A publication of ISCA*:
International Society for Computers
and Their Applications

INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS

TABLE OF CONTENTS

***"International Journal of Computers and Their Applications is Peer Reviewed".**

Volume 30, No. 1, March 2023 **ISSN 1076-5204**

International Journal of Computers and Their Applications

A publication of the International Society for Computers and Their Applications

ISCA Headquaters • 278 Mankato Ave, #220, Winona, Minnesota 55987, USA
Phone 507-458-4517 • Email isca@isca-hq.org • URL: http://www.isca-hq.org

Editorial

It is my distinct honor, pleasure, and privilege to serve as the new Editor-in-Chief of the International Journal of Computers and Their Applications (IJCA) for the second year. I have a special passion for the International Society for Computers and their Applications. I have been a member of our society since 2014 and have served in various capacities. These have ranged from being on program committees of our conferences to being Program Chair of CATA 2021 and CATA 2022 and currently serving as one of the Ex-Officio Board Members. I am very grateful to the ISCA Board of Directors for giving me this opportunity to serve society and the journal in this role.

I would also like to thank all the editorial board, editorial staff, and the authors for their valuable contributions to the journal. Without everyone's help, the success of the journal would be impossible. I look forward to working with everyone in the coming years to maintain and further improve the journal's quality. I want to invite you to submit your quality work to the journal for consideration of publication. I also welcome proposals for special issues of the journal. If you have any suggestions to improve the journal, please feel free to contact me.

Dr. Ajay Bandi
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
Email: AJAY@nwmissouri.edu

In 2023, we have four issues planned (March, June, September, and December). March issue includes the selected papers from the SEDE 2022 and open submissions. Drs. Fred Harris, Rui Wu, and Alex Redei are the program co-chairs of the SEDE 2022. June issue will include the selected best papers from CAINE 2023 and open submissions. The September issue will contain the best papers from CATA 2022 and open submissions. The last issue is taking shape with a collection of submitted papers.

I would also like to announce that I will begin searching for a few reviewers to add to our team. There are a few areas in which we would like to strengthen our board. If you would like to be considered, please contact me via email with a cover letter and a copy of your CV.

Ajay Bandi, Editor-in-Chief
Email: AJAY@nwmissouri.edu

Guest Editorial
March 2023

This issue of of the International Journal of Computers and their Applications (IJCA) is split into two parts. The first part is a collection of four refereed papers selected from SEDE 2022, the second part is IJCA contributed papers which have gone through the normal review process. The papers in this issue cover a broad range of research interests in the community of computers and their applications.

ISCA Fall 2022 SEDE Conference: SEDE 2022 - The 31st International Conference on Software Engineering on Data Engineering, was held October 17-18, 2022. Due to the pandemic it was held virtually. Each paper submitted to the SEDE 2022 conference was reviewed by at least two members of the international program committee, as well as by additional reviewers, judging for originality, technical contribution, significance and quality of presentation. The proceedings for this conference can be found online at https://easychair.org/publications/ volume/SEDE_2022. We had a very good keynote presentation from Dr. Jalal Kiswani entitled "Software Development: Past, Present, and Future" where he walked through the demand for better ways to do software development, which started with waterfall, then agile and iterative development, followed by components re-use, and lately, moving towards Rapid Application Development (RAD) approaches led by prototyping, low-code no-code tools, platforms, and frameworks.

After the conference, the four best papers were recommended by the program committee members to be considered for publication in this special issue of IJCA. The authors were invited to submit a revised version of their papers. After extensive revisions and a second round of review, these papers were accepted for publication in this issue of the journal. The topics and main contributions of the papers are briefly summarized below:

JAYESH SONI, NAGARAJAN PRABAKAR, and HIMANSHU UPADHYAY of Florida International University present their work "MLE-NET: A Multi-Layered Ensemble Approach for an Enhanced Anomaly Detection." They start of by showing that anomaly detection is an important task in many areas. They then proposed a hybrid and ensemble multi-layered approach for robust anomaly detection in their work. The results showed that the hybrid and ensemble multi-layered approach outperforms state-of-the-art anomaly detection methods in terms of robustness and accuracy. They were also able to capture both local and global anomalies which is more comprehensive than traditional methods and has the potential to be applied in a variety of applications.

CHRISTOPHER LEWIS, and FREDERICK C. HARRIS, JR. of the University of Nevada, Reno present their paper "Virtual Reality: An Overview, and How to do Typing in VR." In this paper they present a brief history, current research areas, and areas for improvement in VR. They show that VR has many advantages and that there are a sizable number of deficits that VR needs to solve to be more widely adopted. They then provide some insight into successful typing methods for VR through the use of a user study and a comparison of input methods. It was found that a combination of dictation and a 3D input method led to better results than solely dictation. It was also found that testing input methods with multiple types of input (urls, email addresses, sentences, and paragraphs) gave more varied and detailed results. They found that the addition of more varieties of text in researching typing methods is incredibly important as VR transitions from a novelty entertainment and scientific games platform to an interface that many people can use daily in their work.

BRADFORD A. TOWLE, JR. of Florida Polytechnic University in Lakeland Florida present his work entitled paper "Optimal Control Frequencies for large Number of Virtual Agents in Augmented Reality Applications." In this work, he presents virtual agents (any entity within the program that must periodically run logic and has some graphical effect) and discusses the optimal control frequency for virtual agents across three common AR platforms (the HoloLens1, the HoloLens2, and the Android Note 8). Over the last several years large game engines, including Unity 3D and Unreal, have encouraged AR development. This study was developed with the UNITY 3D platform and the framerate was computed as the number of virtual agents increased. The differences between the hardware was well described and the results of the study showed the change in framerate as the number of agents is increased and the frequency of the updates is changed.

JOSHUA DAHL, ERIK MARSH, CHRISTOPHER LEWIS, AND FREDERICK C. HARRIS, JR. of the University of Nevada, Reno present their paper "uMuVR: A Multiuser Virtual Reality and Body Presence Framework for Unity." In this work they present a new framework for VR under Unity. They show that many frameworks were not designed to support mixed virtual and non-virtual interactions. Therefore, they developed a framework that that lays an extensible and forward-looking foundation for the development of mixed interactions based upon a novel method of ensuring that inputs, visuals, and networking can all communicate without needing to understand the others' internals. They show that their framework provides utilities for representing user avatars in a physicalized manner while supporting a range of different input methods. They compare other frameworks, test networking and compression, and provide a clear separation of Inputs, Visuals, and Networking. Their examples and applications present avatars and voice communication across the multiuser VR environment.

As the SEDE 2022 leadership we would like to express our deepest appreciation to the authors and the program committee members of the conference these papers were selected from.

