Tile and UAD2$^{\mathrm{pro}}$ is visualized.

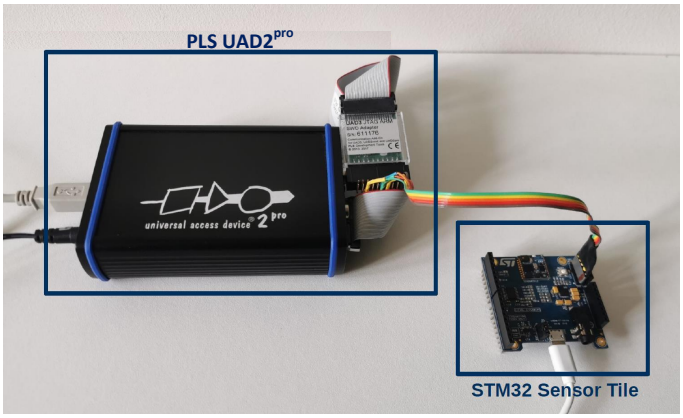


Figure 3. STM32 Sensor Tile connected to the PLS UAD2$^{\mathrm{pro}}$.

AI applications for the sensor tile are already pre-implemented in the FP-AI-Sensing1 function package provided by ST from the STM32Cube software development system [16]. From these examples, the Acoustic Scene Classification (ASC) was selected for the practical test of the diagnostic concept. In this application, the ambient noise is analyzed with a neural network based on microphone data and assigned to the classes "indoor", "outdoor" and "vehicle". The input matrix x_{HW} is a transformed spectrogram with the dimensions $\dim(x_{\mathrm{HW}}) = (32, 30, 1)$ and the output matrix y_{HW} is a three-element matrix $\dim(y_{\mathrm{HW}}) = (3, 1)$ with the respective class probabilities. The golden model and the silver model of the neural network for the ASC are already available in the FP-AISensing1 function package, so the training and adaptation phases can be skipped. The Golden and Silver Models are available as a Tensorflow Keras model (`ASC.h5`) and as a Tensorflow Lite 8-Bit quantized Model (`ASC.tfl`).

In order to determine suitable interfaces for data transfer using the UDE debugger, the FP-AI-Sensing1 function package must first be compiled and build with the STM32Cube Integrated Development Environment (IDE). The resulting Executable and Linkable Format (ELF) file `SENSING1.elf` is then programmed into the flash memory of the sensor tile with the UDE Multi Program Loader and a breakpoint is set in line 188 of the `asc_processing.c` module for the ASC diagnosis, as shown in Figure 4. At this code line, both the input matrix x_{HW} and the associated output matrix y_{HW} can be read via the automation interface of the UDE.

The diagnostic loop can then be run using the analysis system and the reference models. In Figure 5 an example of diagnosis loop with 100 iterations is shown. It can be seen that the output matrices of the hardware implementation and those of the silver model always match. The averaged MSE is $\mu_{\mathrm{MSE}} = 3.3 \cdot 10^{-4}$ and is therefore within a range which is to be expected based on the tensorflow lite quantization. Further tests with larger iteration counts and different acoustic environments have shown the same behavior, so that in summary it can be said that the equations 2 and 3 apply and the hardware implementation is working correctly.

B. Recognizing People

A development board for image processing algorithms from STMicroelectronics based on the 32-bit low-power MCU STM32H743VI was selected as a further application example of the diagnostics environment. The OpenMV (Open Source Machine Vision) Cam H7 camera module can be used, among other things, to implement neural networks for computer vision applications [17]. The OpenMV Github repository offers already implemented AI entry-level examples [18]. The focus of the following test is on recognizing people. A classification network assigns a two-element probability matrix y_{HW} with the classes "person" and "no person" to the input image matrix

 ISBN: ISBNFILL

```
asc_processing.c
175 ASC_StatusTypeDef ASC_NN_Run(float32_t *pSpectrogram, float32_t *pNetworkOut)
176 {
177   ai_i8 AscNnOutput[AI_ASC_OUT_1_SIZE];
178   ai_i8 AscNnInput[AI_ASC_IN_1_SIZE];
179
180   /* Z-Score Scaling on input feature */
181   for (uint32_t i = 0; i < SPECTROGRAM_ROWS * SPECTROGRAM_COLS; i++)
182   {
183     pSpectrogram[i] = (pSpectrogram[i] - featureScalerMean[i]) / featureScalerStd[i];
184   }
185
186   aiConvertInputFloat_2_Int8(AI_ASC_MODEL_NAME, AI_ASC_MODEL_CTX,pSpectrogram, AscNnInput);
187   aiRun(AI_ASC_MODEL_NAME, AI_ASC_MODEL_CTX, AscNnInput,AscNnOutput);
188   aiConvertOutputInt8_2_Float(AI_ASC_MODEL_NAME, AI_ASC_MODEL_CTX,AscNnOutput, pNetworkOut);
      breakpoint
190   return ASC_OK;
191 }
```


Figure 4. View of the core function of ASC. At the position of the selected breakpoint, the input matrix x_{HW} and the associated output matrix y_{HW} for the diagnostics from the hardware can be read.

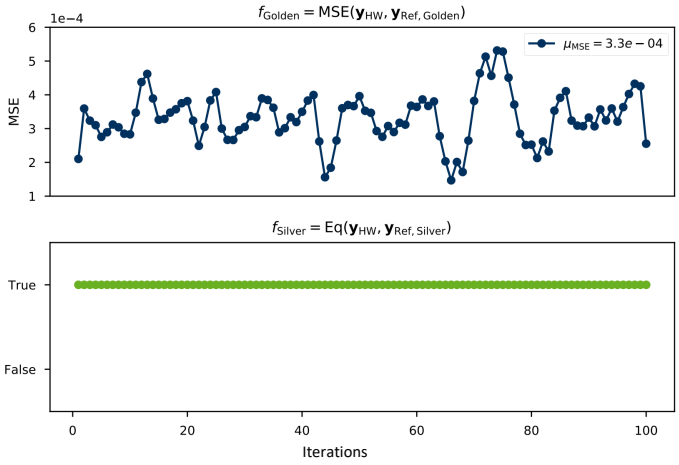


Figure 5. Evaluation diagram with 100 iterations of the diagnostic loop for the Sensor Tile diagnosis system.

Figure 6. Hardware setup with OpenMV Cam H7 connected with the PLS UAD$^{\mathrm{pro}}$ and an example of an input image for recognizing people. The reference models and hardware implementation certainly recognized the person (Ilkay Wunderlich) in the image.

x_{HW}. The neural network uses a grayscale image with the resolution $\dim(x_{\mathrm{HW}}) = (640, 480)$ as the input matrix for the person classification. Similar to the implementation of the diagnostic system for the ASC, a breakpoint is set at a suitable point in order to read out the input and output matrices via UDE. Here, too, the evaluation of the diagnostic loop confirms that the outputs of the OpenMV Cam and the predictions of the silver model always match. The deviations between y_{HW} and $y_{\mathrm{Ref,\,Golden}}$ are withing acceptable range as well.

Figure 6 shows the hardware setup, an example image for person detection and the corresponding output matrices y_{HW} and $y_{\mathrm{Ref,\,Silver}}$.

Using the developed diagnostic system based on the debugger UDE, not only the correctness of the AI implementation can be verified for the computer vision application but also the collected data x_{HW} from each iteration is saved and can later be used as new training or validation data. Especially for computer vision tasks, it is a great benefit to collect data of the target camera with its distinctive lens and exposure settings to increase the performance of the neural network.

VI. Conclusion

As the examined application examples show, efficient diagnostic systems can be implemented with suitable tools for AI-based embedded applications. With the comparisons between the predictions of the target hardware and the reference models, developers can ensure that the AI implementation is working correctly. The debugger UDE realizes the Python-based diagnostic system as a system in the loop. Additional features such as data collection, processing time measurement or continuous integration can be flexibly integrated into or around the diagnostic loop via the UDE Python interface.

The future work is on the one hand to generalize the Python-based concepts in order to add a direct neural network diagnosis feature to UDE and, on the other hand, to extend the approach to other emerging or established AI methods such as spiking neural networks, decision trees, hidden Markov models, etc.

 ISBN: ISBNFILL

ACKNOWLEDGMENT

The investigation of the described novel diagnosis concept was developed by PLS as well as the chair of VLSI-Design, Diagnostic and Architecture of the Institute of Computer Engineering at the Technical University Dresden.

REFERENCES

[1] University of Central Florida, "AI can detect COVID-19 in the lungs like a virtual physician, new study shows: Algorithm can accurately identify COVID-19 cases, as well as distinguish them from influenza," https://www.sciencedaily.com/releases/2020/09/200930144426.htm, ScienceDaily, 2020.

[2] X. Zeng, Z. Wang and Y. Hu, "Enabling Efficient Deep Convolutional Neural Network-Based Sensor Fusion for Autonomous Driving," https://doi.org/10.1145/3489517.3530444, Association for Computing Machinery, 2022.

[3] Y. Fisher et al., "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," https://arxiv.org/abs/1805.04687, arXiv, 2018.

[4] M. Cordts et al., "The Cityscapes Dataset for Semantic Urban Scene Understanding," https://arxiv.org/abs/1604.01685, arXiv, 2016.

[5] J. Zhang and J. Li, "Testing and verification of neural-network-based safety-critical control software: A systematic literature review," https://www.sciencedirect.com/science/article/pii/S0950584920300471, Information and Software Technology, 2020.

[6] A. Kirchknopf, D. Slijepcevic and I. Wunderlich "Explaining YOLO: Leveraging Grad-CAM to Explain Object Detections," https://diglib.tugraz.at/download.php?id=621f34738ca16&location=datacite, Proceedings of the Workshop of the Austrian Association for Pattern Recognition, 2021.

[7] S. Huang et al., "Design and Implementation of Convolutional Neural Network Accelerator with Variable Layer-by-Layer Debugging," https://doi.org/10.1145/3234804.3234806, Association for Computing Machinery, 2018.

[8] PLS Programmierbare Logik & Systeme GmbH, "Universal Debug Engine (UDE)," https://www.pls-mc.com/products/universal-debug-engine, retrieved 14.12.2022.

[9] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," https://www.tensorflow.org/, 2015.

[10] A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library," http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf, 2019.

[11] I. Wunderlich, B. Koch, and S. Schönfeld, "An Overview of Arithmetic Adaptations for Inference of Convolutional Neural Networks on Re-configurable Hardware," ALLDATA 2020, The Sixth International Conference on Big Data, Small Data, Linked Data and Open Data, pp. 34–40, 2020.

[12] STMicroelectronics, "X-CUBE-AI: AI expansion pack for STM32CubeMX," https://www.st.com/en/embedded-software/x-cube-ai.html, retrieved 14.12.2022.

[13] Tensorflow, "Tensorflow Lite," https://www.tensorflow.org/lite, retrieved 14.12.2022.

[14] N. H. Spencer, "Squared Euclidean Distances," Essentials of Multivariate Data Analysis, CRC Press, p. 95, ISBN 978-1-4665-8479-2, 2013.

[15] STMicroelectronics, "STEVAL-STLCS01V1: SensorTile connectable sensor node: plug or solder," https://www.st.com/en/evaluation-tools/steval-stlcs01v1.html, retrieved 14.12.2022.

[16] STMicroelectronics, "FP-AI-SENSING1: function pack for ultra-low power IoT node with artificial intelligence (AI) application based on audio and motion sensing," https://www.st.com/en/embedded-software/fp-ai-sensing1.html, retrieved 14.12.2022.

[17] K. W. Agyeman, I. Abdelkader, K. Wong and Y. Xu, "OpenMV Cam," https://openmv.io/, retrieved 14.12.2022.

[18] K. W. Agyeman, I. Abdelkader, K. Wong and Y. Xu, "OpenMV: Open-Source Machine Vision," https://github.com/openmv/openmv, retrieved 14.12.2020.

 ISBN: ISBNFILL

Video-based Object Detection Using Voice Recognition and YoloV7

Issa Abdoul Razac Djinko
Department of Computer Science
and Information Technology
University of the District of Columbia
Washington, DC, USA
Email: issaabdoulrazac.djin@udc.edu

Thabet Kacem
Department of Computer Science
and Information Technology
University of the District of Columbia
Washington, DC, USA
Email: thabet.kacem@udc.edu

***Abstract*—Artificial Intelligence (AI) developments in recent years have allowed several new types of applications to emerge. In particular, detecting people and objects from sequences of pictures or videos has been an exciting field of research. Even though there have been notable achievements with the emergence of sophisticated AI models, there needs to be a specialized research effort that helps people finding misplaced items from a set of video sequences. In this paper, we leverage voice recognition and Yolo (You Only Look Once) real-time object detection system to develop an AI-based solution that addresses this challenge. This solution assumes that previous recordings of the objects of interest and storing them in the dataset have already occurred. To find a misplaced object, the user delivers a voice command that is in turn fed into the Yolo model to detect where and when the searched object was seen last. The outcome of this process is a picture that is provided as evidence. We used Yolov7 for object detection thanks to its better accuracy and wider database while leveraging Google voice recognizer to translate the voice command into text. The initial results we obtained show a promising potential for the success of our approach. Our findings can be extended to be applied to various other scenarios ranging from detecting health risks for elderly people to assisting authorities in locating potential persons of interest.**

***Index Terms*—Artificial Intelligence, Object Detection, Voice Recognition.**

I. Introduction

Two hundred years ago, no one could have imagined the technology would evolve to the extent it reached today. The basic technology we take for granted would have been considered witchcraft in the 1800s. Yet nowadays, technology has radically re-imagined the applications and services that we rely on in our daily lives. In particular, AI [1] is a field that aims to equip machines with the capability of dealing with information from the perception to the inference. Actually, this field is not quite new since the first reference to AI dates back to 1943 when McCullouch and Pitts [2] formally defined the first artificial neuron for the Turing Machine. Since then, this field has seen considerable development at a fast pace in recent years as it can be used for event prediction, speech recognition or even visual perception.

Conversely, video surveillance has benefited from the recent developments in cloud computing and communication but the challenge of detecting objects from images and video sequences has not been fully overcome yet. Several research efforts, such as in [3], proposed techniques based on machine learning, deep learning or even optical flow methods in this context with concrete performance gains.

In this paper, we tackle this problem from a different angle as we leverage video-based object detection and voice recognition to help people locate misplaced items. Our approach takes advantage of Yolo and Google voice-text recognizer. The latter is used to set up keywords by converting the user's voice into text before feeding it to Yolo that is used in turn to detect the object the user is searching for. When our Yolo model receives a keyword, it assigns a specific version of the trained model to find the objects and saves the resulting video, sorted by date, into a pre-defined folder. Therefore, the user can know where the searched item was seen last by watching the latest video.

The research objectives of this paper are (1) to explore how machine learning can be leveraged to identify objects from sequences of images and videos with high performance (2) to build an easy-to-use solution to find misplaced items in the household by combining machine learning and voice recognition (3) to leverage the outcome of this paper in tackling similar problems such as detecting health risks for elderly people.

The rest of the paper is organized as follows. Section II presents the related work. Section III explains the basic notions of Yolo. Section IV describes our approach in details. Section V presents the simulation setup while Section VI discusses the results of the conducted experiments. Section VII concludes our paper and highlights our future work.

II. Related Work

Karmarkar and Honmane [10] proposed a system to help visually impaired people using Yolo. The model was trained with the Coco dataset of 330K images of various daily used objects [10]. A bounding box is then generated around each detected object. The method generates five values to estimate the position and displacement of the object. When the camera focuses on the object, a triangle is developed around it and the closer the object is to the camera, the width and angles of the triangle increase. Then, the approach determines the distance to the object in question. The paper is different from our in the sense that our study assumes that the user is able to see and

 ISBN: ISBNFILL

locate the recording video in order to find where the object was seen last. Also, we do not rely on the distance from the object to guide the user.

Zhang et al. [12] proposed using Yolo version 3 from camera feeds of an Unmanned Aerial Vehicle (UAV) to detect pedestrians. The authors rely on the box prediction feature of YOLO and the Feature Pyramid Network (FPN) to draw boxes upon the detected objects . The paper is very efficient when it comes to following something in motion, something or someone who is moving very fast. Our paper is different as it does not follow an object in motion, but we instead assume that the objects are located within the user's home and but they were just misplaced.

Jana and Biswas [11] proposed an approach that relies on recorded videos to identify any objects. By processing 40 frames per second (fps), the model divides images into NxN number of grids, effectively identifying the grids containing objects, and constructs a bounding box around them. The authors apply Yolo version 2 to detect and classify the objects, then assign an accuracy percentage. The findings of this paper are aligned with ours in the sense that the detection of objects is done through video recordings. However, we are running our own custom Yolo version detection model that achieves much better performance gains, as shown in Figure 1. Also, our goal is to find specific objects in our videos and the model was trained with our own dataset to guarantee maximum detection.