Frederick C. Harris, Jr, SEDE 2022 Conference Chair
Rui Wu, SEDE 2022 Program Co-Chair
Alex Redei, SEDE 2022 Program Co-Chair

IJCA Contributed Papers: As was mentioned earlier, the second part of this issue is made up of papers that were contributed to the International Journal of Computers and their Applications (IJCA). The topics and main contributions of the papers are briefly summarized below:

INDRANIL ROY from Southeast Missouri State University, Cape Girardeau, NICK RAHIMI from the University of Southern Mississippi, Hattiesburg, ZIPING LIU from Southeast Missouri State University, Cape Girardeau, BIDYUT GUPTA from Southern Illinois University, Carbondale and NARAYAN DEBNATH from Eastern International University, Vietnam present their paper “On Generalization of Residue Class Based Pyramid Tree P2P Network rchitecture”. This paper discusses a 2-layer non-DHTbased structured P2P network that is interest-based and consists of different clusters with peers possessing instances of a particular resource type. Although the network offers efficient data look-up protocols with low latency, it is restricted by the assumption that no peer in any cluster can have more than one resource type. The paper addresses this limitation by proposing effective solutions to generalize the architecture and modifying the previously reported data look-up protocols to accommodate this idea while maintaining the same look-up latencies.

JUBAIR MAHMOOD from Al-Farabi University College in Baghdad, MOHAMMED AHMED JUBAIR and THULFIQAR H. MANDEEL from Imam Ja'afar Al-Sadiq University, Al-Muthanna in Iraq present their paper “Application of Artificial Intelligence to Predict Permeability in Low Permeability Limestone Formation”. The paper proposes the use of artificial intelligence (AI) to predict permeability in low permeability limestone formation, which is important for oil, gas, and water resource calculations. Traditional methods have limitations due to heterogeneous rock properties, and researchers have used core analysis and flow zone indicator (FZI) methods. The paper suggests that supervised machine learning algorithms can provide better predictions for high-dimensional data. The AI algorithm was applied to a dataset for the Khasib formation in the East Baghdad oil field, and the predicted permeability values showed better agreement with actual values than the FZI method. Results were measured using the coefficient of determination (R2).

ANAL KUMAR and HERMANN JAMNADAS from Fiji National University, Nadi, VISHAL SHARMA from Fiji National University, Nasinu, Fiji, S M MUYEEN from Qatar University, Doha, and A. B. M SHAWKAT ALI from University of Fiji, Lautoka present their paper “Review of image steganography tools” This paper reviews various image steganography tools and techniques, highlighting the importance of understanding their strengths and weaknesses in computer forensic inspection. The authors conducted tests on a selection of Java and Windows-based steganography tools, evaluating their encode and decode time, visual differences, and file size variance. The paper emphasizes the need for using steganography tools that minimize the chance of detecting hidden messages and suggests that their findings will be helpful for those interested in utilizing or testing image-based steganography tools.

As guest editors, we would like to express our deepest appreciation to the authors and the reviewers. We hope you will enjoy this issue of the IJCA. More information about ISCA society can be found at http://www.isca-hq.org.

Guest Editors:
Frederick C. Harris, Jr, University of Nevada, Reno, USA,
Rui Wu, East Carolina University, Greenville, USA
Alex Redei, Central Michigan University, USA
Ajay Bandi, Northwest Missouri State University, USA

March 2023

MLE-NET: A Multi-Layered Ensemble Approach for an Enhanced Anomaly Detection

Jayesh Soni*, Nagarajan Prabakar†, Himanshu Upadhyay‡, and Leonel Lagos
Florida International University, Miami, FL 33174, USA.

Abstract

Anomaly detection is an important task in many areas, including cybersecurity, healthcare, and finance, where it is crucial to identify abnormal behaviors or patterns. However, traditional anomaly detection methods can be sensitive to outliers and lack robustness to distributional changes in the data. In order to overcome these limitations, a hybrid and ensemble multi-layered approach for robust anomaly detection has been proposed in this work. The approach consists of a combination of multiple one class classifiers, each trained on a different subset of the data, and a Variational Autoencoder (VAE). The one class classifiers are used to identify local anomalies, while the VAE is used to model the underlying distribution of the data and detect global anomalies. These are the two sets of hybrid features. Next, different one-class classifiers have their strength and limitations. The final decision on whether an instance is anomalous is made by combining the outputs of the one class classifiers and the VAE through an ensemble learning mechanism. Thus, we propose an adaptive weightage approach that gives the weight to each classifier. Next, these reduced hybrid features are passed as input to the second phase. In this phase, we have a deep neural network that learns the patterns of the dataset and generates an adaptive dynamic threshold to discriminate the input feature as an anomaly or benign. The results showed that the hybrid and ensemble multi-layered approach outperforms state-of-the-art anomaly detection methods in terms of robustness and accuracy. Furthermore, the combination of the one class classifiers and the VAE provides a complementary approach that captures both local and global anomalies, making the approach more comprehensive than traditional methods. In conclusion, this work presents a novel hybrid and ensemble multi-layered approach for robust anomaly detection that can effectively address the limitations of traditional methods. The approach has the potential to be applied in a wide range of applications.

Key words: Hybrid multi-layered ensemble, anomaly detection, one class classifiers, variational auto encoders (VAEs), adaptive weightage.

1 Introduction

Anomaly detection is a crucial task in many domains, including cybersecurity, healthcare, and finance, where the ability to identify abnormal behavior patterns is essential. Traditional anomaly detection methods, such as statistical methods and distance-based methods, can be sensitive to outliers and lack robustness to distributional changes in the data. These limitations can lead to false positive or false negative detections, which can have significant consequences in applications such as fraud detection or network security.

Most previous studies suggest that supervised machine learning algorithms can only identify anomalies present in the training dataset. Nonetheless, deviations from normal behavior are referred to as irregularities. As a result, these irregularities may not resemble those already present in the dataset [15]. Additionally, various anomaly detection techniques rely on different and specific rules in the dataset. These algorithms are often specific to a particular domain and detecting anomalies across multiple domains and scenarios with a single model is challenging [1]. The process of training multiple one-class classifiers [17-18] repeatedly with different hyper-parameter optimization techniques is time-consuming. The traditional anomaly detection approach also requires features that are processed in a specific way, which consumes a significant amount of computational resources. While deep learning-based anomaly detection algorithms [14] have shown improved efficiency, they require the data to be in a specific distribution and the developed methods are not easily transferable across domains. To address these limitations, recent research has focused on developing more robust anomaly detection methods that can effectively handle distributional changes and outliers. One promising direction is the use of deep learning-based methods, such as Variational Autoencoders (VAEs), which have shown great promise in modeling the underlying distributions of complex data. VAEs can be used to detect anomalies by identifying instances that deviate significantly from the modeled distribution.

However, VAEs are known to have limitations when it comes to detecting local anomalies, which are anomalies that are specific to a certain region of the data. To address this issue, multiple one class classifiers can be used to identify local anomalies by training each classifier on a different subset of the data. The outputs of the one class classifiers can then be combined to make the final decision on whether an instance is anomalous.

* Applied Research Center.
† Knight Foundation School of Computing and Information Sciences.
‡ Electrical and Computer Engineering.

2 Literature Review

Anomaly detection approaches are classified into three categories based on the availability of data: supervised [8, 11, 24], semi-supervised, and unsupervised. The supervised approach trains the model using binary or multi-class data, but it is not commonly used for anomaly detection due to the class imbalance issue and limited training data [3]. The unsupervised approach detects anomalies solely based on the normal class of data, using methods such as support vector machines [16] and data descriptors [23]. These algorithms have the drawback of being highly sensitive to complex hyper-parameters and not being applicable to multi-class datasets. Clustering techniques [7, 12] have also been used, but these approaches consume a significant amount of computational time and are biased towards a static threshold value. An anomaly detection approach based on deep learning involves training an AutoEncoder and computing the anomaly score based on the reconstruction error [25]. Compared to traditional methods, deep learning-based anomaly detection algorithms have demonstrated better results in capturing complex features of the data [19]. They also offer scalability as an advantage. A recent hybrid approach, as implemented in [6], uses an autoencoder to learn the latent space of high dimensional complex data and provides this learned latent space as input to one-class classifiers for anomaly detection, combining the feature extraction ability of the neural network with the discriminative capabilities of the one-class classifiers. However, this approach relies solely on the autoencoder for feature extraction. To address this issue, we propose an enhanced approach based on EA-Net [20] that not only uses the autoencoder for feature extraction but also integrates several weak one-class classifiers with repeated level of feature detector, resulting in low false-positive rates.

3 Contribution of the work

In this work, we propose a hybrid and ensemble multi-layered approach for robust anomaly detection that combines the strengths of VAEs and one class classifiers. The approach consists of multiple one class classifiers, each trained on a different subset of the data, and a VAE that models the underlying distribution of the data. The final decision on whether an instance is anomalous is made by combining the outputs of the one class classifiers using weightage approach and the VAE through an ensemble learning mechanism. The combination of the one class classifiers and the VAE provides a complementary approach that captures both local and global anomalies, making the approach more comprehensive than traditional methods. We perform multiple experiments where the layer of multiple one class classifier and VAE is repeated to reduce the feature dimension. At the end, we train a deep neural network to provide the final probability of an observation being normal or anomalous.

The rest of this paper is organized as follows: Section 4 presents the proposed hybrid and ensemble multi-layered approach for robust anomaly detection. Section 5 describes the experimental results conducted to evaluate the performance of the proposed method and a comparison with state-of-the-art methods. Finally, Section 6 concludes the paper and discusses potential future work.

4 Proposed Framework

In this section, we describe the proposed MLE Framework, which is illustrated in Figure 1. Figure 1a represents MLE Framework with One Layer Feature Reduction, Figure 1b represents MLE Framework with Two Layer Feature Reduction and Figure 1c represents MLE Framework with Three Layer Feature Reduction. The framework consists of two phases: Hybrid Feature Extraction with different layer and Anomaly Detection.

4.1 Hybrid Feature Extraction

In the Hybrid Feature Extraction component, we derive hybrid features from the high-dimensional data. This is achieved through a combination of multiple one-class classifiers and a variational AutoEncoder. The feature extraction