Priyankan and Fernando [13] proposed an approach based on Yolo to identify different species of fish by running an analysis on fish images. They created a mobile application by gathering these components using the following experimental setup: a PC equipped with core i7 CPU, 16GB of DDR3 memory, Ubuntu 14.04 64-bit, NVIDIA DIGITS 5, MATLAB R2012, and B-BOX-label. Since Yolo can use 40–90 fps, this approach used a neural network consisting of 24 convolutional layers. Then, the authors trained the model with 800 to 900 images and found 16 fish species. The model takes 3 to 20 seconds to detect the species. The test result revealed a 77% accuracy in bounding and classifying the fish species. Our paper is also customizing a model on a set of images of specific objects. However, our paper makes object detection based on videos while adding a voice-text-translation feature to fine-tune the objects the user is looking for.

III. Video-based Object Detection Fundamentals

Yolo is an object detection algorithm that divides any image into grids and determines the pixels in which the object is. When the location of the object is determined in the image, a bounding box is then drawn around it and labeled [7]. Yolo was first introduced in 2016 by Jason Redmon et al. [5]. Since then, there have been a great deal of Yolo versions that were proposed: actually 7 versions, and each one comes with its different levels of training. Figure 1 shows the differences in terms of performance of these versions when trained with the Coco dataset. The purple curve shows the latest one, i.e. Yolov7. It can be observed that Yolov7 outperforms the rest of the other five versions by a significant margin.

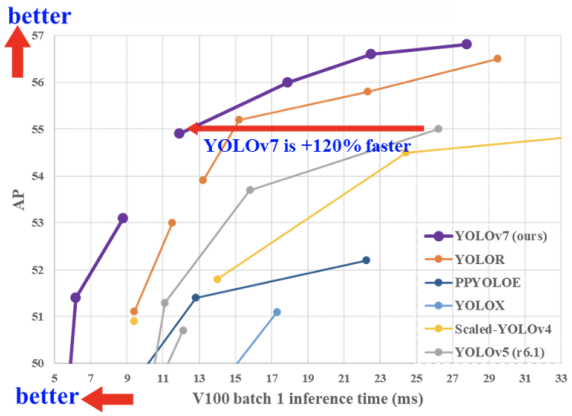


Fig. 1. Comparison of Yolo-v7 with previous versions [6].

IV. Proposed Approach

A. Approach Rationale

The motivation behind this paper is to provide an easy-to-use service for people to find misplaced items by combining several techniques including video-based object detection using a custom Yolov7 model, a Google voice recognizer and video surveillance data.

B. Approach Description

The approach can be summarized in 7 steps, as shown in Figure 2:

- The user sends a voice command to the application by providing the keyword of the searched object.
- The Google recognizer transcribes the analog voice input into text.
- It compares the scripted input with a predetermined dataset to see if any item was called forth.
- When this is completed, theGoogle voice-text recognizer function sends the desired keyword to the Yolov7 module that we customized with a set of specific keywords.
- When the Yolo module receives the keyword, it goes through the videos that were recorded every day in certain time intervals to detect when the object was seen last.
- The algorithm then saves a video with the bounding boxes including the time and date that the object was detected in a folder of our choosing.
- Finally, upon locating the latest video entry in that folder, the user gets an evidence of the last location where the searched object was seen last.

C. Video-Based Object Detection Algorithm

In line 1 of the algorithm, we define the different inputs with capital letters for simplicity. The user was given the letter A, the Google voice-to-text recognizer was given the letter Y, and the different custom models were given the letter X. Line 2 starts the event-based loop that detects when the user sends a voice command before the Google recognizer picks it up from

 ISBN: ISBNFILL

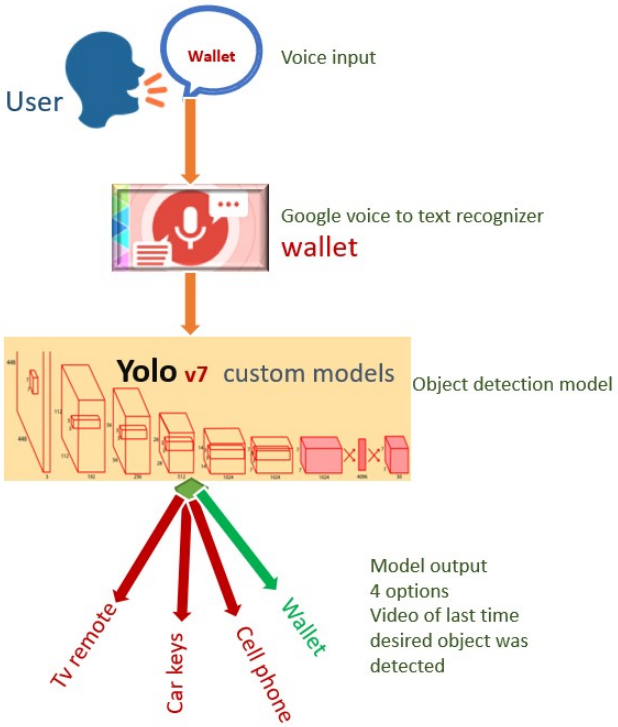


Fig. 2. High-level architectural view of the approach.

there and transcribes the voice into clear text. In line 3, the Google recognizer asks for a confirmation from the user about the transcribed voice command. In lines 4-6, the algorithm checks if the keyword was validated before passing it to the object detection function leveraging the Yolo model. Lines 6-9 deal with the situation in which a keyword is not confirmed, which in turn restarts the process until another keyword is confirmed. In lines 10-14, after a keyword is sent to the Yolo model, the object detection function detects when the searched object was seen last, generates an image output, and saves it in a predetermined folder.

V. Simulation

The simulation process was the most interesting part of this project. First, we had to clone the Yolov7 from GitHub [8] then installed the required dependencies. In order to get better performance results, we did not want to use an already trained model for this project. That is why, we decided to train our custom model ourselves using the initial cloned model that was trained on the Coco dataset. Therefore, we took some images of the objects from the videos for the convolutions to successfully go through and find the objects. After collecting the images, we labelled them using ”Makesense.ai”, which is a free open-source tool to label images. The images were 461 in total: 80% for training and 20% for validation. The labeling process was quite long because we had to do this one image at a time. In the beginning of the training, the model collected 362 labelled images for training and 100 for validation. There were 95 convolutions in this base weight while the number

Algorithm 1: Video-based object detection algorithm.

```
1  Inputs: A=user, Y=Google voice-text recognizer,
     X=Yolov7 Custom Model;
2  while voice detected by Y do
3  |   Y asks for confirmation it is an input from A;
4  |   if keyword is confirmed then
5  |   |   Y sends keyword to X;
6  |   end
7  |   else
8  |   |   Star over;
9  |   end
10 |   if X receives keyword then
11 |   |   X runs last video;
12 |   |   X output object detection;
13 |   |   X saves output in folder;
14 |   end
15 end
```

of epochs was 100. Finally, we created a folder that would receive the output of the trained model.

Figure 3 shows the output of the anaconda power-shell Prompt at the very last epoch after the completion of the training process. The model found 100 images per recognized keyword with different precision levels per keyword, while some of the performance metrics we used included the recall rate and the mean average precision. The amount of time it took for the training to be completed was about 10 hours run on the local machine. It is important to note that we wanted to establish a proof of concept to show the feasibility of our approach and that is why we combined the four keywords and their corresponding images before extending this work in the future.

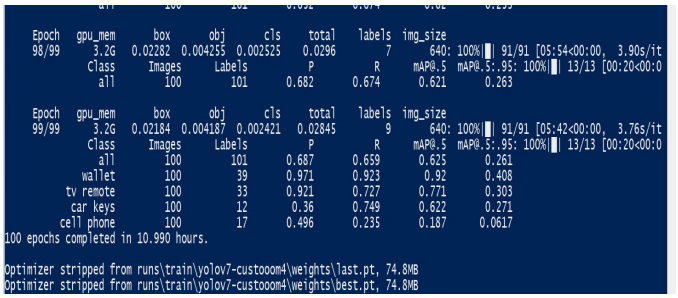


Fig. 3. Training details at the last epoch.

VI. Results and Discussions

A. Confidence Level

The different versions of Yolo come with advantages and drawbacks: the more advanced the detection model, the more data and time it requires to be trained. We considered version 7 to be appropriate for our paper as it offers much better performance, as explained in Section III. We also decided to use the base weight model of Yolov7 called ”Yolov7.pt”. As

 ISBN: ISBNFILL

Figure 4 shows, the confidence level curve for each searched keyword started very well. The various colors correspond to the different keywords we used in our experiments. It can be observed, for instance, that the light blue, which represents the wallet keyword, seemed to have benefited from much more images than the rest, thus making it overshadow the rest of the variables. The dark blue color represents the average of all variables. On average, the wallet confidence level curve performed much better at around 90 percent. For some reason, the cell phone, represented by the curve in red, exhibited a poor performance.

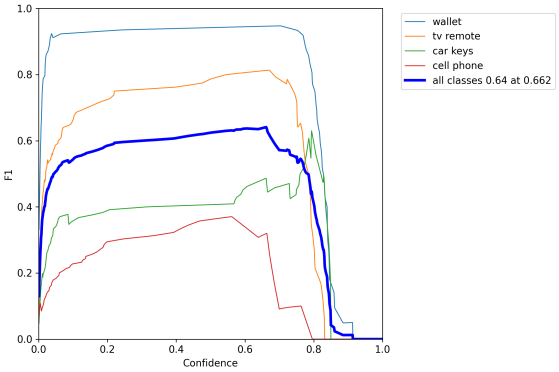


Fig. 4. Confidence level curve.

B. Precision and Confidence Level

As one of the goals of this paper was to improve the performance, we ran several rounds of training with different numbers of epochs in order to ensure we achieve the highest precision possible. As shown in Figure 5, the last training round showed high levels of prediction for the average variable of all keywords, which was on the rise, and overall most variables hit the 90% accuracy of precision level. This ultimately means that the model was able to see an object and make an accurate prediction of what it might be. Also, the model predicted that there are 90 car key images in the dataset. However, we had 100, then it can be concluded that the model was 90% precise with regard to that keyword.

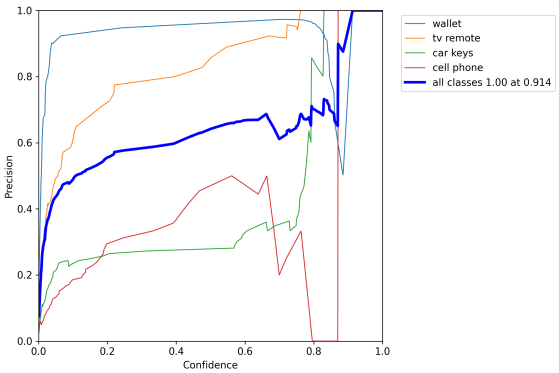


Fig. 5. Precision vs confidence.

C. Recall and precision

The recall is defined to represent the correlation between the true positives and the total number of predictions — the better the recall, the better the model. False positives and false negatives can be an issue in the output of the model, such as wrong labelling of some images. Therefore, the better the correlation between the precision and the recall, the better the model is. In Figure 6, we observed a big gap between the variables. This translates to the need to run more training with much more images from different angles and heights as the model is sensitive to the data it is fed with.

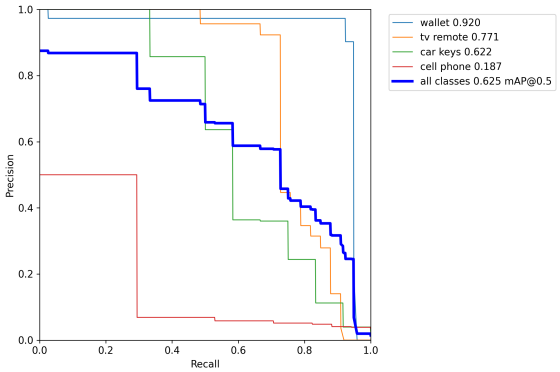


Fig. 6. Precision and recall curve.

D. Confusion matrix

Figure 7 shows the confusion matrix of our model, which is a method that is commonly used in the classification process. This matrix also shows the difference between what the model predicted versus what the real object being predicted actually is [9]. For instance, if the model predicts a tree, and in reality the object is a tree, then we call that a true positive. If the model predicts that the object is not a tree and, while the object is not a tree indeed, we call that true negative.

On the other hand, if the model predicts that there is a tree when it is not, that is called a false positive. When the model predicts no tree when there is one, that is called a false negative. In our case, the model did very well predicting the wallet variable. Indeed, it predicted the wallet with 95% accuracy. Regarding the tv remote, it reached up to 89% prediction accuracy. The cell phone keyword received a 71% accuracy in our model prediction.

E. Summary of the results

At the end of the 10 hours of training, we received a brief summary of the newly-trained model. Figure 8 shows the general summary of the model's performance. The new information is about the classification performance, the fact that the model recognized an object for what it is. The predictions in the training are shown in the first row and the validation is shown in the second row.

The resulting images of the model, shown in Figure 9, are satisfying in the sense that we accurately see a bounding box

 ISBN: ISBNFILL

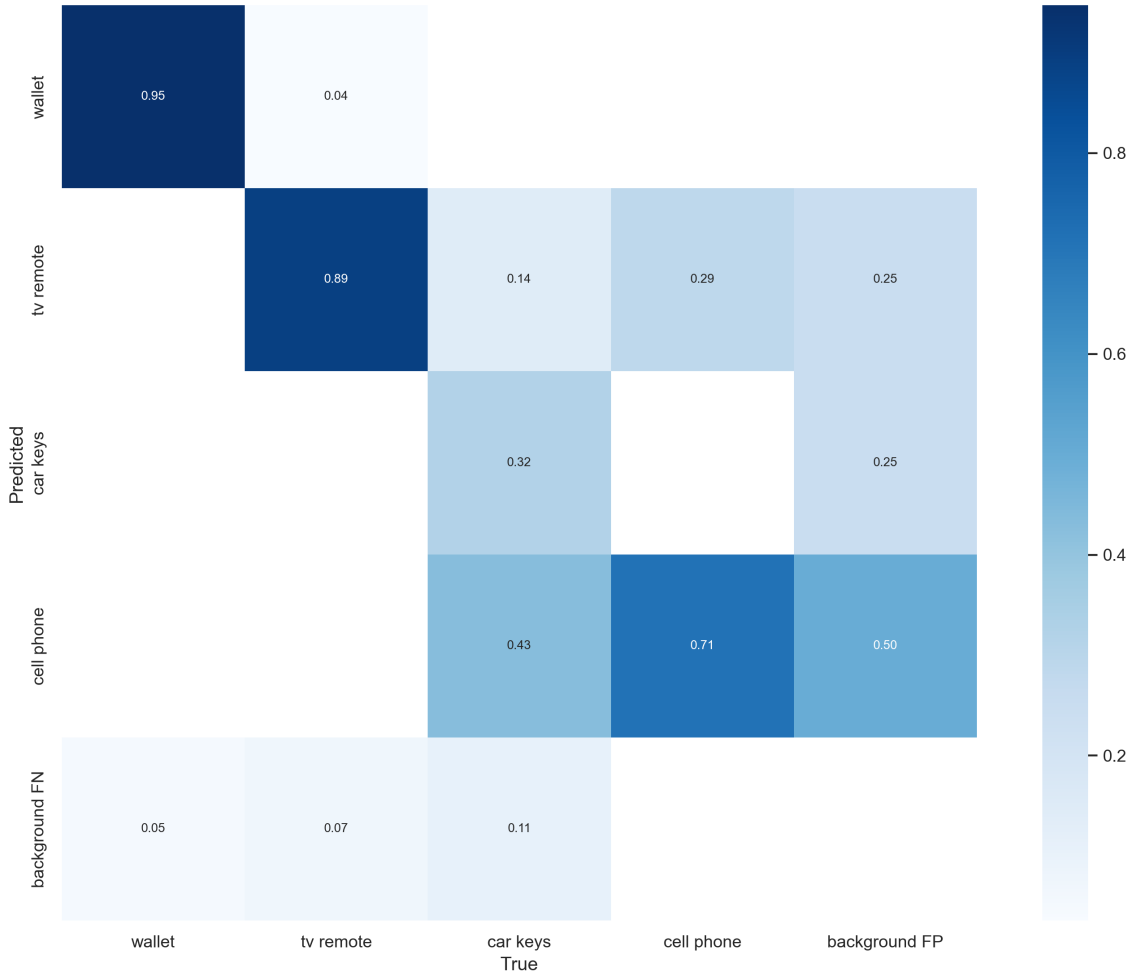


Fig. 7. Confusion matrix.

created around the objects of interest. The accuracy of the labels can be improved with more training and much more data to be fed into the model. The goal is to get an accurate distinction of the objection we might have lost. Therefore, when we place the four variables among many other objects, the model would be able to create a box with the objects we are looking for. Moreover, for a human eye that can recognize and categorize multiple objects instantly, we can say that the experiments that we conducted show the approach have promising potential of success at larger scale.

VII. Conclusion and Future work

In this paper, we leveraged Yolo object detection model along with Google voice recognition in order to help people find misplaced items in their household. Actually, we had more experience working with the previous versions of Yolo, but we had to figure out how to improve on them with regards to needed data for training, and much more computing power to sustain the high cost of processing. That is why we chose Yolov7 thanks to its better performance when compared to the previous versions. Also, we chose the base weight to train our custom model and fed the training model with 461 images in total before observing the outcome of the training phase. When the training was completed, the results showed a good potential of success. The objects that the training seemed to recognize best were the wallet and cell phone. That can be explained by the fact that both had a lot of images in training. The outcome of the experiments we conducted was a model that accurately creates a bounding box around the interested objects among many other uninterested objects and saves that in video evidence in a pre-determined folder. This can be considered a success because humans can easily recognize objects as long as they know where they might be.