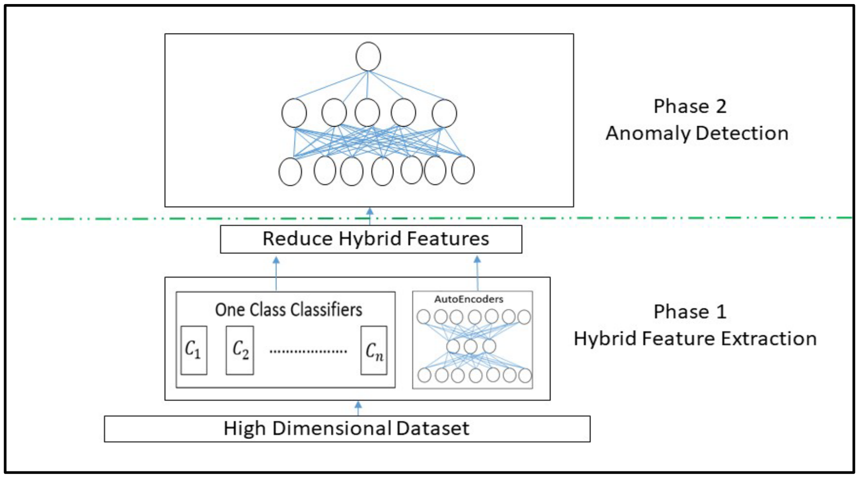


Figure 1a: MLE framework with one layer feature reduction

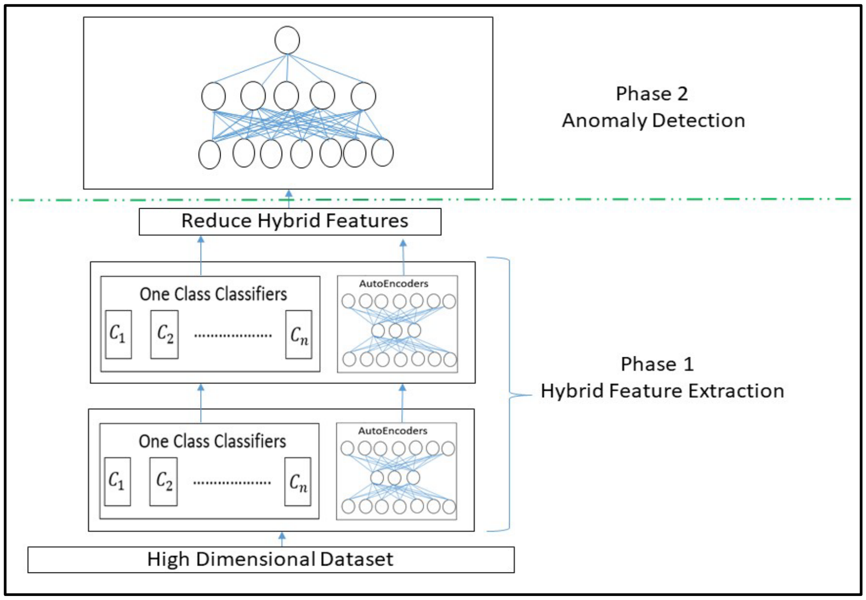


Figure 1b: MLE framework with two-layer feature reduction

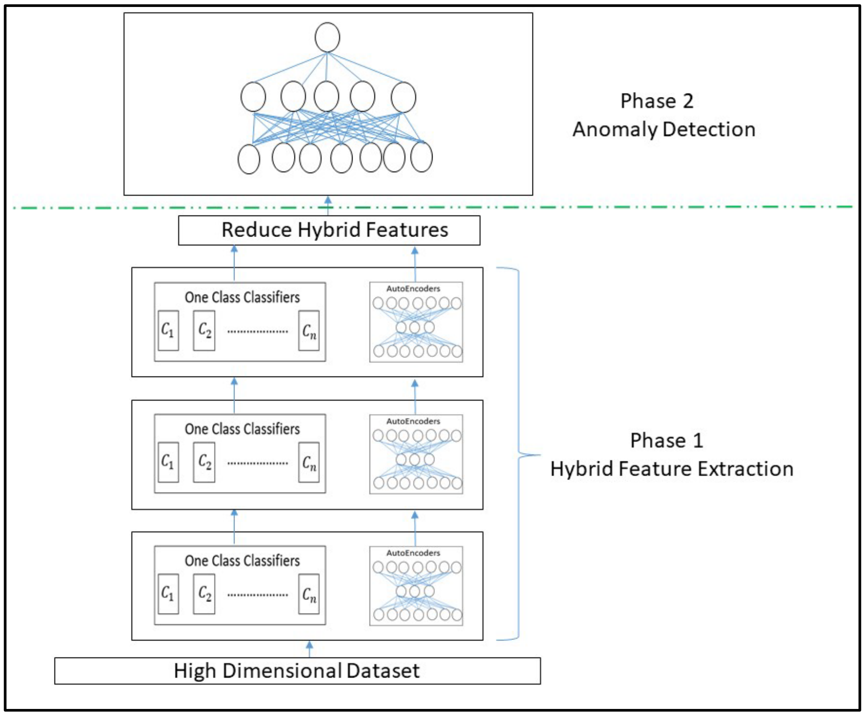


Figure 1c: MLE framework with three-layer feature reduction

mechanism is demonstrated in Figure 2 and involves the following one-class learner models: One Class Support Vector Machine (OCSVM), Isolation Forest, Mahalanobis Classifier, Local Outlier Factor, and Elliptical Envelope. The normal class data is input into each learner model ($\mathcal{L}$) to obtain anomaly scores. Each one-class classifier has distinct characteristics, and thus, we apply an adaptive weighting to each of these algorithms. After that, we implement the K-Fold cross-validation technique with a value of K set to 10. The cumulative error is calculated by determining the total number of False

Positives produced by the algorithm at each iteration.

$$Avg\ FP(\mathcal{L}_1) = \frac{\sum_{i=1}^{k} FP(\mathcal{L}_1)_k}{|Val\ Data| * k} \qquad (1)$$

Now, based on the above equation, we calculate the weight of each of the classifiers as follows:

$$Weight_{Classifier} = \ = 1 - Avg\ FP(\mathcal{L}_1) \qquad (2)$$

The output from the multiple one-class classifiers becomes one set of features.

Next, we train deep learning-based variational AutoEncoder to reduce the dimensionality of the dataset to a smaller latent space, as shown in Figure 3. This algorithm takes as input the feature set and will reduce it to a lower dimension.

Next, it will reconstruct the original feature from the compressed space. The error in reconstruction is the loss. The backpropagation algorithm is applied to update the weight and reduce the loss. We use KL Divergence loss for the backpropagation.

Thus, these hybrid sets of features are then fed to Anomaly Detector. Algorithm 1 depicts the two-step process for anomaly detection.

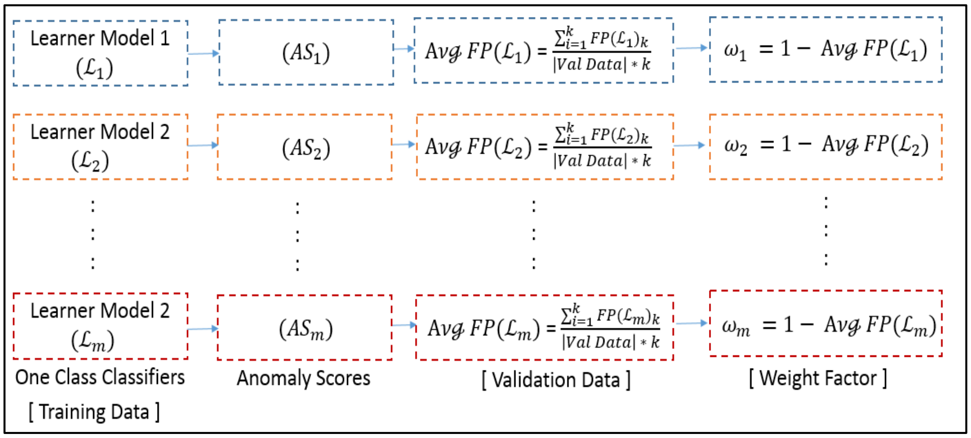

Learner Model 1 $(\mathcal{L}_1)$
(AS_1)
$Avg\ FP(\mathcal{L}_1) = \frac{\sum_{i=1}^{k} FP(\mathcal{L}_1)_k}{|Val\ Data| * k}$
$\omega_1 = 1 - Avg\ FP(\mathcal{L}_1)$
Learner Model 2 $(\mathcal{L}_2)$
(AS_2)
$Avg\ FP(\mathcal{L}_2) = \frac{\sum_{i=1}^{k} FP(\mathcal{L}_2)_k}{|Val\ Data| * k}$
$\omega_2 = 1 - Avg\ FP(\mathcal{L}_2)$
Learner Model 2 $(\mathcal{L}_m)$
(AS_m)
$Avg\ FP(\mathcal{L}_m) = \frac{\sum_{i=1}^{k} FP(\mathcal{L}_m)_k}{|Val\ Data| * k}$
$\omega_m = 1 - Avg\ FP(\mathcal{L}_m)$
One Class Classifiers
[Training Data]
Anomaly Scores
[Validation Data]
[Weight Factor]

Figure 2: One class classifier

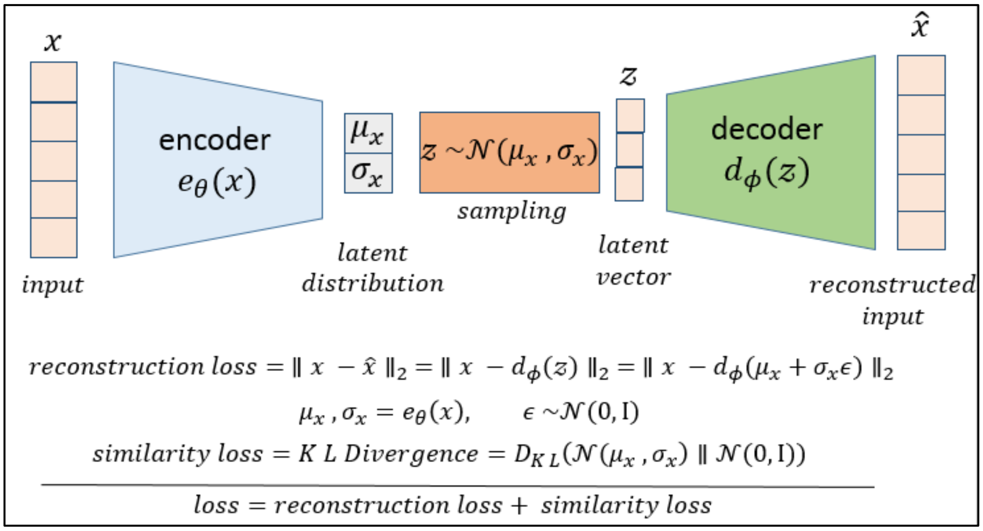

x
input
encoder $e_\theta(x)$
μ_x
σ_x
latent distribution
$z \sim \mathcal{N}(\mu_x, \sigma_x)$
sampling
z
latent vector
decoder $d_\phi(z)$
$\hat{x}$
reconstructed input
$reconstruction\ loss = \| x - \hat{x} \|_2 = \| x - d_\phi(z) \|_2 = \| x - d_\phi(\mu_x + \sigma_x \epsilon) \|_2$
$\mu_x, \sigma_x = e_\theta(x), \quad \epsilon \sim \mathcal{N}(0, \mathrm{I})$
$similarity\ loss = K\ L\ Divergence = D_{KL}(\mathcal{N}(\mu_x, \sigma_x) \parallel \mathcal{N}(0, \mathrm{I}))$
$loss = reconstruction\ loss + similarity\ loss$

Figure 3: Variational autoencoder for low dimensional embedding

```
**Algorithm 1** Multi-layered Ensemble Anomaly Algorithm
**Input**: DataSet
**Output:** Normal or Anomalous Data Points
N = Number of Rows
L = Number of Feature Reduction Layers
**for** *k* in range *0 to L* **do**
    $Classifier_{Output}$ = Train multiple One Class Classifiers on subset of N and Generate Prediction
    $FP$ = False Positives on the $Validation_{Data}$
    $Avg.FP(\mathcal{L}_1) = \frac{\sum_{i=1}^{k} FP(\mathcal{L}_1)_k}{|Val\ Data| * k}$
    $Weight_{Classifier} = 1 - Avg.FP(\mathcal{L}_1)$
    $Weighted_{Features}$ = $Classifier_{Output}$ * $Weight_{Classifier}$
    $AE_{Output}$ = Output from trained *Variational AutoEncoder*
**end for**
$Combined_{Features}$ = $Weighted_{Features}$ U $AE_{Output}$
*DNN* = Trained Neural Net on $Combined_{Features}$
**for** *i* in range *0 to N* **do**
    $Output_{DNN}$ = Prediction using *DNN* for $Data_i$
    **if** $Output_{DNN}$ > $Adaptive_{Threshold}$ **then**
        *Data point is anomalous*
    **else**
        *Data point is normal*
    **end if**
**end for**
```

4.2 Anomaly Detector

The proposed framework has a second level which comprises of a one-hidden-layer deep neural network containing 10 units. The input for this level is the hybrid features generated from the first level. The deep neural network is trained to output the probability of an observation being normal or anomalous. To determine if the incoming test data row is normal or anomalous, K-Fold Cross Validation is used to calculate the value for the dynamic threshold.

5 Experimental Result Analysis

The performance of the proposed algorithm is evaluated on two intrusion detection datasets, CIC-ID2017 and UNSW-NB15. Both datasets have unique characteristics and feature sets of varying sizes.

CIC-ID2017 dataset is a collection of 2.8 million records with 79 features released by the Canadian Institute for CyberSecurity in 2017. The dataset was generated over a period of five days and contains information on real-world network traffic, including normal and malicious traces in PCAP format.

UNSW-NB15 is a dataset created in the Australian Center for Cyber Security (ACCS) lab using the IXIA PerfectStorm tool. It consists of two million records with 44 features and provides a realistic representation of normal network activities and synthetic attack behaviors. The dataset includes nine different types of recorded attacks.

The following evaluation metrics are used:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (3)$$

$$Precision = \frac{TP}{TP+FP} \qquad (4)$$

$$Recall = \frac{TP}{TP+FN} \qquad (5)$$

$$F1 - Score = \frac{2*Precision*Recall}{Precision+Recall} \qquad (6)$$

Where TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative.

Table 1 compares the Proposed Ensemble Anomaly Detection algorithm with the other detection methods for the CIC-IDS2017 dataset.

Table 1: Metrics for CIC-IDS2017 dataset

Technique	False Positive Rate
Consolidated J-48 [10]	6.64
LIBSVM [4]	5.13
FURIA [9]	3.16
WiSarD [5]	2.86
DT-Rule [2]	1.14
MLE-Net with one layer	**0.56**
MLE-Net with two layers	**0.39**
MLE-Net with three layers	**0.45**

In reference [10], the authors utilized different resampling methods to train classification-based machine learning models that are based on the class distribution of the training data. In FURIA [9], the authors introduced a unique fuzzy rule-based classification method that learns from fuzzy rules rather than traditional ones based on unordered sets. LIBSVM [4] enhances the traditional SVM algorithm using quadratic minimization. WiSarD [5] transforms data into n-tuple patterns for training the model using tuples as inputs. The DT-Rule framework by Ahmed et al. [2] trains an ensemble of JRip, Forest PA, and REP tree models. Traditional methods mostly focus on binary classification; however, our proposed ensemble anomaly approach outperforms others with a minimum FPR of 0.56% with one layer of Feature Detector, 0.39% with one layer of Feature Detector and 0.45% with one layer of Feature Detector. Figures 4 and 5 show the evaluated metrics of our approach on CICIDS2017 compared to other models.

Table 2 shows the comparison results of the proposed Ensemble Anomaly Detection algorithm with the other detection methods for the UNSW-NB15 dataset.