In the future, we plan to train separate models in the cloud according to their specific object in order to reduce the training time. Also, to improve the performance, the voice input would choose one object linked with its model. In addition, we plan to feed the model with a much higher number of images per model per object from all types of angles and distances to guarantee maximum accuracy. Finally, we also plan to explore how we can extend this work to detect health risks for elderly

 ISBN: ISBNFILL

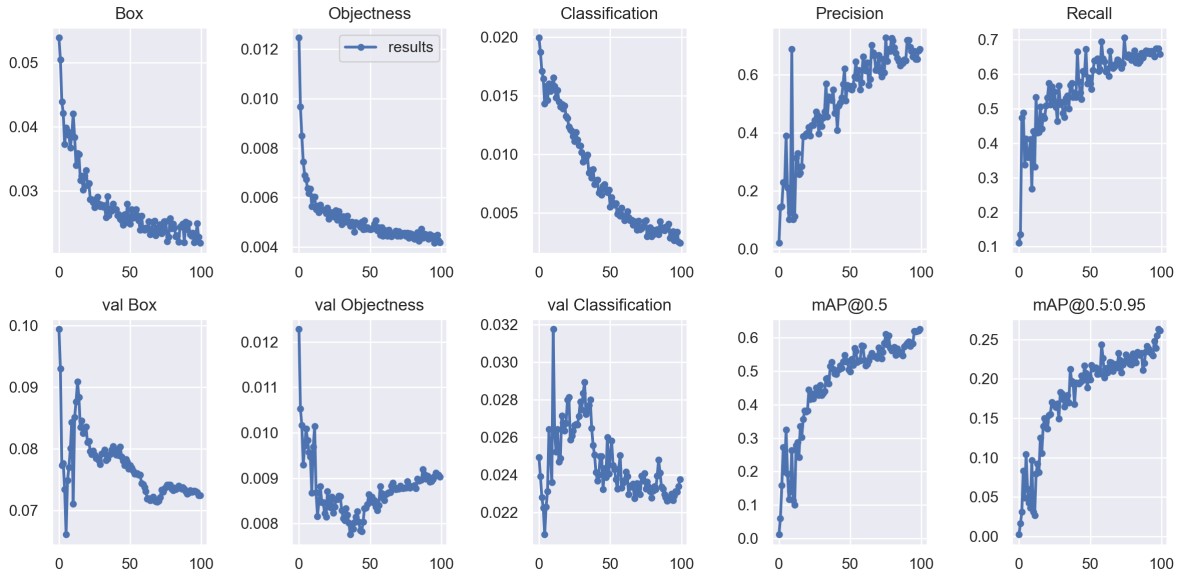


Fig. 8. Results.

Fig. 9. Training output.

people such as falling.

ACKNOWLEDGMENT

This work was funded by the National Science Foundation (NSF) award 2011689.

REFERENCES

[1] J. M. Helm et al., "Machine learning and artificial intelligence: definitions, applications, and future directions". Current reviews in musculoskeletal medicine, 13(1), pp. 69-76.

[2] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity". The bulletin of mathematical biophysics, 1943 pp.115-133.

[3] Y. T. Liu, "The Ultimate Guide to Video Object Detection", Toward Data Science available at https://towardsdatascience.com/ug-vod-the-ultimate-guide-to-video-object-detection-816a76073aef, [accessed January 2023].

[4] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions" Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection". The 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788.

[6] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "YOLOv7: Trainable bag of freebies sets new state of the art for real-time object detectors", arXiv preprint arXiv:2207.02696, July 2022.

[7] H. Wen, F. Dai, and Y. Yuan, "A Study of YOLO Algorithm for Target Detection". The 2021 International Conference on Artificial Life and Robotics (ICAROB2021), January 2021, pp.70-73.

[8] Y. K. Wong, Implementation of YoloV7 on GitHub. available at https://github.com/WongKinYiu/yolov7, [accessed March 2023].

[9] S. Visa, B. Ramsay, A. L. Ralescu, and E. Van Der Knaap, "Confusion matrix-based feature selection". The Twenty Second Midwest Artificial Intelligence and Cognitive Science Conference, 2011, pp. 120-127.

[10] M. R. R. Karmarkar, and V. N. Honmane, "Object Detection System For The Blind With Voice Guidance". The International Journal of Engineering Applied Sciences and Technology, 2021, pp. 2455-2143.

[11] A. P. Jana, and A. Biswas, "Yolo based Detection and Classification of Objects in video records". The 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), May 2018, pp. 2448-2452.

[12] Sr. D. Zhang et al., Y. "Using YOLO-based pedestrian detection for monitoring UAV". The Tenth International Conference on Graphics and Image Processing, 2018 Vol. 11069, pp. 1141-1145.

[13] K. Priyankan and & T. G. I. Fernando, "Mobile Application to Identify Fish Species Using YOLO and Convolutional Neural Networks". The International Conference on Sustainable Expert Systems, 2021, pp. 303-317.

 ISBN: ISBNFILL

Using Historical Social Media Retrieved Trust Attributes to Help Distinguishing Trustworthy Users

Teng-Chieh Huang
The University of Texas at Austin
The Center for Identity
Austin, Texas, USA
e-mail: tengchieh@utexas.edu

K. Suzanne Barber
The University of Texas at Austin
The Center for Identity
Austin, Texas, USA
e-mail: sbarber@identity.utexas.edu

***Abstract*—With the penetration of social media across the world, the information generated by the users has increased exponentially. The wisdom of crowds can now be easily accessed from the Internet. The problem is, how to correctly interpret the true public opinion without distorting it? Considering the spam or malicious users hidden in the social media spreading misinformation and disinformation, the solution might not be trivial. Some previous work uses quantity accumulation trying to mitigate the influence of bad users. More recent works add machine learning techniques to help with the correct judgment. However, the importance of the individual user - the actual person who is behind the screen, does not attract the attention it deserves. In this work, focusing on the history of user behavior, we provide a different angle to understand the connection between the credibility of social media users and the trustworthiness of their virtual representatives. By analyzing Twitter data from November 2017 to November 2021 which contains three types of users (typical, topic-related, and expert) on two target domains (politics and finance), we can gain deeper insights on the social media users and their trustworthiness.**

***Index Terms*—social media, trust, trust attributes, time series forecast, random forests classification.**

I. Introduction

The invention of the Internet and the emergence of social media has exponentially increased the *quantity* of sources a person can contact, but some of these sources may not provide high *quality* information. Social media data is noisy, loaded with contaminated data, advertisements, and scams. The presence of misinformation and disinformation on social media is a rising concern. For those who rely on social media for information, be it for personal use or modern-day analysis of social media data, it is important to know who they can trust. There exist users who abuse the capability of social media by spreading spam, fake news, or biased information. These malicious users not only fool their audience, but may also distort the information retrieved from social media for analytical or prediction purposes.

We aim to find trustworthy information on social media by finding trustworthy users. Specifically, this research addresses how to recommend trustworthy information from specific information sources using an innovative method of combining interpersonal trust on the Internet and classification algorithms for various target domains. We improve the existing and develop new *trust filters*–measurable properties of social media user profile, behavior, connections, posts, etc.–used to identify trustworthy users.

We take a comprehensive list of trust attributes (i.e., properties of a social media user account or posts such as the number of tweets or the number of followers) as potential trust filters from previous work. Further, we consider some established trust filters from previous work.

Using these potential trust filters, we develop a novel, domain-independent method to distinguish three types of social media (specifically Twitter) users from one another: typical users, domain-related users, and experts. We measure trust filters and see how they can rate the three types of users. The value of the trust filter for typical users is the average value for all the users. Domain-related users (e.g., stock-related users) are those that tweet with a set of domain-related handles (e.g., @a_stock_ticker). Experts are real world experts of the field, who we extract from reputable journalistic sources outside social media.

By (1) observing the distribution of the historical trust filter values across user types and domains, and studying the correlation between trust filters, (2) using random forest for time series forecasting to predict new trust filter values, and (3) applying the random forest classifier to compare their performance in distinguishing the types of users, we identify which trust attributes can best distinguish expert, domain-related, and typical users from one another. Most importantly, we keep our work independent of the subject domain by performing the same set of experiments in the finance as well as the politics domain. Moreover, this model can be applied to other target domains, such as health and entertainment.

The rest of the paper is structured as follows. Section II discusses the research that uses social media as a predicting tool in the focus domains of finance and politics. Section III describes how to categorize user groups, data retrieval, and the implementation of the trustworthy users differentiation. Section IV shows the effectiveness of the proposed method with different combinations. Section V concludes the contribution of this work and indicates possible future work.

 ISBN: ISBNFILL

II. Background

A. Social media predictions in the finance domain

There exists a body of research exploring the power of crowd wisdom as an indicator or a predictor of certain phenomena. In the finance domain, for instance, high social media coverage at the stock level can predict high subsequent return volatility and trading activity [1]. The extracted sentiments of social media users may be used to predict the stock market [2]. Techniques such as text mining [3] and machine learning are frequently used to enhance the predicting power of social media.

For the sake of completeness, we cover some of the most notable related work that seek to predict stock markets, our example target domain, albeit with very different methods. A widely cited work [4] demonstrated how the Twitter mood, in general, predicts the stock market closing values with high accuracy. Nassirtoussi et al. [5] tried to predict foreign exchange markets based on the text of breaking financial news headlines. Dimpfl et al. [6] studied the dynamics of stock market volatility using Internet search queries and found that high stock market volatility can follow high volumes of Internet searches. Nguyen et al. [7] performed stock market prediction based on social media analysis as well. Instead of taking all sentiments into account, they considered only the sentiments of specific topics of the company to predict stock price movement to increase the forecast accuracy. Oliveira et al. [8] sought to predict stock markets through Twitter posts. Among all the factors, they found sentiment and posting volume to be particularly important for the forecasting of the Standard and Poor's 500 (S&P 500) index.

B. Social media predictions in the politics domain

The politics domain, particularly elections, is another hotspot for testing the prediction power of social media. Ever since Twitter started becoming an essential online social platform, researchers have been exploring its potential of predicting the election outcome [9]. The 2016 presidential election of the United States, in particular, brought Twitter under the spotlight of public attention. Compared to the Clinton campaign's strategy, the Trump campaign's style in social media points towards de-professionalization and even amateurism as a counter trend in political communication [10]. Moreover, fake news spreading on Twitter [11], social bots distorting public opinion [12], and even Russian interference [13] all played roles on Twitter during the 2016 presidential election. Therefore, it is critical to distinguish real users from fraudulent or malicious sources before utilizing social media as a prediction tool.

The work in [14] showed the distinction among different types of user groups: typical users, target domain-related users and experts of specific domains based on the trust attributes. This work presented the possibility of utilizing social media as a tool to distinguish the more experienced and possibly credible users. This work also suggested some tweet-derived attributes could become promising candidates to differentiate the trustworthy users in specific target domains.

III. Methodology

A. User group categorization

For both the finance and politics domain, the users are categorized based on their level of expertise, involvement, and reputation to the specific target domain. A more detailed definition of the three groups of Twitter users for each target domain is specified below.

1) Finance domain:

- Typical users: The 1% random sampling of all Twitter users (Also known as the Spritzer version of tweets), which stands for the baseline among all groups. In this work, 3000 typical users were randomly selected from the Spritzer version of tweets.
- Stock-related users: The users who posted at least one tweet with a reference to the symbols of the stock market companies, such as $AAPL or $TSLA, during the sampling period. This user group represents the users who might have interest in the stock market, while they may or may not be financial experts. Among all stock-related users collected in 2020-2021 tweet data, 1000 of them were randomly selected for this work.
- Financial experts: This group of users is retrieved from well-known online articles which recommended the top-notch financial experts to follow on Twitter [15]–[17]. These lists are compiled by their authors or by Wall Street analysts and journalists, who are traditionally considered financial authorities. There are 180 recommended financial experts and all of them were included in this work.

2) Politics domain: To keep the consistency between different target domains, the definitions of three groups of Twitter users were similar to the ones defined in the finance domain, which are shown below.

- Typical users: The same 3000 users set as the finance domain.
- Politics-related users: The users who posted tweets with hashtags of the candidates, during the sampling period. 1000 of them were selected in the same manner as stock-related users.
- Political experts: Following the same concept as the financial experts, the political experts were recommended by online articles regarding the specific expertise [18]–[24]. There are 119 recommended political experts and all of them were included in this work.

B. Twitter data retrieval

The list of typical users was randomly selected from the "Spritzer" version of tweets. The lists of two domain-related (finance and politics) users were collected from the same dataset with previous defined requirements. The lists for experts were recommended by the sources mentioned previously. After the list was established, all tweets posted by the listed users were then downloaded using a Python library named "tweepy." The sampling period was from November 1st 2017 to October 31st 2021. The tweets were then divided into 8

 ISBN: ISBNFILL

segments, each of which contains half-year-long tweets, and trust attributes which will be introduced in the next section.

C. Tweet-retrieved trust attributes

We derive a set of trust attributes retrieved from tweets for each user and use the trust attributes as indicators of the trustworthiness of users. Here, an attribute refers to a user, text, or social connection information of a single social media user. 16 trust attributes (shown in Table I) are selected based on the analysis of previous research [14] which include tweets content, Twitter user information, and social network structure. The trust attributes we chose are neutral, suitable for universal purpose across all domains and do not require any specialized retrieval technique to calculate the attribute for a user. The *expertise_score* is determined by the "keywords" of the corresponding target domain. In the finance domain, the keywords are defined as stock symbols, which is the same as the definition to retrieve stock-related users. In the politics domain, the keywords are sets of frequently used election-related hashtags among political-related users. Those tweets containing keywords are counted as domain-related tweets.

Trust attributes were calculated semiannually from each user's tweets. Therefore, at most 8 sets (2017$\sim$2021 and twice per year) of trust attributes could be possessed by a single user from the time frame we sampled. The time series of trust attributes represents the change of user behavior in time, which means that by analyzing and understanding how trust attributes change with time, we should be able to forecast the future behavior of each user.

D. Time series analysis

To predict the future trust attributes based on historical data, first we must identify the requirements. The forecast model should be able to forecast multiple trust attributes from multiple historical trust attributes. Here, we assumed trust attributes could be influenced by other trust attributes, which should be self-explanatory. Attributes like Avg_len_tweet and Avg_n_word_tweet are highly related to each other, as are Len_per_word. Many other trust attributes may have implicit influence on others, like Followers_count and Retweet_ratio. However, it is extremely difficult for the classic time series forecast to train a single model that can predict several targets. Not to mention that the short length of data history and huge quantity of users which will require gigantic computation efforts, might provide only mediocre predicting accuracy. Slicing more time frames to the same period of time might be a solution, but since the sampling database was "Spritzer" version as explained in Section III-B, the tweets generated from a single user in a short period of time could be sparse. If the trust attributes were calculated quarterly or monthly, the majority of users would have merely 1, 2, or even no tweets in most of the time slots.

To mitigate the above-mentioned problems, instead of utilizing traditional time series forecasting methods, this work applied the Random Forests (RF) time series model. There are several advantages for using the RF time series model than other methods of time series forecasting.

- Handles non-linear time series data: many popular classic time series forecasting models such as AutoRegressive Integrated Moving Average (ARIMA) assume linear relationships between variables [28]. However, in the real-world case, many data sets are non-linear and hence require complicated preprocessing of the data to guarantee linearity. This increases the overall complexity of the computation, especially when more variables are considered. On the other hand, RF can handle both linear and non-linear variables well.
- Does not require long historical data: the RF model does not require long continuous time series data, since it only needs the length of predefined lag variables, which is flexible. The lag was set to be 1 in the experiment.
- Decision trees are a great fit to simulate individual user's behavior: RF is an ensemble learning method. For each individual user, it is not essential for the model to perfectly forecast trust attributes. What we need is the forecast of trust attributes to have high accuracy macroscopically, i.e., the forecast attributes have high accuracy in each user group.
- Generates multiple outputs with only one trained model: Python *sklearn* library supports multiple-outputs fitting in a single model. This is extremely beneficial to this work because it is not just one variable, but 16 variables are being predicted. By training only one model, the computational efforts can be hugely decreased.

To train the RF time series forecasting model, we first converted time series data into supervised learning data. Only users with at least 4 recorded active postings out of 8 time slots were included. There are 902 users in the finance domain (547 typical users, 260 stock-related users, and 95 financial experts) and 967 users in the politics domain (547 typical users, 376 politics-related users, and 44 political experts). The last time slot was used to test the overall prediction accuracy. Therefore, with lag variable setting to be 1, each user could have at least 2 training sets. The number of estimators (decision trees) was 5000. The result will be shown in Section IV.

E. Random forests classification

Different from random forests time series forecast in the previous section, the random forest classification here is used to test the capability of the RF time series forecast to enhance the classification accuracy. To avoid ambiguity, the following section will use the acronym RF-T for random forests time series model and will refer to random forests classification model used to distinguish user groups as RF-C.

Trust attributes were tested having the capability to distinguish between user groups [14]. In this paper, we further tested their capability by training RF-C with different combinations of various historical data and the forecast data made by RF-T. The RF-C models were trained by the following datasets.