Table 2: Metrics for UNSW-NB15 dataset

Technique	**False Positive Rate**
E-Max [13]	23.79
Two-level Classification [6]	15.64
Stack Ensemble [21]	8.90
GBM [22]	8.60
Proposed Approach with one layer	**4.37**
Proposed Approach with two layers	**2.93**
Proposed Approach with three layers	**3.57**

Our proposed approach has demonstrated a substantial improvement in performance compared to previous works. For example, E-Max [13] uses statistical analysis to rank attributes and employs correlation techniques to determine features, which are then used to train five different classification algorithms. Zong et al [26] use a two-level classification approach, training the model to detect majority and minority classes in the dataset. The authors of [21] propose a two-level ensemble with a feature selection method and two-level classification. Tama et al [22] use the Gradient Boosting Classifier, trained with grid search optimization techniques, but the major drawback of this approach is the lengthy training time due to the complexity of hyper-parameter optimization. Our proposed ensemble anomaly approach was found to have the lowest false positive rate (FPR) of 4.37% with one layer of Feature Detector, 2.93% with one layer of Feature Detector and 3.57% with one layer of Feature Detector. Figures 6 and 7 show the evaluated metrics of our approach on UNSW-NB15 compared to other models.

6 Conclusion

In this study, we investigate anomaly detection for datasets with highly imbalanced classes. Traditional binary and multiclass classifiers are less effective at detecting anomalies as they are only trained on labeled data. To address this, various one-class classifiers have been developed, which learn the normal behavior on the subset of the dataset by using the normal class as input. Any deviation from the normal decision boundary is considered an anomaly. However, relying on only one classifier is not sufficient for highly complex, high-dimensional real-world datasets. To tackle this, we propose a hybrid two-phase anomaly detection framework. We first train multiple one-class classifiers and an AutoEncoder algorithm at

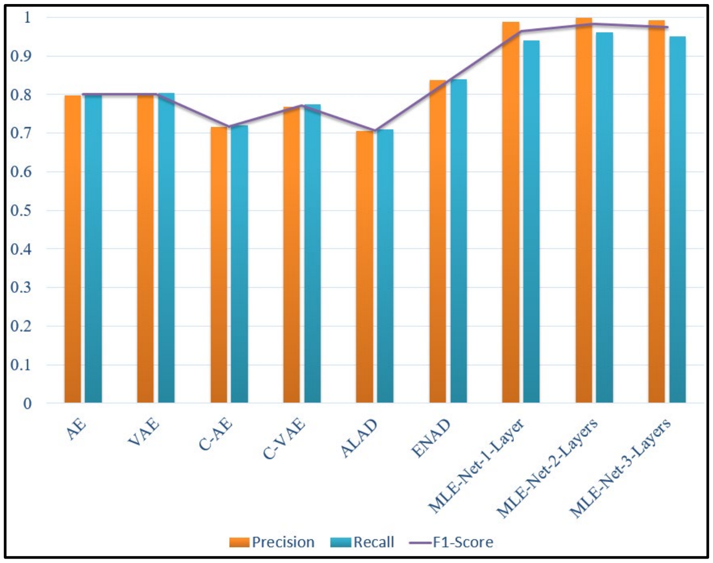


Figure 4: Evaluation metrics for CICIDS2017 dataset

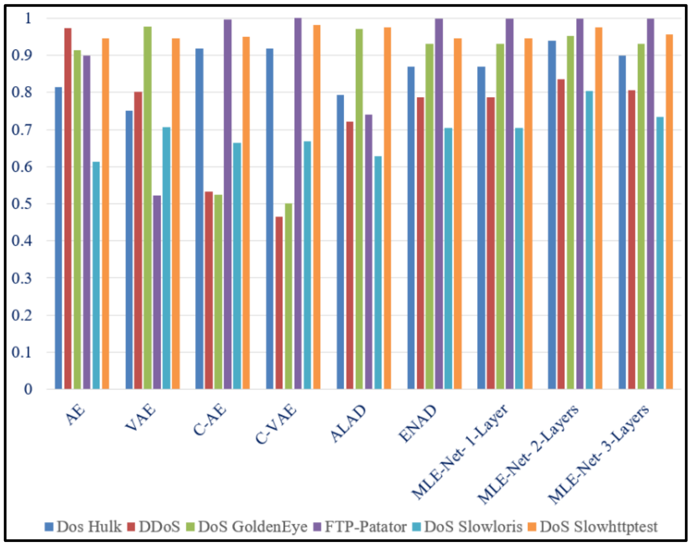


Figure 5: Accuracy for CICIDS2017 dataset

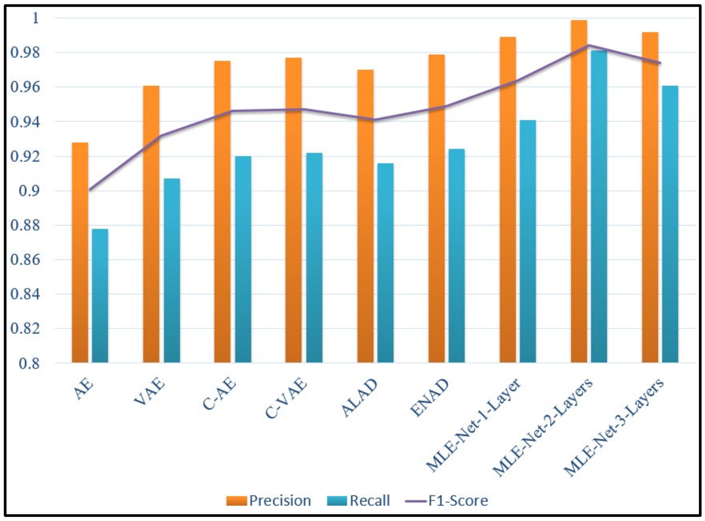


Figure 6: Evaluation metrics for UNSW-NB15 dataset

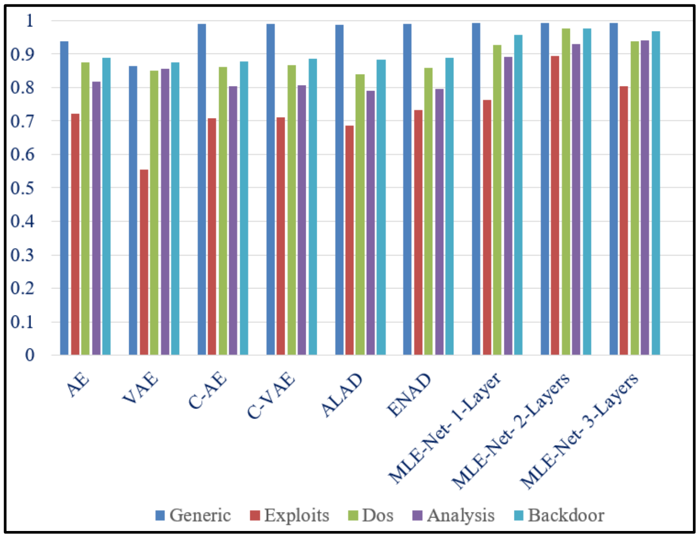


Figure 7: Accuracy for UNSW-NB15 dataset

the first phase, then apply weights to the results from each classifier. We introduced layers in the first phase and experimented three layered versions. The reduced feature sets are then passed to the second phase, which trains a deep neural network to output the probability of normal and anomalous points. Our approach was evaluated on the open-source benchmark datasets CIC-ID2017 and UNSW-NB15 and was found to have a low false-positive rate with two layers in the first phase.