- T_{-1}: One time step before the latest set of trust attributes T.

 ISBN: ISBNFILL

TABLE I
THE DEFINITION OF TRUST ATTRIBUTES.

Trust attributes	Definition	From previous work
Expertise_score	Ratio of tweets which is domain related	[25]
Statuses_count	Total number of posted tweets in user history	[25], [26]
Followers_count	Number of followers of a user	[25]–[27]
Friends_count	Number of friends of a user	[26], [27], [27]
Avg_len_tweet	Average tweet length in characters per tweet	[26]
Avg_n_word_tweet	Average number of words per tweet	[26]
Avg_hashtag	Average number of hashtag symbols per tweet	[25]–[27]
Avg_tweet_URL	Proportion of tweets containing URLs	[25]–[27]
Avg_tweet_question	Proportion of tweets containing "?"	[26]
Avg_tweet_exclamation	Proportion of tweets containing "!"	[25], [26]
Avg_tweet_uppercase	Proportion of tweets composed by upper-case letters	[26]
Retweet_ratio	Proportion of retweets in user's posts	[26]
Len_per_word	Average length of words among all tweets	[25]
Avg_retweet	Average number of retweets a user received per tweet	
Favorite_per_retweet	Average number of favorites received per tweet	
Len_per_word	Average length of characters per word	

- $T_{-1 \rightarrow -3}$: 1, 2, and 3 time steps before the latest set of trust attributes T.
- $T_{-1} + T_{Forecast}$: T-1 and the forecast of T .
- $T_{-1 \rightarrow -3} + T_{Forecast}$: 1, 2, and 3 time steps before T and the forecast of T.

The latest set of trust attributes T were used as testing sets to verify the accuracy of RF-C in each combination.

IV. EXPERIMENT RESULTS

A. Random forests time series forecast

Based on historical trust attributes, RF-T can forecast the future trust attributes of users with great accuracy. For trust attributes having high time-dependency, such as Followers_count (Fig. 1), the forecast values are close to actual values. As for trust attributes with lower time-dependency, such as Avg_len_tweet (Fig. 2), the predicting error tends to be more.

For a baseline comparison, we applied trust attributes one time step before T, which is T_{-1}, as a simplified guess of T. Table II shows Root Mean Square Error (RMSE) of all trust attributes when using T_{-1} and $T_{Forecast}$ as a comparison to T. For both domains, $T_{Forecast}$ has a better RMSE than T_{-1}. This shows the probability of using RF-T to forecast trust attributes.

B. Use time series forecast to enhance social media user classification

Fig. 3 to Fig. 6 highlight the quality of trustworthiness forecast predictions offered by the proposed RF-T method for the finance and the political domains using four training sets. These experimental results report on (1) accuracy measured by the ratio of correct predictions to total predictions and (2) "Area Under the receiver operating characteristic Curve" (AUC), measuring the overall quality of the proposed trust filters (attributes) in predicting classifications of user trustworthiness. A number of decision trees (estimators) were tested from 10 to 1000.

Considering the small difference in RMSEs in the finance domain between T_{-1} and $T_{Forecast}$, it is interesting to see how $T_{Forecast}$ improved the RF-C model, especially regarding AUC. It is also worth mentioning that no matter with or without adding $T_{Forecast}$, $T_{-1 \rightarrow -3}$ provides worse accuracy and AUC than T_{-1}. The reason might be that adding older data to train RF-C ends up disrupting the model because outdated data cannot correctly represent the latest trend.

Another observation is that for some cases, adding $T_{Forecast}$ to the training sets did not increase accuracy and AUC. Our explanation is that, when the accuracy or AUC is high enough, adding $T_{Forecast}$ could not enhance the performance since it is already saturated. If we change the classification targets to a different set of users instead of the same set of users, the potential of the enhancement might be better expressed.

Table III and Table IV provide the detailed values of accuracy and AUC when the estimator is 500. We can see that, once accuracy is over 0.95, there is no big difference regarding the addition of $T_{Forecast}$.

TABLE II
RMSE OF T_{-1} AND $T_{Forecast}$.

	T_{-1}	$T_{Forecast}$
Finance Domain	1391	1336
Politics Domain	11872	2848

TABLE III
ACCURACY AND AUC OF FOUR COMBINATIONS OF TRAINING SETS FOR THE FINANCE DOMAIN WHERE THE NUMBER OF ESTIMATORS IS 500.

	Accuracy	AUC
T_{-1}	0.9688	0.9078
$T_{-1} + T_{Forest}$	0.9673	0.9199
$T_{-1 \rightarrow -3}$	0.9247	0.8543
$T_{-1 \rightarrow -3} + T_{Forecast}$	0.9558	0.9089

 ISBN: ISBNFILL

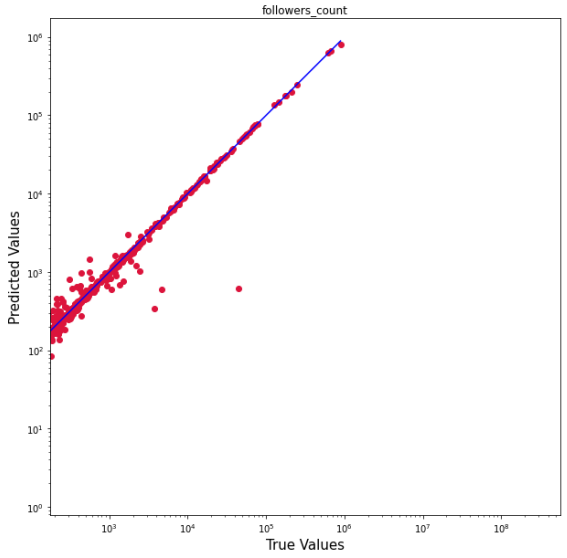


Fig. 1. Comparison of T and $T_{Forecast}$ in Followers_count in the finance domain.

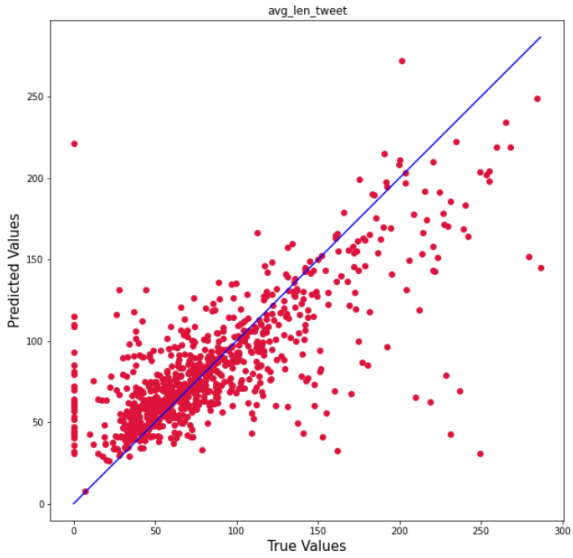


Fig. 2. Comparison of T and $T_{Forecast}$ in Avg_len_tweet in the finance domain.

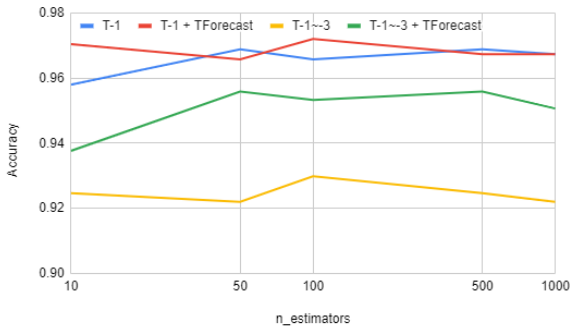


Fig. 3. Accuracy of four combinations of training sets for the finance domain. The number of estimators ranges from 10 to 1000.

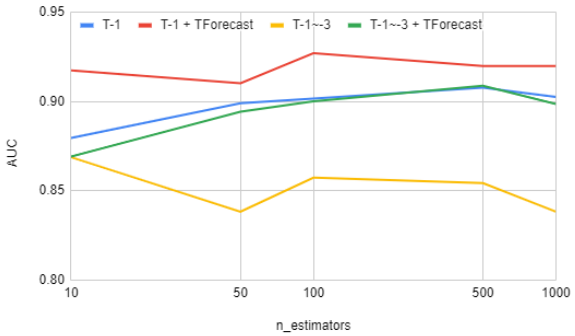


Fig. 4. AUC of four combinations of training sets for the finance domain. The number of estimators ranges from 10 to 1000.

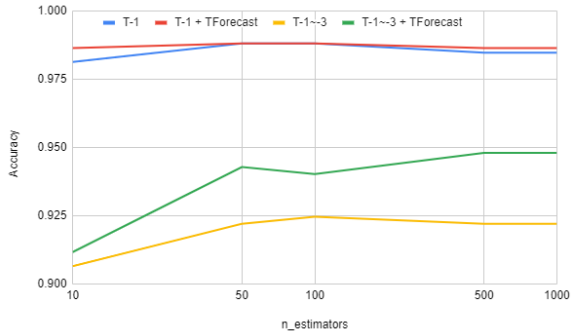


Fig. 5. Accuracy of four combinations of training sets for the politics domain. The number of estimators ranges from 10 to 1000.

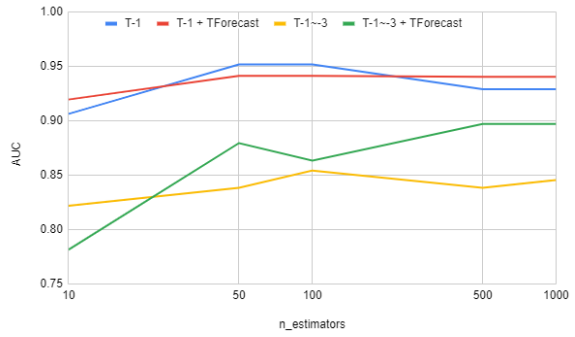


Fig. 6. AUC of four combinations of training sets for the politics domain. The number of estimators ranges from 10 to 1000.

TABLE IV
ACCURACY AND AUC OF FOUR COMBINATIONS OF TRAINING SETS FOR THE POLITICS DOMAIN WHERE THE NUMBER OF ESTIMATORS IS 500.

	Accuracy	AUC
T_{-1}	0.9848	0.9291
$T_{-1} + T_{Forest}$	0.9865	0.9404
$T_{-1 \rightarrow -3}$	0.9221	0.8383
$T_{-1 \rightarrow -3} + T_{Forecast}$	0.9481	0.8970

 ISBN: ISBNFILL

V. Conclusion

This work concentrated on developing and evaluating trust filters, measurable trust attributes of social media users that may be able to predict their level of trustworthiness. In this paper, we investigated two target domains, politics and finance, on Twitter. We split social media users into three different groups: typical, domain-related, and expert users. The list of experts was distilled from reputable online sources outside social media. A list of trust attributes was selected, shown to be effective by previous work, as proposed trust filters. We measured the value of these established and proposed trust attributes for Twitter users and generated the distribution of each trust attribute for each of the three user types. The random forest regressor for time series forecast (RF-T) is applied to provide better direction of distinguishing users. We applied the random forests classification algorithm to gauge the effectiveness of trust filters in classifying user types. The results show that, with the addition of RF-T predicted trust attribute values, there is a marked improvement in prediction accuracy and the ability to predict the classification of a user's trustworthiness (Area under the receiver operating characteristic curve, AUC). Our work paves the way for improving the quality of information extracted from social media by focusing on users' trustworthiness and increasing the reliability and utility of this data to explain phenomena, help with decision making, and even predict trends. One possible future work might be using the proposed method to find out more trustworthy users, and extracting users' opinions to make a prediction on certain domains. By comparing the predicting power with other user sets, we can appreciate the effectiveness of this method.

References

[1] P. Jiao, A. Veiga, and A. Walther, "Social media, news media and the stock market," *Journal of Economic Behavior & Organization*, vol. 176, pp. 63–90, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167268120300676

[2] C. Liang, L. Tang, Y. Li, and Y. Wei, "Which sentiment index is more informative to forecast stock market volatility? evidence from china," *International Review of Financial Analysis*, vol. 71, p. 101552, 2020.

[3] A. Sun, M. Lachanski, and F. J. Fabozzi, "Trade the tweet: Social media text mining and sparse matrix factorization for stock market prediction," *International Review of Financial Analysis*, vol. 48, pp. 272–281, 2016.

[4] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.

[5] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah, and D. C. L. Ngo, "Text mining for market prediction: A systematic review," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7653–7670, 2014.

[6] T. Dimpfl and S. Jank, "Can internet search queries help to predict stock market volatility?" *European Financial Management*, vol. 22, no. 2, pp. 171–192, 2016.

[7] T. H. Nguyen, K. Shirai, and J. Velcin, "Sentiment analysis on social media for stock movement prediction," *Expert Systems with Applications*, vol. 42, no. 24, pp. 9603–9611, 2015.

[8] N. Oliveira, P. Cortez, and N. Areal, "The impact of microblogging data for stock market prediction: using twitter to predict returns, volatility, trading volume and survey sentiment indices," *Expert Systems with Applications*, vol. 73, pp. 125–144, 2017.

[9] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment." *ICWSM*, vol. 10, no. 1, pp. 178–185, 2010.

[10] G. Enli, "Twitter as arena for the authentic outsider: exploring the social media campaigns of trump and clinton in the 2016 us presidential election," *European journal of communication*, vol. 32, no. 1, pp. 50–61, 2017.

[11] N. Grinberg, K. Joseph, L. Friedland, B. Swire-Thompson, and D. Lazer, "Fake news on twitter during the 2016 us presidential election," *Science*, vol. 363, no. 6425, pp. 374–378, 2019.

[12] A. Bessi and E. Ferrara, "Social bots distort the 2016 us presidential election online discussion," *First monday*, vol. 21, no. 11-7, 2016.

[13] R. S. Mueller and M. W. A. Cat, "Report on the investigation into russian interference in the 2016 presidential election," 2019.

[14] T.-C. Huang, R. N. Zaeem, and K. S. Barber, "Identifying real-world credible experts in the financial domain," *Digital Threats: Research and Practice*, vol. 2, no. 2, pp. 1–14, 2021.

[15] S. Constable. (2015) "must-follow" twitter feeds on markets and economics. [Online]. Available: https://www.forbes.com/sites/simonconstable/2015/08/14/must-follow-twitter-feeds-on-markets-and-economics/

[16] J. Cummans. (2015) 100 insightful futures traders worth following on twitter. [Online]. Available: https://commodityhq.com/investor-resources/100-insightful-futures-traders-worth-following-on-twitter/

[17] L. Lopez and L. Shen. (2015) The 129 finance people you have to follow on twitter. [Online]. Available: https://www.businessinsider.com/117-finance-people-to-follow-on-twitter-2014-9

[18] S. Emmrich. (2020) The 9 twitter accounts to follow on election day. [Online]. Available: https://www.vogue.com/article/the-nine-best-twitter-accounts-to-follow-on-election-day

[19] J. J. Roberts. (2020) 14 twitter and instagram accounts you should follow for election news day. [Online]. Available: https://fortune.com/2020/11/03/election-night-2020-who-to-follow-twitter-instagram-accounts-follow-updates-trump-biden/

[20] M. Cruz. (2020) 5 journalists to follow on twitter: Politics & news edition. [Online]. Available: https://onepitch.co/blog/5-journalists-to-follow-on-twitter-news-and-politics-edition/

[21] M. Hawkins. (2020) The top 15 conservatives to follow on twitter. [Online]. Available: https://www.thoughtco.com/the-top-conservatives-to-follow-on-twitter-3303615

[22] J. Donatelli. (2017) The 25 must-follow twitter accounts for the anti-trump resistance. [Online]. Available: https://www.lamag.com/culturefiles/social-media-twitter-resistance/

[23] R. Adams. (2020) Top 50 us politics twitter accounts to follow. [Online]. Available: https://www.theguardian.com/world/richard-adams-blog/2010/oct/05/top-50-twitter-accounts-us-politics-election

[24] D. Byers. (2015) Twitter's most influential political journalists. [Online]. Available: https://www.politico.com/blogs/media/2015/04/twitters-most-influential-political-journalists-205510

[25] M. Gupta, P. Zhao, and J. Han, "Evaluating event credibility on twitter," in *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM, 2012, pp. 153–164.

[26] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on twitter," in *Proceedings of the 20th International Conference on World Wide Web*, ser. WWW '11. New York, NY, USA: ACM, 2011, pp. 675–684. [Online]. Available: http://doi.acm.org/10.1145/1963405.1963500

[27] E. J. Ruiz, V. Hristidis, C. Castillo, A. Gionis, and A. Jaimes, "Correlating financial time series with micro-blogging activity," in *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, ser. WSDM '12. New York, NY, USA: ACM, 2012, pp. 513–522. [Online]. Available: http://doi.acm.org/10.1145/2124295.2124358

[28] M. J. Kane, N. Price, M. Scotch, and P. Rabinowitz, "Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks," *BMC bioinformatics*, vol. 15, no. 1, pp. 1–9, 2014.