References

[1] C. C. Aggarwal, "Outlier Ensembles: Position Paper," *ACM SIGKDD Explorations Newsletter*, 14(2):49-58, 2013.

[2] Ahmed Ahmim, , Leandros Maglaras, Mohamed Amine Ferrag, Makhlouf Derdour, and Helge Janicke, "A Novel Hierarchical Intrusion Detection System based on Decision Tree and Rules-Based Models," 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), IEEE, pp. 228-233, 2019.

[3] Varun Chandola, "Anomaly Detection: A Survey Varun Chandola, Arindam Banerjee, and Vipin Kumar," 2007.

[4] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):1-27, 2011.

[5] Massimo De Gregorio and Maurizio Giordano, "An Experimental Evaluation of Weightless Neural Networks for Multi-Class Classification," *Applied Soft Computing* 72:338-354, 2018.

[6] Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie, "High-Dimensional and Large-Scale Anomaly Detection using a Linear Oneclass SVM with Deep Learning," *Pattern Recognition* 58:121-134, 2016.

[7] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A Geometric Framework for Unsupervised Anomaly Detection," Applications of Data Mining in Computer Security, Springer, Boston, MA, pp. 77-101, 2002.

[8] P. Gangwani, J. Soni, H. Upadhyay, and S. Joshi, "A Deep Learning Approach for Modeling of Geothermal Energy Prediction," *Int. J. Comput. Sci. Inf. Secur.*, 18(1):62-65, 2020.

[9] Jens Hühn and Eyke Hüllermeier, "FURIA: An Algorithm for Unordered Fuzzy rule Induction," Data Mining and Knowledge Discovery, 19(3):293-319, 2009.

[10] Igor Ibarguren, Jesús M. Pérez, Javier Muguerza, Ibai Gurrutxaga, and Olatz Arbelaitz, "Coverage-Based Resampling: Building Robust Consolidated Decision Trees," *Knowledge-Based Systems*, 79:51-67, 2015.

[11] S. Joshi, H. Upadhyay, L. Lagos, N. S. Akkipeddi, and V. Guerra, "Machine Learning Approach for Malware Detection Using Random Forest Classifier on Process List Data Structure," Proceedings of the 2nd International Conference on Information System and Data Mining - ICISDM '18, pp. 98-102, 2018, https://dl.acm.org/doi/10.1145/3206098.3206113.

[12] L. McInnes, J. Healy, and S. Astels, "HDBscan: Hierarchical Density Based Clustering, *J. Open-Source Software*., 2(11):205, 2017.

[13] N. Moustafa and J. Slay, "The Evaluation of Network Anomaly Detection Systems: Statistical Analysis of the UNSW-NB15 Data Set and the Comparison with the KDD99 Data Set," *Information Security Journal: A Global Perspective,* 25(1-3):18-31, 2016.

[14] Guansong Pang, Chunhua Shen, and Anton van den Hengel, "Deep Anomaly Detection with Deviation Networks." Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 353-362, 2019.

[15] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K. R. Müller, and M. Kloft, "Deep Semi-Supervised Anomaly Detection," arXiv preprint arXiv:1906.02694, 2019.

[16] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. "Estimating the Support of a High-Dimensional Distribution," *Neural Computation* 13(7):1443-1471, 2001.

[17] J. Soni, S. K. Peddoju, N. Prabakar, and H. Upadhyay, "Comparative Analysis of LSTM, One-Class SVM, and PCA to Monitor Real-Time Malware Threats Using System Call Sequences and Virtual Machine Introspection," International Conference on Communication, Computing and Electronics Systems, Springer, Singapore, pp. 113-127, 2021.

[18] J. Soni, and N. Prabakar, "December. KeyNet: Enhancing Cybersecurity with Deep Learning-Based LSTM on Keystroke Dynamics for Authentication," International Conference on Intelligent Human Computer Interaction, Springer, Cham. pp. 761-771, 2021.

[19] J. Soni, N. Prabakar, and H. Upadhyay, "Behavioral Analysis of System Call Sequences using LSTM Seq-Seq, Cosine Similarity and Jaccard Similarity for Real-Time Anomaly Detection," 2019 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, pp. 214-219, December 2019.

[20] J. Soni, N. Prabakar, and H. Upadhyay, "EA-NET: A Hybrid and Ensemble Multi-Level Approach for Robust Anomaly Detection," Proceedings of 31st International Conference, 88:18-27, November 2022.

[21] Bayu Adhi Tama, Marco Comuzzi, and Kyung-Hyune Rhee. "TSE-IDS: A twostage classifier ensemble for intelligent anomaly-based intrusion detection system." *IEEE Access*, 7(2019): 94497-94507.

[22] Bayu Adhi Tama and Kyung-Hyune Rhee, "An In-Depth experimental Study of Anomaly Detection using Gradient Boosted Machine," *Neural Computing and Applications* 31(4):955-965, 2019.

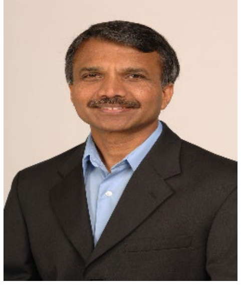
[23] D. M. Tax and R. P. Duin, "Support Vector Domain Description," *Pattern Recognition Letters*, 20(11-13):1191-1199, 1999

[24] H. Upadhyay, L. Lagos, S. Joshi, and A. Abrahao, “Big Data Framework with Machine Learning for D and D Applications – 19108,”. United States, 2019.

[25] Chong Zhou and Randy C. Paffenroth, “Anomaly Detection with Robust Deep Autoencoders,” Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 665-674, 2017.

[26] Wei Zong, Yang-Wai Chow, and Willy Susilo, “A Two-Stage Classifier Approach for Network Intrusion Detection,” International Conference on Information Security Practice and Experience, Springer, Cham, pp. 329-340, 2018.