 ISBN: ISBNFILL

Intelligent Motion Planning in Human-Robot Collaboration Environments

Zahid Iqbal, Liliana Antão, Vítor H. Pinto, Gil Gonçalves

SYSTEC, Research Center for Systems and Technologies
DIGI2, Digital and Intelligent Industry Lab
Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
Email: {zahid, lpsantao, vitorpinto, gil} @fe.up.pt

***Abstract*—A robot needs to have adequate Artificial Intelligence (AI) support for operating as a human co-worker, making its behaviour flexible and autonomous. Applications and development methodology for collaborative robots (cobots) differ substantially from the norm of traditional robot applications marked by pre-programmed, repetitive tasks with well-defined behaviours and little or no autonomy on the part of the robot. We present a modular solution for application development for cobots. Intelligent workspace monitoring, motion planning, and re-planning make for essential solution components. Our solution incorporates safety by design, imparts the robotic arm a partly autonomous behaviour and is a step in the direction of collaborative interactions with the arm. We implement perception and planning as separate modules and demonstrate how these modules integrate. The main focus of the work is the motion planning component, developed in Robot Operating System (ROS) and MoveIt. The proposed framework can receive multiple goal points, monitor safety thresholds, and account for many humans in the workspace modelled as dynamic obstacles. In particular, we show path planning with obstacle avoidance and report some performance measurement results of different built-in planning algorithms.**

***Index Terms*—collaborative robots, motion planning, sampling-based planning, MoveIt.**

I. Introduction

Several industries employed robots to improve production volumes and acquire better precision and accuracy in the overall production process. Traditional industrial applications (between the 1960s through 1990s) used robots for simple repetitive tasks with well-defined, pre-programmed behaviours. From the 2000s onwards, the development of robotic technology is driven primarily by advancements in the industrial Internet of Things (IoT) sensors [1][2], industrial wireless communication protocols [3], and software, in particular, AI techniques such as Machine Learning (ML), to make robots more autonomous. McMorris [4] and International Federation of Robotics (IFR) [5] nicely outline important milestones in technology development for robots. Typically, a sequence of steps makes up an industrial process. Along the process, human-robot interaction happens at defined points, for instance, when loading or unloading items. Such cases present obvious hazards due to the size of the robotic arm and proximity of the human [6]. Typical industrial environments confine robots to separate operation spaces isolated from human workers and bring robot operation to a halt if a human enters this space [7][8], safety being the primary concern.

A notable transition aims to position robots as co-workers for humans. While robots have proven efficient with hazardous, unpleasant or repetitive jobs, a complex process calls for creative thinking, flexibility, decision making or adaptability where the human role becomes crucial. When placed side-by-side with humans, we need to equip robots with strong AI making their behaviour nearly as skilful and flexible as that of humans. We call this approach Human-Robot Collaboration (HRC) [9]. To that end, we have cobots or collaborative robots that include some integrated safety features, such as force-limited joints and smooth surfaces to minimise impact hazards. Computer vision or similar techniques [10][11] that detect the presence of humans in the environment make for an essential component of the application developed for cobots. Most works on collaborative interactions concern safety aspects. Rybski et al. [11] and Bdiwi et al. [12] identify zones based on human position in the co-space and choose a safety operation accordingly. The work in [10] turns a standard industrial robot into a human-safe platform. It uses Light Emitting Diode (LED) markers for actors in the workspace and thus poses limitations for scalability and the presence of arbitrary persons in the co-space. Safety being a baseline, our work further explores task distributions using hierarchical task models and is promising for large industrial settings.

We present a modular solution that incorporates safety by design, imparts the robotic arm a partly autonomous behaviour, and is a step towards collaborative interactions with the robotic arm. We develop the work, in particular, for Universal Robot manipulator UR5 [13]. Implementation is carried out within ROS [14], using the motion planning framework MoveIt [15]. We define distinctly *task planning* and *path planning*. We assume that target scenarios comprise specific task objectives that we can discretize in granular operations. A task planner automatically generates a graph (i.e., a hierarchical task model) in which nodes constitute sub-tasks or atomic operations [16]. Thus, it encapsulates the different operation sequences that

 ISBN: ISBNFILL

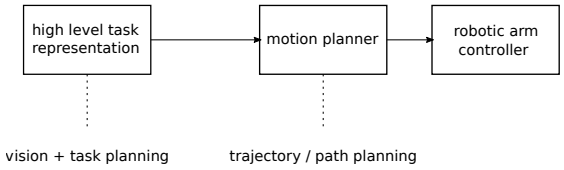


Fig. 1. Overall architecture

can fulfil a given task objective (root node of the graph). If the operation's actor is the robot, the task planner will input the specific coordinates that the path planner employs to compute a correct robot motion. For implementation, we divide the activity into two parts, namely, *perception* and *planning*. Perception provides the requisite vision information to the system about the robot workspace and the available objects, i.e., objects to manipulate, obstacles or humans. It already encapsulates the task planning referred to earlier. Path planning refers to robot motion planning that will take it to a target point while avoiding collision with workspace objects. We have decoupled the system making perception and path planning separate modules. The path planner uses the information provided by the perception module and guides the movement of the robotic arm to the target position. The work reported in this paper, in particular, encompasses motion planning. Having perception available as an independent module takes off the significant computational effort of the path planner. Conducting perception offline allows planners to process the workspace more efficiently. Fig. 1 shows a high level relationship of the main system components.

The presented work concerns industrial robotic arms or so-called manipulators. In particular, the proposed ideas apply to trajectory planning of `UR5`. The contributions of the paper are as follows:

(a) We provide an overview of the collaboration approach highlighting some aspects in which it differs from other works.
(b) We describe in detail the development steps of the motion planning component of our solution within ROS and MoveIt and provide some results that validate the implementation.

The rest of the paper is organized as follows. In the following section, we describe the tools and frameworks in which we have developed our solution. Section III presents the solution approach, i.e., how different modules integrate. In Section IV, we detail the development steps of our method. Section V presents some results from the implementation, in particular, the motion planning. Finally, in Section VI, we conclude the paper and indicate some future work directions.

II. Tool support for developing a motion planning application

In this section, we overview the main tools and framework used to develop the motion planning for the robotic arm.

A. Robot Operating System (ROS)

ROS provides a flexible platform for writing software for robots. Over the years, robot hardware, applications, and capabilities have significantly grown. A typical robotic system is complex, often combining several components that incorporate cameras, laser scanners, and odometry information. A functional robot system must include code for sensing, coordinate transforms, trajectory planning, navigation, hardware abstraction, control and more. For individual engineers, it becomes very challenging to cover each aspect of robot software development and then seamlessly integrate it into a complete system. ROS manages this complexity efficiently by providing tools, libraries, and open packages for standard robot functionality, allowing developers to reuse existing code from the available repositories and focus on their project-specific features. Such an approach lends fast and easy development of working prototypes for testing and experimentation in a simulated world, saving cost and time. Hardware interfaces allow running the program on the real robot. ROS supports C++ and Python for its API and Linux (Ubuntu) as its platform.

Within ROS, `roscore` is the system core providing a collection of nodes and programs. `roscore` contains three main functional module suits, which are `ROS Master`, `Parameter Server` and `rosout` logging node. The Master provides name registration and lookup for the rest of the nodes. The Parameter Server is a global key-value store through which nodes can share configuration information. `node` is the smallest unit of computation within ROS, i.e., an executable process. An application may distribute its functionality across several nodes. ROS maintains a peer-to-peer computation graph of ROS nodes which send and receive messages over named channels called `topics`. A package is the basic unit of ROS. It has the minimum structure to develop an application. It contains the application-specific configurations and launch files that allow running nodes from the same package and other packages. Concerning communication, there are different possibilities, i.e., topics, services, or actions. Topics follow a publish-subscribe model of asynchronous communication. Nodes publish to a topic to send a message and subscribe to it to receive a message; sender and receiver are decoupled. Service follows a request-response paradigm where communication is bi-directional and synchronized. The service client requests a service and the server responds to the request. Action is similar to service, however, it is used when the requested objective takes a long time to complete and intermediate feedback is necessary. The communication is asynchronous in this case (Fig. 2).

B. MoveIt

MoveIt is a set of software packages with specific capabilities for mobile manipulation, such as kinematics, motion planning and control, 3D perception and navigation. MoveIt runs on top of ROS and builds on the ROS messaging and build systems, and uses some of the common tools in ROS, e.g., the ROS Visualizer (RViz) and the Unified Robot

 ISBN: ISBNFILL

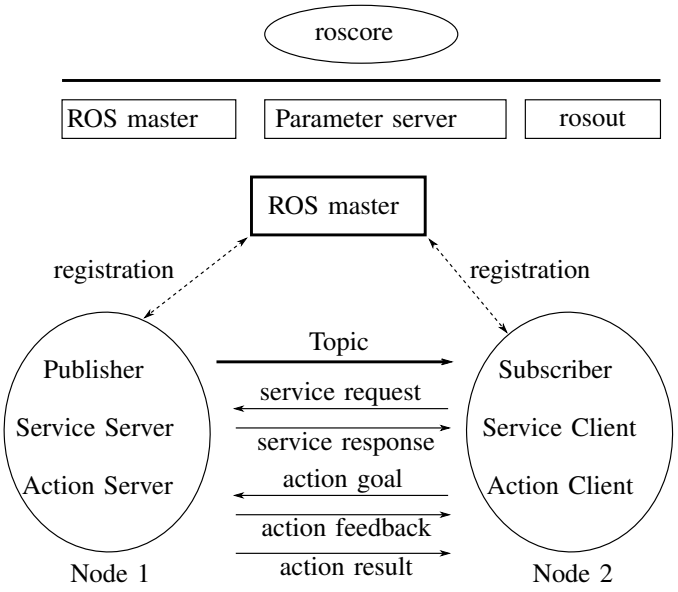


Fig. 2. ROS core communication system, and different message communication modes between nodes (adapted from [17]).

Description Form (URDF). It uses plugins for most of its functionality; motion planning (Open Motion Planning Library (OMPL) [18]), collision detection (default: Fast Collision Library (FCL) [19], and kinematics (default: OROCOS Kinematics and Dynamics Library (KDL) for forward and inverse kinematics for generic arms. A motion-planning framework such as MoveIt aims to lower the barrier of entry to robotic software. Additional to the underlying principle of code reuse and easy customisation as in ROS, MoveIt has specific design goals [20] primarily addressing the user-base without the breadth of knowledge to customise each toolchain with the right parameters or where time, effort or expertise needed to integrate different software components into the robot are considerable.

III. Proposed approach - an overview

Our solution to collaborative motion planning comprises three phases, namely *perception*, *pre-planning* and *planning* phases. The perception and pre-planning phases involve scene acquisition, voxelization of the workspace and generation of atomic operations. The planning phase is the focus of this work, which involves building collision-free paths between given start and goal positions. Fig. 3 presents the overall architecture of the motion-planning solution for collaborative environments.

The output of the perception phase is a voxel grid. A voxel represents a value on a regular grid in three-dimensional space, useful for many applications such as Dynamic Roadmaps [21] to create a mapping from an area in the workspace to states in the configuration space. For efficient path planning with obstacle avoidance and considering a collaboration scenario, an intelligent monitoring system is inevitable [22][23]. The present work uses the monitoring solution in [22]. We integrate the voxel-based grid with the motion planner. The voxel-based grid renders the entire scene of the collaborative environment as a grid composed of cubes with a given dimension, known as voxels. We can describe the granularity of the grid with the size of one side of the cube. The resolution can be set higher or lower by choosing the dimension of the unit cube in the grid, i.e., for a higher resolution, we choose a smaller size of the voxel, thus corresponding to more voxels in the grid, and vice versa. The pre-planning phase can produce voxel grids with varying degrees of information. A simple voxel grid is the occupancy voxel grid, which will set a voxel 1 or 0 depending if the corresponding workspace is free or occupied.

A. Mapping voxels to the system coordinates

A complementary framework loads the voxel grid into a 3d array within the planning solution in ROS and allows us to access the position of each voxel. Fig. 4 shows an overview of the approach. When the information is available, we segment regions of interest in the workspace, i.e., obstacles or humans. Corresponding voxels to such objects are occupied in the grid. With the distinct regions available, we can use specific algorithms to create bounding shapes around them, e.g., bounding boxes or spheres. ROS / MoveIt needs key dimensions, i.e., for a sphere it needs its radius and centre point, to render objects of interest in the simulation or real tests. Within MoveIt, we can calculate each point on the robotic arm and in the workspace in reference to the planning frame, which is the frame of the `base-link` of the arm, and its origin is at position $(0, 0, 0)$. In robotics, this frame is often referred to as the `world` frame. With this information, and knowing the dimensions of individual voxels and the hyper-cube that represents the complete voxel grid, we map the goal positions and obstacle positions from the voxel grid into the MoveIt so that we can test path planning with obstacle avoidance in real environments.

We define two structures, one representing a voxel-point vp in the grid VG. The other structure represents a point rp that we use to specify the position of any object within ROS, so-called real-point. The notion $vp_i(x)$ gives the value of x coordinate for ith voxel-point. Similarly, $rp_j(y)$ provides the value of y coordinate for a real-point j. Let sz denote the voxel size in the given VG, and let lx, ly, lz indicate the lower limits on the voxel grid. Then, we can find the value of the real coordinate points for a given voxel point, as shown in equations (2)-(4). Considering that coordinates are expressed in meters, we divide the final result by 100 in each case.

$$vp = (x, y, z) \tag{1}$$

$$rp_j(x) = \begin{cases} lx, & \text{if } vp_j(x) = 0 \\ \dfrac{lx + vp_j(x) \times sz}{100}, & \text{otherwise} \end{cases} \tag{2}$$

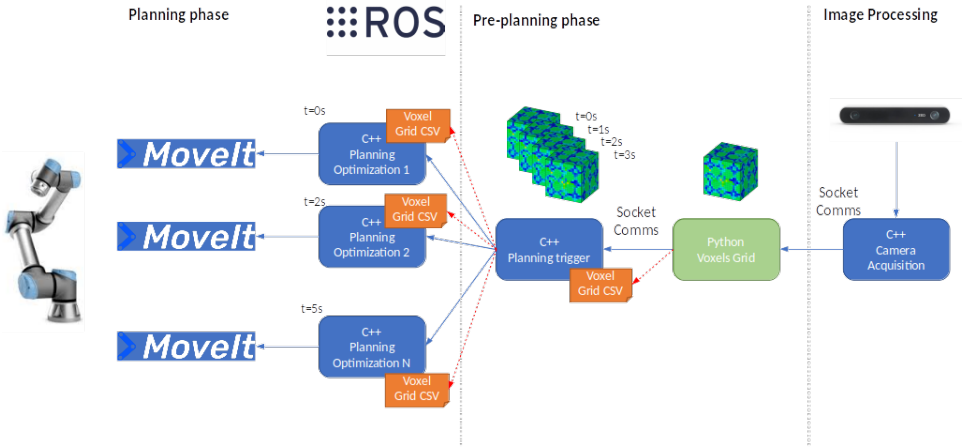


Fig. 3. Path planning solution architecture and module relationship.

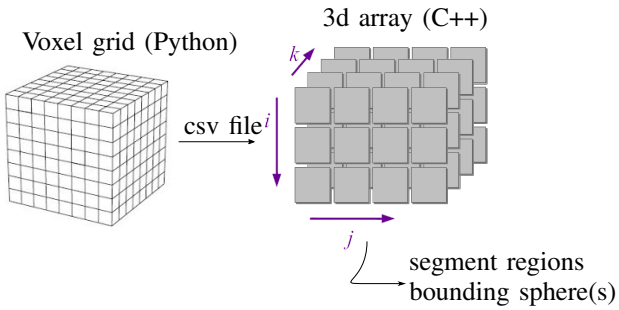


Fig. 4. An overview of the preliminary approach for integration of voxel grid within the planning solution.

$$rp_j(y) = \begin{cases} ly, & \text{if } vp_j(y) = 0 \\ \dfrac{ly + vp_j(y) \times sz}{100}, & \text{otherwise} \end{cases} \quad (3)$$

$$rp_j(z) = \begin{cases} lz, & \text{if } vp_j(z) = 0 \\ \dfrac{lz + vp_j(z) \times sz}{100}, & \text{otherwise} \end{cases} \quad (4)$$

In dynamic scenarios, the scene can change over time, and the robotic arm must be able to plan accordingly. We can input fresh voxel grids to the planning program with a certain frequency. The refresh rate of the voxel grid depends on the response time of the previous planning query. Currently, we are looking into solutions to address this issue.

IV. Development of the Motion Planning Solution

We detail the main steps required for manipulator control with ROS, including developing a model of the robot's physical structure, publishing coordinate transforms data and applying standard algorithms, such as path planning. We have used ROS industrial repositories of `UR5` [24], in particular, `ur_modern_driver` [25] to control the actual robotic arm.

A. Modeling the workcell with URDF

URDF is an XML format to describe the robot model in ROS. The principal components describing a robot model are the joints and links. The link component outlines the body by its physical aspects (dimensions, position of origin, colour etc.), and the joint describes the kinematic properties of the connection (axis of rotation, type of joint, connected links etc.). A joint connects two links, a parent link that precedes the joint and a child link that follows the joint. Such a topology makes a tree structure, where each link has precisely one parent, but can have multiple child nodes.

For the robot to move, we set the joint type as revolute and specify the axis around which the following child link will rotate. The visual tag in the urdf file lends a visual appearance to the respective element in the simulator. The visual representation of an element could be specified by primitive geometric shapes such as a box or a cylinder or by using carefully created meshes such as COLLAborative Design Activity (COLLADA) models. The `urdf` model of `UR5` used in this work uses meshes for defining visual components. In particular, the `urdf` model describes parts of the robot that do not change over time. Some examples include the topology, the relation between links and joints, and the complete link tree. Listing 1 shows part of the `urdf` model of the `UR5` robotic arm. The information in the `urdf` does not depend on the robot 3d position. For the ROS system to know about the robot model, a launch script loads the `urdf` file to the `Parameter Server`; `robot_description` is the name of the ROS parameter where the `urdf` file is stored on the `Parameter Server`. This format allows us to model other objects in the workspace, such as the table where the robotic arm is mounted.

```
<link name="base_link">
   <visual>
     <geometry>
       <mesh filename="package://ur_description/
            meshes/ur5/visual/base.dae"/>
     </geometry>
```

```
    <material name="LightGrey">
      <color rgba="0.7 0.7 0.7 1.0"/>
    </material>
  </visual>
  ...
 </link>
<joint name="shoulder_pan_joint" type="revolute">
  <parent link="base_link"/>
  <child link="shoulder_link"/>
  <origin rpy="0.0 0.0 0.0" xyz="0.0 0.0
      0.089159"/>
  <axis xyz="0 0 1"/>
 <limit effort="150.0" lower="-6.28318530718" upper
     ="6.28318530718" velocity="3.15"/>
  <dynamics damping="0.0" friction="0.0"/>
 </joint>
```

Listing 1. Part of UR5 urdf file.

We can use a command line tool check_urdf to check the model validity. This tool attempts to parse the file and either prints a description of the resulting kinematic chain or an error message. Fig. 5 shows the output after running check_urdf on the UR5 urdf model.

```
robot name is: myworkcell
---------- Successfully Parsed XML ---------------
root Link: world has 1 child(ren)
    child(1):  table
        child(1):  base_link
            child(1):  base
                child(1):  tool0_controller
            child(2):  shoulder_link
                child(1):  upper_arm_link
                    child(1):  forearm_link
                        child(1):  wrist_1_link
                            child(1):  wrist_2_link
                                child(1):  wrist_3_link
                                    child(1):  ee_link
                                    child(2):  tool0
```

Fig. 5. The link tree of UR5 manipulator mounted on a flat surface 'table'.

B. Creating the MoveIt pacakge based on URDF

The MoveIt Setup Assistant (MSA) provides a graphical user interface for configuring a robot with MoveIt. It can automatically set up the task pipeline for producing an initial configuration quickly. Its main functions comprise generating a Semantic Robot Description Format (SRDF) file, creating the collision matrix of the robot and defining the planning groups. The robot model URDF (section IV-A) is a prerequisite for the MoveIt setup assistant. The configurations set by the assistant include a self-collision matrix, planning group definitions, robot poses, end effector semantics and virtual and passive joints list. The first step of the Setup Assistant (SA) is the generation of a self-collision matrix for the robot used in future planning to speed up collision checking. This collision matrix encodes pairs of links on a robot that we can safely discard from collision checking due to the kinematic infeasibility of a collision. We have modelled a flat table surface into the work cell. The collision matrix, calculated at the initial step, will now find which links are in collision accounting for the table. Hence, the generated plans at runtime will only be made in the reachable workspace, making the planning faster and correct. The user can provide information on different motion planning aspects in a step-by-step process. Virtual joints attach the robot to the world. Planning groups semantically describe parts of the manipulator, i.e., what constitutes a gripper or which joints and links comprise an arm. MoveIt plans for a group considering only the joints that belong to that group. MSA allows adding certain fixed poses of the manipulator. We can define query positions as the initial and goal configurations of the manipulator, and end-effectors can be labelled. In the last step, the MSA generates several configurations and launch files that can be used inside a ROS package.

C. Planning and algorithms integration

On the planning side, we use MoveIt to integrate state-of-the-art sample-based motion planning algorithms in our solution. Using the motion planning APIs of MoveIt, the MoveGroupInterface, we have implemented and tested simple motions of the robotic arm. In particular, we can specify either pose goals or joint-space goals. Pose goals define the end-effector position in 3-d cartesian coordinates, whereas a joint-space goal identifies a distinct final configuration for the joints (given by individual joint angles). For both cases, we can plan the movement of the robotic arm to the desired goal. These tests have been done within the graphical simulator RViz and on the UR5 robotic arm. The Kinematics solver and planner algorithm are the main components of a motion-planning solution. MoveIt has built-in support for this functionality; it uses plugins for computing kinematics and path planning, Open Motion Planning Library (OMPL) [18] for motion planning, and Kinematics and Dynamics Library (KDL) for forward and inverse kinematics. In recent tests, we have used trac_ik inverse kinematics solver [26] that performs better in terms of its speed and success rate of finding a solution (in a set of 10000 random tests, it was 10% more successful and solving time was about half than KDL, on average [27]). In the OMPL library, we have random sample-based planners such as Probabilistic RoadMaps (PRM), Rapidly Exploring Random Trees (RRT), RRTConnect etc. We have tested with PRM and RRTs in our work. The framework allows us to add collision objects (obstacles) to the workspace. Collision objects are geometric primitives such as a box or a cylinder that we can easily define through their key dimensions and 3d position. In this case, the planned trajectory will avoid the obstacle or report failure when it cannot reach the target configuration.

We organize different nodes into a launch file to run the system. In Listing 2, we see an excerpt of the main ROS launch file of the system.

```
<launch>
  <rosparam command="load" file="$(find
      liu_moveit_config)/config/joint_names.yaml"/>
  <arg name="sim" default="true" />
  <arg name="robot_ip" unless="$(arg sim)" />
  <include file="$(find liu_moveit_config)/launch/
      planning_context.launch" >
    <arg name="load_robot_description" value="true"
        />
  </include>
  <group if="$(arg sim)">
```

 ISBN: ISBNFILL

```
  <include file="$(find industrial_robot_simulator)
      /launch/robot_interface_simulator.launch" />
 </group>
 <group unless="$(arg sim)">
  <include file="$(find ur_modern_driver)/launch/
      ur5_bringup_compatible.launch" >
    <arg name="robot_ip" value="$(arg robot_ip)"/>
  </include>
 </group>
 <group if="$(arg sim)">
 <node name="robot_state_publisher" pkg="
     robot_state_publisher" type="
     robot_state_publisher" />
 </group>

 ...
 ...

</launch>
```

Listing 2. `moveit_planning_execution.launch` file to bring up UR5.

V. Evaluations

In this section, we report the results from system execution. In particular, we present an analysis of the nodes' communication graph, the performance of different planning algorithms and an experiment with path planning and obstacle avoidance.

A. ROS computation graph

The program can be tested in RViz as well as on the robotic arm by simply setting `sim:=true` or `false` in the following command which launches the main script from Listing 2.

```
roslaunch liu_moveit_config
moveit_planning_execution.launch
sim:=false  robot_ip:=192.168.0.103
```

Besides Listing 2, we run other nodes, i.e., for publishing goal positions, or the coordinate points of the bounding boxes for dynamic obstacles such as humans. Fig. 6 shows the ROS graph of our application. The developed motion planning node `CMotionPlanner` receives a target pose on topic `sp1_pose`, whereas, it receives coordinates for the human bounding box on topic `chatter`. The primary node provided by MoveIt is the `move_group` node which serves as an integrator pulling individual components together. It loads three kinds of information from ROS `param server`: URDF of the work cell in `robot_description` parameter, SRDF of the work cell in `robot_description_semantic` parameter, and different configurations created by MSA for kinematics, trajectory control. `ur_driver` node publishes the real-time joint states on `/joint_states` and `robot_state_publisher` converts the `/joint_states` messages to corresponding `/tf` messages. It looks at the urdf of the robot and listens on the `joint_state` messages. It then calculates where each link of the robot is and then broadcasts it to the rest of the system on `tf`, which is subscribed by `move_group` too. Thus, it handles the common task of computing forward kinematics. Finally, the `move_group` node talks to the controller on the robot using the `FollowJointTrajectoryAction` interface by publishing a goal point on `follow_joint_trajectory/goal`.

TABLE I. PERFORMANCE OF DIFFERENT ALGORITHMS.

		pose goal		joint-space goal	
	algorithm	states	time (s)	states	time (s)
no obstacle	PRM	1412	5.002649	1417	5.005281
	PRMstar	868	5.001812	847	5.004916
	RRT	12	0.039855	11	0.042702
	RRTstar	2159	5.002191	2227	5.009980
	RRTconnect	4	0.034107	7	0.035261
obstacle	PRM	1552	5.003105	1568	5.002542
	PRMstar	992	5.009589	972	5.013361
	RRT	10	0.042280	26	0.062595
	RRTstar	1856	5.024425	1792	5.001743
	RRTconnect	5	0.028506	6	0.034078

B. Planning with obstacle

In this experiment, we use the PRMstar algorithm. We consider a pose goal given by the configuration vector $\{-0.1304, 0.43455, 0.19730, 1.41, -2.46, -4.88\}$ where the first three elements indicate the position and the last three indicate the orientation of the end-effector. We test the motion of the UR5 arm to this goal configuration in the presence and absence of an obstacle. The obstacle is a human bounding box with the following vertices' coordinates $(0.046, 0.95, 0.6)$, $(0.046, 0.48, 0.6)$, $(0.046, 0.95, -0.59)$, $(0.046, 0.48, -0.59)$, $(-0.03, 0.95, 0.6)$, $(-0.03, 0.48, 0.6)$, $(-0.03, 0.95, -0.59)$, $(-0.03, 0.48, -0.59)$. Fig. 7 shows the results for this test. Fig. 7 (a) depicts the case with no obstacle and start (yellow) and goal positions of the robotic arm. In Fig. 7 (b), we see the path trail that UR5 follows to reach the goal position in the absence of an obstacle. Fig. 7 (c) and (d) show the collision object in the workspace and the path trail. Notice that the arm follows a different path to avoid the obstacle. For case (b), the algorithm creates 808 roadmap states and the solution is found in 4.998913s. For cases (c) and (d), the algorithm creates 835 roadmap states and the solution is found in 5.014565s.

C. Performance of planning algorithms

We measure the performance of some sample-based motion-planning algorithms, namely PRM, PRMstar, RRT, RRTstar, and RRTconnect. Similar to the last experiment, we consider two cases, i.e., with and without an obstacle. Table I lists the results. RTConnect performs the best with the minimum number of states in all the experiments. RRTConnect uses two RRTs, one rooted at the source and another rooted at the goal point, and seeks to grow both trees until they get connected, thus finding the path. This methodology leads to faster solution time.

VI. Conclusions and Future work

Application development for robots is a challenging task with several dimensions, i.e., motion planning, scene acquisition, and control. A modular approach can cope with such

 ISBN: ISBNFILL

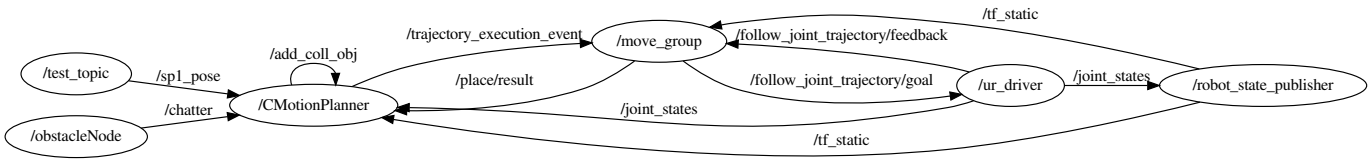


Fig. 6. `rqt_graph` view of the nodes and topics involved in actuating the model.

complexity. We showed how we could combine independently developed components namely perception and planning. In particular, we showed how we modelled the proposed solution architecture. ROS is a popular platform that offers open-source packages for different aspects of robot software development. We detailed the steps required to model a motion planning application within ROS and its motion planning framework MoveIt. Since the planner is not responsible for mapping free and occupied spaces, there is an efficiency benefit, especially for static scenarios. The paper focused on the motion planning aspect. We presented the ROS communication graph of the running system, validating the presence of all system components as indicated by respective nodes and their communication topics. For space limitation, we did not present additional capabilities of our solution, e.g., safety tracking and re-planning. Our results showed that the manipulator successfully adapts the trajectory in the presence of obstacles. Due to its unique strategy for exploring configuration space, RRTConnect performed better as a path planner in different test cases.

A complete work cell can be modelled in a urdf, containing the static objects. Thus, all the modelled objects are checked when generating a self-collision matrix. At run-time, the configuration space to be checked when planning trajectories is smaller, and hence the faster solution. This will be part of future work. We also plan to prepare a case study based on a collaborative assembly operation that involves all components of our proposed approach and demonstrates its efficacy.

ACKNOWLEDGMENTS

The authors acknowledge the support of the ARISE Associated Laboratory (LA/P/0112/2020) and R&D Unit SYSTEC -Base (UIDB/00147/2020) and Programmatic (UIDP/00147/2020), and also the support of project RECLAIM – "RE-manufaCturing and Refurbishment LArge Industrial equipment"- GA no. 869884, funded by the European Commission under the H2020 and project PRODUTECH 4S&C - SUSTAINABLE & CIRCULAR PRODUTECH, with reference POCI-01-0247-FEDER-046102, co-funded by FEDER, through COMPETE 2020.

REFERENCES

[1] W. Z. Khan, M. H. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah, "Industrial internet of things: Recent advances, enabling technologies and open challenges," *Computers & Electrical Engineering*, vol. 81, p. 106522, 2020.

[2] Z. Sheng, C. Mahapatra, C. Zhu, and V. C. M. Leung, "Recent Advances in Industrial Wireless Sensor Networks Toward Efficient Management in IoT," *IEEE Access*, vol. 3, pp. 622–637, 2015.

[3] X. Li, D. Li, J. Wan, A. V. Vasilakos, C.-F. Lai, and S. Wang, "A review of industrial wireless networks in the context of industry 4.0," *Wireless networks*, vol. 23, pp. 23–41, 2017.

[4] B. McMorris, "A History Timeline of Industrial Robotics," https://futura-automation.com/2019/05/15/a-history-timeline-of-industrial-robotics/, 2022, accessed: 2022-10-15.

[5] IFR, "IFR International Federation of Robotics - Robot History," https://ifr.org/robot-history, 2021, accessed: 2022-10-10.

[6] J. A. Marvel, "Performance Metrics of Speed and Separation Monitoring in Shared Workspaces," *Transactions on Automation Science and Engineering*, vol. 10, no. 2, pp. 405–414, 2013.

[7] P. Anderson-Sprecher, "Intelligent Monitoring of Assembly Operations," Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-12-03, June 2011.

[8] J. Fryman and B. Matthias, "Safety of Industrial Robots: From Conventional to Collaborative Applications," in *7th German Conference on Robotics (ROBOTIK)*. VDE, 2012, pp. 1–5.

[9] P. Tsarouchi, S. Makris, and G. Chryssolouris, "Human–robot interaction review and challenges on task planning and programming," *International Journal of Computer Integrated Manufacturing*, vol. 29, no. 8, pp. 916–931, 2016.

[10] P. A. Lasota, G. F. Rossano, and J. A. Shah, "Toward Safe Close-Proximity Human-Robot Interaction with Standard Industrial Robots," in *10th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2014, pp. 339–344.

[11] P. Rybski, P. Anderson-Sprecher, D. Huber, C. Niessl, and R. Simmons, "Sensor Fusion for Human Safety in Industrial Workcells," in *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2012, pp. 3612–3619.

[12] M. Bdiwi, M. Pfeifer, and A. Sterzing, "A new strategy for ensuring human safety during various levels of interaction with industrial robots," *CIRP Annals*, vol. 66, no. 1, pp. 453–456, 2017.

[13] U. Robots, "UR5 collaborative robotic arm," https://www.universal-robots.com/products/ur5-robot/, 2015, accessed: 2019-12-20.

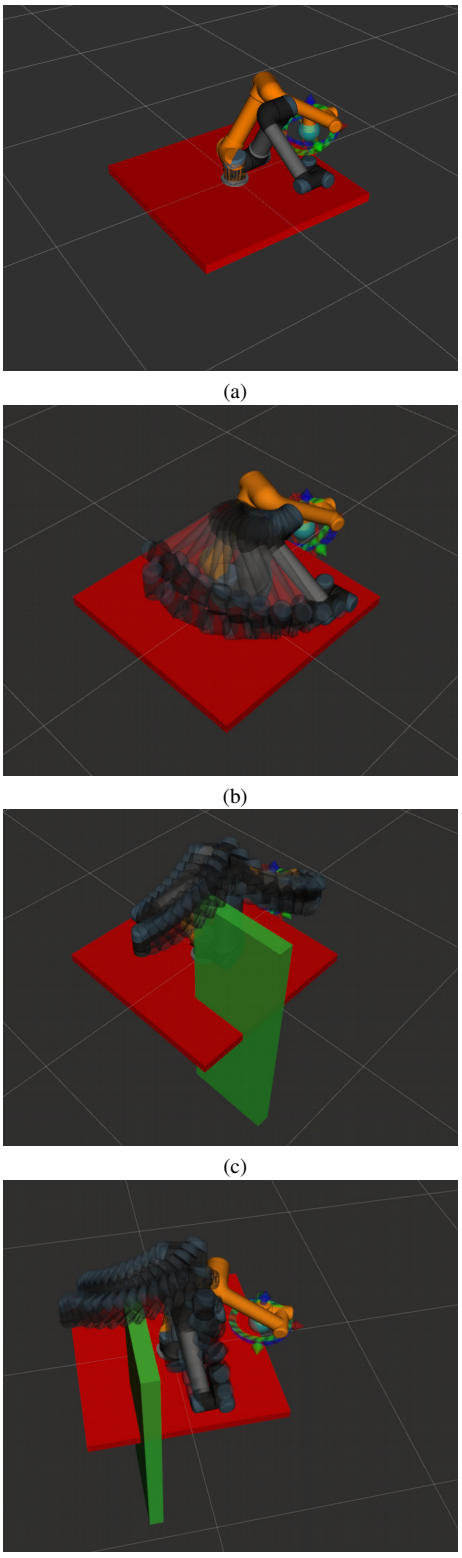
[14] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[15] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[16] A. Roncone, O. Mangin, and B. Scassellati, "Transparent Role Assignment and Task Allocation in Human Robot Collaboration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1014–1021.

[17] Y. Pyo, H. Cho, R. Jung, and T. Lim, *ROS Robot Programming*. Robotis Co., Ltd., 2017, p. 50.

 ISBN: ISBNFILL



(a) (b) (c) (d)

Fig. 7. Motion planning of UR5 manipulator.

[18] I. A. Sucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[19] J. Pan, S. Chitta, and D. Manocha, "FCL: A General Purpose Library for Collision and Proximity Queries," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 3859–3866.

[20] D. Thornton Coleman IV, "Methods for Improving Motion Planning Using Experience," Ph.D. dissertation, University of Colorado, 2017.

[21] M. Kallman and M. Mataric, "Motion Planning Using Dynamic Roadmaps," in *International Conference on Robotics and Automation (ICRA)*, vol. 5. IEEE, 2004, pp. 4399–4404.

[22] L. Antão, J. Reis, and G. Gonçalves, "Voxel-based Space Monitoring in Human-Robot Collaboration Environments," in *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 552–559.

[23] R. Nogueira, J. Reis, R. Pinto, and G. Gonçalves, "Self-adaptive Cobots in Cyber-Physical Production Systems," in *24th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 521–528.

[24] I. GitHub, "Universal Robot," https://github.com/ros-industrial/universal_robot, 2019.

[25] T. T. Andersen, "Optimizing the Universal Robots ROS driver." https://orbit.dtu.dk/en/publications/optimizing-the-universal-robots-ros-driver, Technical University of Denmark, Tech. Rep., 2015.

[26] P. Beeson and B. Ames, "TRAC-IK: An Open-Source Library for Improved Solving of Generic Inverse Kinematics," in *15th International Conference on Humanoid Robots (Humanoids)*. IEEE-RAS, 2015, pp. 928–935.

[27] P. Beeson and B. Ames, "trac_ik," https://bitbucket.org/traclabs/trac_ik/src/master/, 2019, accessed: 2020-01-25.

 ISBN: ISBNFILL

An Active-Logic Based Agent's Reasoning for Avoiding Futile Action Repetition

Anthony Herron
Dept. of Computer Science
Bowie State University
Bowie, United States
herrona0814@students.bowiestate.edu

Darsana P. Josyula
Dept. of Computer Science
Bowie State University
Bowie, United States
darsana@cs.umd.edu

***Abstract*—An agent working on tasks with tight deadlines must be cognizant of the passage of time and perform effective actions that would let it complete its tasks on time. Avoiding actions that do not contribute towards task completion is desirable; executing actions whose post-conditions are already met might not help in completing the task at hand. Moreover, repeating an action after post-conditions become true can be a wasted effort that affects the ability of the agent to complete its tasks. In this paper, we explore several sets of axioms that can be used by a time-situated agent for avoiding repeated actions. We also examine how the agent's knowledge temporally evolves while using these axioms for a time-constrained task and illustrate how these axioms affect task performance on a target search task.**

***Index Terms*—active logic, action-selection axioms, agent reasoning.**

I. Introduction

Over the span of an agent's lifetime, its knowledge changes based on its actions and perceptions. These changes, in turn, impact the decisions of the agent, influencing future actions and outcomes. When agents have a repertoire of actions at their disposal, there is a decision involved regarding which of the possible actions should be done and which should be avoided. Knowing whether an action has been previously tried and whether the post-conditions of the action hold true can help the agent decide whether to redo that action.

All actions whose preconditions are met are possible actions that can be executed, but not all of those actions will lead the agent toward a goal state. Particularly, performing the same action that has already been tried may not lead to a different outcome. A smart agent would take actions that steer it towards its goals, which in many cases would be the actions whose post-conditions are not met and are related to the goal. After executing an action once, if the post-conditions are true, a smart agent would avoid said action as constantly repeating the same action that has the post-conditions already met may not take the agent closer to its goal.

Besides, doing and redoing actions takes time; the clock is still running when performing futile actions. When agents work on tasks with tight deadlines, time is a limited resource that the agent has to manage carefully using the knowledge that it has at its disposal wisely. For an agent to be aware of this passage of time, it should be situated in time [1]. While being cognizant of the deadline for a task, the agent must choose and perform actions in a way that will lead to task completion. It would not make much sense for an agent to needlessly perform actions that have been done already since these repeated actions will generally waste precious time.

For instance, while searching for a car key when one is rushing to leave for the airport, time is limited as there is a deadline to meet (to reach the airport), and there are many areas to cover during the search. The goal is then to perform actions that will allow the agent to cover the areas where the key will likely be present quickly. Searching in places where the key doesn't exist affects the time it takes to find the key. Repeatedly searching a location where the agent already searched and ascertained that the key does not exist, instead of searching a new location, is futile and would consume valuable time left to complete the task. So, it is important to avoid such repeated actions that are not going to take the agent closer to completing the task.

In the search example, there is a better chance of finding a key in a spot that one hasn't searched previously. Even then, sometimes one has to retrace the areas that one has already searched in order to search in locations beyond the areas already searched or to get a better look at the areas previously searched. Hence, an agent that strictly avoids any repeated action is not desirable. If the entire area has already been searched and the target is not found, a wise option is to search the areas that have already been checked again. This would address the issue of simply overlooking the item, as in the keys being underneath something.

The conditions that lead to an action being selected or not influence agent behavior and task performance. If a successful post-condition prevents an action from being selected, an agent may never revisit a location that it has already been before. On the other hand, without a post-condition check, the agent may mindlessly keep repeating actions. Not only that, the amount of knowledge that is noted while the conditions are checked affects the choice of actions selected and hence agent behavior and task performance.

In this paper, we study the complex interactions in time between action-selection decisions and knowledge-condition checks using a time-situated agent based on active logic [2] in the context of reducing unnecessary repeated actions. We present six sets of axioms for action selection that an active-

 ISBN: ISBNFILL

logic based agent utilizes to avoid repeated actions. Utilizing these sets of axioms in a search task setting with a deadline, we discuss the temporal interactions between action selection and knowledge-condition checks.

The following sections are organized as follows: Section II discusses the relevant literature in this area of research. Section III discusses the characteristics and functioning of active logic. Section IV explores six different action-selection axiom sets used in this work and how they avoid repeated actions. Section V details the experiment conducted, highlighting the different variables considered. Section VI examines the results from using the axiom sets during a search task.

II. Related Work

The reduction of redundant actions has previously been explored in the pursuit of creating intelligent agents. Chrpa, McCluskey, and Osborne [3] discussed techniques for determining whether an action is or is not redundant based on action dependencies. This work could not determine every possible redundant action within a plan as they are not well defined in the plan.

The work of Balyo [4] sought a similar goal but also focused on attempting to eliminate redundant actions from plans. The methods discussed covered solutions such as greedy algorithms and action reduction. The resulting elimination solutions were slow in some cases but were successful in eliminating redundant actions.

Baram, Tennenholtz, and Mannor [5] explored redundancy avoidance in reinforcement learning through transition entropy and tried to eliminate the redundancies in both deterministic and stochastic settings through MDPs, an actor-critic framework, and Q-Learning. Actions are given an action redundancy score (deterministic) and an action redundancy ratio (stochastic). The actor-critic framework attempts to update the action redundancy score and action redundancy ratio using a transition buffer but uses an old policy. The proposed Q-Learning algorithm was created to fix that issue.

The work of Zahavy [6] takes a different approach by using deep neural networks to eliminate redundant actions. A binary signal is created based on auxiliary rewards through a Markov decision process that determines if an action should be chosen.

The gaming field is another area where redundant action avoidance is being researched. The primary technique used is Monte-Carlo Tree Search. Santos, Bernardino, and Hauck [7] sought to improve the Rolling Horizon Algorithm with a shift buffer to redistribute actions by shifting actions and generating new actions. The redundancy avoidance is applied to the Monte-Carlo Tree Search, which uses the actions from the shift to find the best possible next actions by testing all actions at the current state.

The work explored in [8] uses the Monte-Carlo Tree Search to focus not only on redundant action avoidance but also on loss avoidance. This process involves pruning the tree such that a minimum amount of actions and the associated losses are kept. The pruning also accounts for nodes generated by Monte-Carlo Tree Search that are deemed redundant.

III. Active Logic

Active logic [9] differentiates itself by being an agent's internal reasoner [10] with the ability to keep track of the passage of time and diffuse direct contradictions [11]. The core of active logic is first-order logic with inference rules to maintain the evolving Knowledge Base (KB) current. The facts in active logic are not set in stone, and every fact that is stored within the agent's KB can be accessed and revised at a later time.

Active logic maintains the passage of time using the clock rule that keeps track of the current time. The clock rule, $now(t) \rightarrow now(t+1)$ uses the fact that now the time is t at time t, to infer that now has become t+1 at time step (t+1). At time t, the agent has a record of the current beliefs and observations within the agent's KB. Under normal circumstances, these get inherited to the next time step. Additionally, using Modus Ponens on the existing knowledge at time t, new facts are inferred and asserted into the KB at time t+1. Thus, at each time step, an agent will have a record of when each piece of knowledge was gained.

Active logic also has a distinct feature of contradiction detection and handling. Agents cannot have two conflicting beliefs at the same time. Active logic distrusts conflicting beliefs and prevents new knowledge from being derived from them until the conflict is resolved. Suppose an active logic sentence P exists in the KB when a new inference (or observation) $\tilde{}P$ occurs at time t. This causes conflict in the agent's beliefs, and the logic distrusts both P and $\tilde{}P$ and asserts $contra(P,\tilde{}P)$ in its KB at time step t+1. The predicate, $contra(P,\tilde{}P)$, serves as a way to note in the KB that a direct contradiction between P and $\tilde{}P$ has occurred.

IV. Action-Selection Axiom sets

An active-logic based agent goes through a $perceive - think - act$ cycle to perform its tasks. The cycle repeats itself, allowing the agent to keep executing actions. The subset of axioms that the agent uses for action selection within the $think$ phase is the focus of this paper. This section discusses six sets of action-selection axioms [12], along with how each axiom set handles repeated action avoidance. The first is the baseline action selection, which contains no knowledge for avoiding repeated actions. This is followed by the remaining five axiom sets that contain knowledge for avoiding repeated actions. When improving the baseline action selection, the core pattern was preserved and adjusted to increase the amount of caution towards repeated actions within the agent. Each pattern includes an additional piece of knowledge that affects the agent's behavior leading to potentially more stringent repeated action avoidance. Repeated action avoidance is handled using five different action-selection axiom sets with increasing levels of caution. The different Repetition Avoidance (RA) axiom sets are RA 2, RA 2-3, RA 2-4, RA 3-4, and RA 3-5.

Each axiom set has a corresponding sequential block diagram that outlines the conditions needed to select an action. The diagrams flow downward through vertical arrows from one level to the next. Each level within the diagram represents a

 ISBN: ISBNFILL

time step during reasoning and is separated by dotted lines. On the left of each level is its corresponding time step, $T + 1$, $T + 2$, etc. The diamonds in the figures represent knowledge checks that occur at the time step, while the blue rectangles symbolize knowledge being asserted at the time step, and a green rectangle indicates beliefs inherited from the previous steps. Multiple arrows from an entity indicate an and condition.

Activities associated with every block that appears at the same time step occur simultaneously. For example, both pre-condition and post-condition checks happen independently but simultaneously at T. At the first time step of reasoning, some knowledge checks will automatically fail as that knowledge has not been asserted yet. However, as time passes and new observations come in, when the same rules fire at a later time step, the knowledge conditions (the antecedents) may hold true for the consequent to get asserted in the KB.

An action that can be executed by the agent is noted in the KB as a *can do action*. Once an action has been selected amongst the *can do actions*, the action-selection specific inference knowledge is removed, and the entire process of selecting an action automatically activates again with a new set of *can do actions*. The action-selection specific inference knowledge in the figures that are removed when an action is selected include *can do actions*, *feasible actions*, *original actions*, *duplicate actions*, and *contingent actions*.

A. Baseline - No Repetition Avoidance (Naive 2 Steps)

The baseline action-selection axiom set infers action whose preconditions are met as possible *can do actions*. For the sake of this paper, this axiom set will be called Naive. Naive takes a total of two time steps for the reasoning to select an action. The only requirement to start reasoning about an action is having current beliefs. When the agent has a new belief, the agent will check whether both post-conditions and preconditions are satisfied separately. An example of a precondition for a move action is the availability of an empty location to move to. When the preconditions are met, a possible action is now available to be performed. If the post-conditions of an action for an object are met, then the action is marked as completed. An example of the post-condition of a move action to a location is for the agent's current position to change to that location.

Naive is the most extreme of the bold action-selection axiom sets, focusing on selecting an action whose preconditions are met. If there are multiple actions with preconditions met available at any time step, one will be selected at random. Due to the naive nature of the baseline action-selection axiom set, the agent will act randomly within the environment. If the neighboring location being considered is empty, there is the possibility that the agent moves to that location with no regard to whether or not the action is a repeated action. As a result, the agent can repeatedly search areas where it has been searching, causing the agent to potentially fail the task. In extreme cases, it is possible for the agent to move in a circular motion and waste precious time because it has no restrictions on where to move.

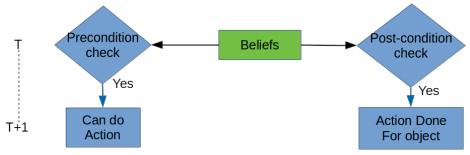


Fig. 1. Naive reasoning abstraction

B. Only Non-duplicated Actions (2 Steps Deliberation)

The 2 Steps Deliberation for action selection, formally named RA 2, completely eliminates any repeated actions by building upon the structure of the baseline pattern and introducing an extra-knowledge check. While the baseline action-selection axiom set selects any action whose preconditions are met, RA 2 selects only those actions that are not repeated as possible *can do actions*. At every time step, the agent performs a check to determine if the post-conditions of an initiated action are met, and when the check succeeds, it stores that knowledge. This knowledge is inherited in future time steps, and hence if an action was executed in a previous $perceive - think - act$ cycle, that knowledge is accessible for later use. The agent uses this knowledge in the subsequent $perceive - think - act$ cycles to check if an action was performed for an object; if the check is true, the agent will not redo that action for the object. RA 2 does not increase the number of time steps in the $think$ phase for action selection when compared to the naive method. However, the agent becomes more cautious due to repetition avoidance.

In the search task example, this axiom set can cause the agent to become boxed in. The agent can potentially visit surrounding areas in one direction, which forces the agent to move to one side of the environment. When all its immediate neighboring locations are visited, the agent has no new locations to move into without stepping on an already visited location. Hence, it becomes stuck at its position and is unable to complete its task frequently.

C. Non-duplicated actions inferred one step before other feasible (preconditions met) actions (2-3 Steps Deliberation)

While the 2-3 Steps Deliberation method (also called RA 2-3) uses the core structure from the previously introduced action-selection axiom sets to select those actions that are not repeated as possible *can do actions* in two steps, it takes a different path by asserting additional knowledge for

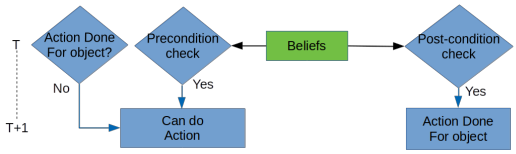


Fig. 2. RA 2 reasoning abstraction

 ISBN: ISBNFILL

each action whose preconditions are met as a *feasible* action. This knowledge is used to select actions to repeat when no action that has not been done previously is available. Thus, a *feasible action* becomes a possible *can do action* in a $perceive - think - act$ cycle as long as no other action got selected within the $think$ phase in two steps. If the agent infers *can do actions* in two steps, then the $think$ phase ends, and the agent transitions to the act state. With this adjustment, the agent can perform feasible actions that the agent had previously done for an object when no actions that were not done previously exist for that object. This differentiates it from RA 2, which becomes inoperable in the same situation. Since a possible repeated action is derived one time step after an action is marked as feasible, while actions that are not attempted previously are immediately selected for execution, the original actions will be preferred over repeated actions by agents using this axiom set. The combination of these concepts requires at least two or three time steps of action-selection reasoning in the $think$ state.

Using the search task example, RA 2-3 uses a relatively balanced approach but leans towards being bold as it attempts to prioritize speedy action. An empty location that has not been visited before, if available, is immediately chosen as a possible location to move to. RA 2-3 takes a different approach to repeated action avoidance by utilizing the stored *feasible actions* knowledge as a means of selecting previously tried actions. For the target search task, the agent notes empty neighbors as feasible locations to move to. This knowledge is used to infer possible locations to move to if the $think$ phase doesn't provide a possible action that has not been done before, i.e., an empty neighboring unvisited location to visit.

D. Non-duplicated actions preferred over feasible duplicated actions (2-4 Steps Deliberation)

The 2-4 Steps Deliberation method (RA 2-4) is very similar to the 2-3 deliberation method because they both add new means of performing an action when an action was previously done for an object. The core structure from the previously introduced action-selection axiom sets remains, but *duplicate actions* are asserted instead of immediately selecting a repeated action. In RA 2-4, *duplicate action* knowledge is derived at the same time step that repeated actions are selected in RA 2-3. The selection of repeated actions from *duplicate actions* becomes possible one time step later. This leads to the $think$ state reasoning for action selection taking at least two or four time steps instead of two or three. A *duplicate action* will only be executed if it is also a *feasible action*.

By asserting *duplicate actions*, the action selection process gains a new level of caution towards repeated actions because the agent will have more time to think about what is occurring. RA 2-4 will still prioritize actions not previously done before, similar to how RA 2-3 performs. RA 2-4 shares the same problem of making decisions too quickly while considering actions that have not been previously selected, as in RA 2-3. The difference is the agent has more time to think about selecting a repeated action due to noting that the action is a *duplicate action*, but RA 2-4 still performs new actions potentially too quickly. When an action was not previously done, a possible action is asserted immediately, which is given the highest priority. *Duplicate actions* will primarily play a role when the agent becomes stuck or when a new action is currently not available. While the RA 2-3 action-selection axiom set prioritizes speed, the RA 2-4 action-selection axiom set prioritizes slightly more knowledge representation and reasoning for repeated actions. With this being a middle ground for the action-selection axiom sets, it is the perfect balance of boldness and caution for agents.

E. Feasible actions marked as original or duplicate; original preferred over duplicate (3-4 Steps Deliberation)

The 3-4 deliberation method (RA 3-4) appears the most different from the previous action-selection axiom sets, but the concept from RA 2-4 remains in this axiom set. The important addition is asserting additional information on the outcomes of checking if an action was done for an object. This is accomplished through *duplicate actions* similar to RA 2-4 but also asserting *original actions* when an action is not previously done for an object. Thus, the possible action from an action not previously done is also pushed down a time step to accommodate for checking if the predicate *original action* exists in the knowledge base at the time. An *original action* will always take precedence over a *duplicate action* if both are available. The *original actions* can also only be executed if they are also *feasible actions*.

Due to the addition of asserting *original actions*, RA 3-4 becomes much more cautious as three time steps are needed to

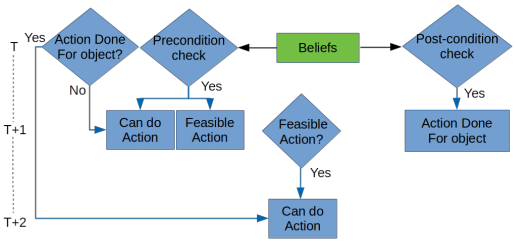


Fig. 3. RA 2-3 reasoning abstraction

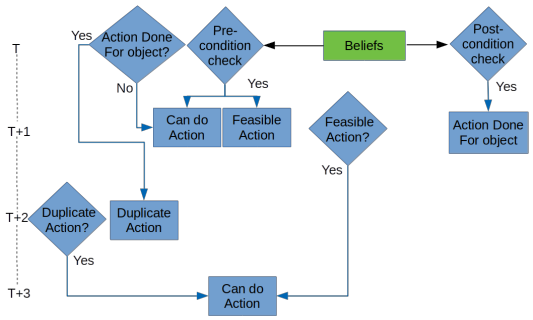


Fig. 4. RA 2-4 reasoning abstraction

select an action even in the best case. The main characteristic of this axiom set is the alternative option for correcting the boxed-in scenario present within RA 2 and addressing the shortcomings of RA 2-3 and RA 2-4 by introducing *original actions* and *duplicate actions* for the agent to consider. In every situation, an *original action* is prioritized over a *duplicate action* because the agent is less likely to complete the task by revisiting locations. If there are only inspected locations surrounding the agent, the agent must do the *duplicate action* to find possible overlooked areas. One possible downside to this action-selection deliberation is an *original action* arriving late into the KB (perhaps preconditions became true a few steps into the $think$ phase), forcing the agent to select the *duplicate action*, resulting in possibly a sub-optimal move. While this is not an inherent flaw of the reasoner, it is worth mentioning as the agent does not know that waiting longer will result in an *original action*.

F. Feasible actions marked as original or duplicate; feasible duplicates marked as contingent; original preferred over contingent (3-5 Steps Deliberation)

The core of the 3-5 deliberation method (RA 3-5) is almost exactly the same as RA 3-4. This action-selection axiom set makes the agent represent and reason with extra knowledge for selecting a repeated action. Instead of going immediately to a possible *can do action* from a *duplicate action*, it asserts a *contingent action* in the next time step. The *contingent action* will only result in a possible action if no *original actions* have appeared and the action is also a *feasible action*. Once there is a *duplicate action* present, then the *contingent action* knowledge is derived in the KB. Like the 3-4 deliberation step-logic, an *original action* still takes precedence over other actions. In the best case scenario, the $think$ state can produce a possible action in three time steps when actions that have not been attempted are available to select from. When such actions are not present, the agent will take at least five time steps to select a repeated action.

The 3-5 Step reasoning uses all of the previous patterns and addresses the issues with each action-selection axiom set. By using both *duplicate action* and *original action* after an action is done, the issue of being too bold in some cases is solved. To address the issue of the agent needing more time to process *duplicate actions*, *contingent actions* were used to make the agent wait one extra time step before asserting a possible action through *duplicate actions*. The result of the combination of knowledge is extreme caution, which is reflected in the amount of time needed to perform any action.

With this being the most cautious of the action-selection axiom sets, the agent takes extra time steps to select the action to return to a previous location. In the search task, this allows going to unexplored locations that lay beyond the visited ones only when immediate neighbors need not be explored. The extra steps for repeated actions in the $think$ step allows more time for inferring new actions that could be selected. There is a chance that waiting more time steps could also lead to similar results, but this can lead to a slippery slope. The addition of more time steps could stall the agent enough that it rarely completes its task.

V. Experiment

This experiment uses an active-logic based agent tasked with finding a unique object within a virtual environment (AI2-THOR) using one of the six action-selection axiom sets. Agents have a deadline of 100 time steps to locate the target during a trial. There are a total of 20 trials containing different starting locations for either the agent or the target. These 20 trials are also repeated as a set, five times for each pace length and axiom set. Pace length represents the units needed for an agent to move to a new location in the environment. For this experiment, the pace lengths used were .2 through .45, with an interval of .5 pace length. Effectively, there were six different pace lengths used. Each set was also repeated three times during the experiment. Thus, every action-selection axiom set was used to perform a total of 1,800 trials.

VI. Results

The results from Fig. 7 contain the average success of finding the target for each action-selection axiom set. The higher numbers (brighter colors) represent better success in finding the target. A successful trial requires an agent to find the target and infer that the task has been completed. If the target has been found within the deadline, but the completion

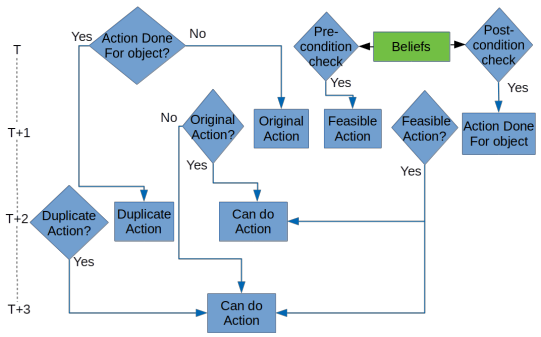


Fig. 5. RA 3-4 reasoning abstraction

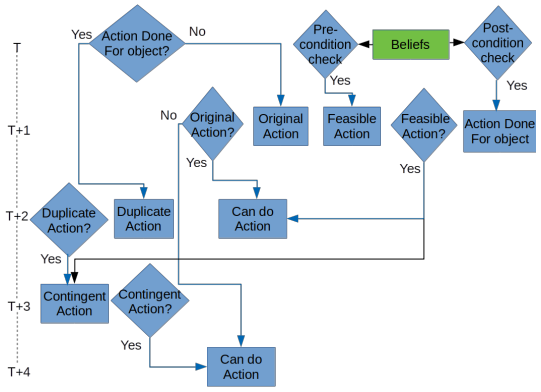


Fig. 6. RA 3-5 reasoning abstraction

of the task has not been inferred during the deadline, the trial will result in failure. The extremes of bold and cautious axiom sets did not perform as well as the middle ground axiom sets. The bold axiom sets struggled to find the target due to acting without any knowledge or failing to find actions. More specifically, RA 2 performed similarly to the balanced axiom sets at lower pace lengths but started to perform much worse at the higher pace lengths. At the lower pace lengths, it is much harder to become boxed in because of the amount of movement required to cover the environment. The opposite becomes true at higher pace lengths as coverage of the environment will take much less time, creating more opportunities to become boxed in. Overall, increasing the pace length had a positive impact on the performance of all axiom sets.

Fig. 8 shows how quickly an agent using one of the action-selection axiom sets completed the task. Unlike Fig. 7, the higher numbers, in this case, represent poor performance. This set of results does not include any trial where the agent fails to find the target. Thus, the times shown will not include any time that an agent has failed to meet the deadline, resulting in reduced times. This is important because RA 2 has the best speed for finding the target but fails to find the target often. With that in mind, RA 2 does complete the task very quickly when it does find the target. When using both Fig. 7 and Fig. 8, the balanced axiom sets produced the best results overall. The completion times of the cautious axiom sets were hindered by the extra reasoning time, which was the expected outcome. The extra reasoning time likely also played a factor in failing to complete the task.

The heatmap in Fig. 9 highlights the average percentage of new actions performed within the environment during the trials. By design, RA 2 eliminated all repeated actions as possible actions as it was deemed futile. Thus, RA 2 selected a new action whenever possible, resulting in a new action percentage of 100%. Also, as expected, Naive performed new actions close to 50% of the time. If all actions have equal weight of selection, the deciding factor is only what is present at a given time step. Therefore, there will be time steps where an agent will only have new actions or only repeated actions. The remaining axiom sets performed actions at a progressive rate overall, selecting slightly more new actions than the

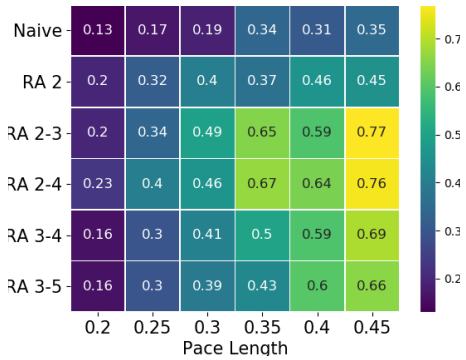


Fig. 7. Accuracy of finding the target

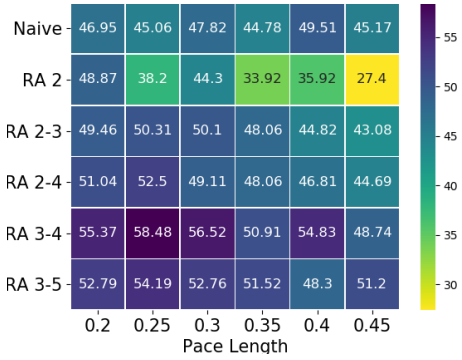


Fig. 8. Average time steps to find the target

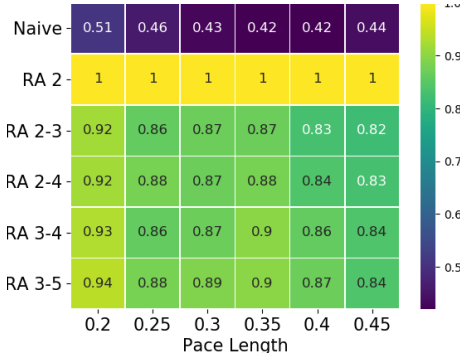


Fig. 9. Percentage of original actions selected

next. Fig. 10 contains the average percentages of new action performed during the experiment.

Overall, the balanced axiom sets performed the best at using the knowledge to avoid repeated actions while completing the task effectively. The best-performing axiom set was RA 2-4, while Naive resulted in the worse performance in every category. RA 2 actually hinders its performance by eliminating all repeated actions. The elimination of repeated actions prevents returning to locations to find undiscovered areas or causes additional time to avoid the action. The additional time added is only a problem if the agent is unable to find the target due to the process. This is also true when referring to avoiding futile actions. Increasing the time, whether it is selecting actions to avoid the futile action or spending additional time to think about the best option, will hurt the potential speed of finding the target; but it will only potentially harm the success rate. Therefore, when comparing the results for the cautious and balanced axiom sets, the cautious axiom sets had a relatively close performance. Fig. 10 shows the average success rate for each axiom set. Based on the success rate, the additional knowledge added in RA 3-5 starts to hurt the performance. Thus, increasing the time to reason about repeated actions will increasingly make the agent fail its task more.

VII. Conclusion

We discussed several axiom sets for action selection that represent and reason with varying knowledge content for avoiding futile repeated actions. We examined how an agent

 ISBN: ISBNFILL

Results for the Action Selection Axiom Sets		
Axiom Set	New Moves	Success Rates
Naive	45%	24.7%
RA 2	100%	36.7%
RA 2-3	86%	50.7%
RA 2-4	87%	52.7%
RA 3-4	88%	44.2%
RA 3-5	89%	42.3%

Fig. 10. Average new moves and success rate percentages

using these axiom sets will have its knowledge evolve as time progresses and how the evolving knowledge affects task performance using a simple target search task. This research examined the strength and weaknesses of each axiom set and discussed the performance of time-situated agents using these axiom sets in a search task. Further work to analyze the behavior of multiple active-logic based agents collaborating in the environment on the same task is ongoing. In the future, we will study the interactions of more complex knowledge (axioms, inferences, and observations) with task performance and success rates for active-logic based agents.

ACKNOWLEDGMENT

This work was supported by DARPA Project No. HR00112090025, U.S. Department of Education Title III HBGI Grant, and ArtIAMAS Subaward No. 11788-Z8523201, Prime Award No. W911NF2120076. Special thanks to D. Perlis, M. Goldberg, C. Maxey, T. Clausner, J. Conroy, D. Conover, N. Vale, and J. Milzman for their input.

REFERENCES

[1] M. Nirkhe, S. Kraus, and D. Perlis., "Thinking takes time: A modal active-logic for reasoning in time," in *Proceedings of BISFAI-95*. AAAI Press, 1995.

[2] D. P. Josyula, A. Herron, and K. M'Bale, "Implementing a task-oriented time-situated agent," in *Eighth Annual Conference on Advances in Cognitive Systems*, 2020.

[3] L. Chrpa, T. L. McCluskey, and H. Osborne, "Determining redundant actions in sequential plans," in *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, vol. 1. IEEE, 2012, pp. 484–491.

[4] T. Balyo, L. Chrpa, and A. Kilani, "On different strategies for eliminating redundant actions from plans," in *Seventh Annual Symposium on Combinatorial Search*, vol. 5, 2014, pp. 10–18.

[5] N. Baram, G. Tennenholtz, and S. Mannor, "Action redundancy in reinforcement learning," in *Uncertainty in Artificial Intelligence*. PMLR, 2021, pp. 376–385.

[6] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor, "Learn what not to learn: Action elimination with deep reinforcement learning," *Advances in neural information processing systems*, vol. 31, pp. 3566–3577, 2018.

[7] B. Santos, H. Bernardino, and E. Hauck, "An improved rolling horizon evolution algorithm with shift buffer for general game playing," in *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 2018, pp. 31–316.

[8] D. J. Soemers, C. F. Sironi, T. Schuster, and M. H. Winands, "Enhancements for real-time monte-carlo tree search in general video game playing," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 1–8.

[9] D. Perlis, K. Purang, D. Purushothaman, C. Andersen, and D. Traum, "Modeling time and meta-reasoning in dialogue via active logic," in *Working notes of AAAI Fall Symposium on Psychological Models of Communication*, 1999.

[10] D. Perlis, J. Brody, S. Kraus, and M. Miller, "The internal reasoning of robots," in *Thirteenth International Symposium on Commonsense Reasoning*, 2017.

[11] J. J. Elgot-Drapkin and D. Perlis, "Reasoning situated in time I: basic concepts," *Journal of Experimental Theoretical Artificial Intelligence*, vol. 2, pp. 75–98, 1990.

[12] A. Herron and D. P. Josyula, "An analysis of the deliberation and task performance of an active logic based agent (student abstract)," in *Thirty-Seventh AAAI Conference on Artificial Intelligence*. AAAI, 2023.

 ISBN: ISBNFILL