Jayesh Soni is a Postdoctoral Associate at the Applied Research Center (ARC) of Florida International University. He received his Ph.D. degree in Computer Science from Florida International University in 2022. His M.Tech. and B.E. degrees are in Computer Science and Computer Engineering respectively. His research is in the area of Applied AI modeling and development in interdisciplinary domain. His passion for this field stems from his belief that AI has the potential to revolutionize various industries and improve the quality of life for people all over the world. His PhD thesis focused on developing novel deep learning methods for anomaly detection in sequential data. This work resulted in several publications in leading conferences and journals. In addition to developing AI models for cybersecurity, he also has a keen interest in exploring other interdisciplinary domains where AI can be applied to solve complex problems. This includes areas such as natural language processing, image and video analysis, and data privacy. By working at the intersection of AI and other fields, he aims to make meaningful contributions to both the AI and interdisciplinary communities.



Nagarajan Prabakar received the M.Eng. degree in automation from the Indian Institute of Science, Bangalore, and the PhD degree in Computer Science from the University of Queensland, Brisbane, Australia. He is currently an associate professor in the School of Computing and Information Sciences at Florida International University, Miami, USA. His research interests include machine learning-based object detection, anomaly detection for system security, and distributed optimization for real-world problems.

Himanshu Upadhyay is an Associate Professor in the Electrical and Computer Engineering at Florida International University, Miami, USA. His current research focuses on artificial intelligence/machine learning, cyber forensics, malware analysis, cyber analytics/visualization, and big data. He architected a range of tiered and distributed application system using Microsoft.Net technology to address strategic business needs, managing a team of researchers and scientists building secured enterprise information systems and mentoring undergraduate and graduate students. He has 30 years of experience in cybersecurity, artificial intelligence/machine learning, information technology, data science, management and engineering role, serving as co-principal Investigator for multimillion - dollar cybersecurity research project for the Department of Defense and Department of Energy’s Office of Environmental Management.

Leonel Lagos is currently the Associate Professor - Moss Department of Construction Management, Director of Research - Applied Research Center at Florida International University (FIU), USA. He is also the Principal Investigator of the Department of Energy-Florida International University Cooperative Agreement. Further, he is serving as the Program Director of Department of Energy-FIU Science and Technology Workforce Development Program. Dr. Lagos received a B.S. in Aerospace Engineering from the University of Florida in 1991, a Masters in Mechanical Engineering, and Ph.D. in Environmental Engineering from Florida International University (FIU) in 1996 and 2007, respectively. Dr. Lagos is an active member of DOE’s Energy Facilities Contractors Group (an advisory group supporting DOE’s environmental restoration mission) and serves as a group member in EFCOG’s D&D and Facility Engineering Working Group. Dr. Lagos also serves as an active Program Advisory Committee (PAC) member for the Waste Management Symposia organization. Dr. Lagos also serves in the Executive Committee for the American Nuclear Society’s Robotics and Remote Systems Division.

Virtual Reality: An Overview, and How to do Typing in VR

Christopher Lewis*, Frederick C. Harris, Jr.*
University of Nevada, Reno, NV, 89557, USA

Abstract

Virtual Reality (VR) has existed for many years; however, it has only recently gained wide spread popularity and commercial use. This change comes from the innovations in head mounted displays (HMDs) and from the work of many software engineers making quality user experiences (UX).

Virtual Reality (VR) has many advantages compared to traditional computer interfaces; however, there are a sizable number of deficits that VR needs to solve to be more widely adopted. Arguably, the largest of these deficits is typing within VR. In the first part of this work, we present a brief history, current research areas, and areas for improvement in virtual reality. In the second part of this work, we aim to shed some insight into successful typing methods for VR through the use of a user study and a comparison of input methods. It was found that a combination of dictation and a 3D input method led to better results than solely dictation. It was also found that testing input methods with multiple types of input culminate in more varied and detailed results.

Key Words: VR; HMD; history

1 Introduction

As virtual reality (VR) continues to explode in popularity in corporate, education, entertainment, and research fields it is increasingly important to determine how VR can be used effectively. It is well known that VR can increase a user's sense of immersion and presence in a virtual environment (VE). The benefits of VR comes somewhat inherently from the medium itself, but also from a good user experience that finds its roots in core human-computer interaction principles. This paper explains portions of why VR is important, and what research has been done on its capabilities.

VR's explosion of growth has left a distinct impression on households in the U.S. Reports [2] show that 23 percent of households have used or owned a VR headset. This figure is heavily dependent on generation, with Silent Gen, Boomers, and Gen-X having 4, 6, and 18 percent having used or owned VR headsets respectively. Gen-Y and Gen-Z have 38 and 45 percent used or owned VR headsets respectively. Monthly active users for VR in 2019 was, reportedly, at 43.1 million people while in 2020 this number shot up to 52.1 million people. It is estimated that by 2030, 23 million jobs will use augmented reality and VR with healthcare, education, and blue-collar training being the most largely impacted.

The rest of this paper is structured as follows: In Section 2 we present an overview of virtual reality. This includes a brief history of Virtual Reality in Section 2.1, some of the current research areas in Section 2.2, and some areas for improvement in Section 2.2. In Section 3 we discuss typing in VR. This starts out with Section 3.2 describing a general background for typing in VR with various input methods and user studies; Section 3.3 details the implementation of the typing methods, requirements for the application, and the organization of the experiment/user study; Section 3.5 details the results obtained from the user study including WPM, EPM, SUS scores, and user responses from the post-survey; Section 3.6 discusses the results found in Section 3.5; Section 3.7 finalizes what information this work has found and gives the opinion that the way we currently measure typing speed needs to be expanded on; and Section 3.8 details expansions to this work that would further research, allow researchers to obtain better data, or would generally allow this work to expand.

2 An Overview of VR

2.1 A Brief History

While the current state of VR is dominated by commercially available head-mounted displays (HMDs) and various peripherals, an incredible amount of effort, time, resources, and research had to be invested first. VR hasn't always predominantly used HMD's, but it is where VR started. In 1960, a cinematographer named Morton Helig would receive a US patent for an invention that could show images, emit sounds, and emit air currents that could vary in velocity, temperature, and odor. This was the first known HMD. This HMD was patented under the name of: "Stereoscopic-television apparatus for individual use" and is US patent number 2,955,156. Helig produced another notable advancement in VR called the Sensorama. The "Sensorama Simulator" was patented in 1962 and displayed 3D stereo video, stereo sound, aromas, wind, and had a seat that vibrated. These inventions mark the emergence of VR. Helig also wrote in his patent about the importance of

*Department of Computer Science and Engineering. Email: christopher_le1@nevada.unr.edu, Fred.Harris@cse.unr.edu

this technology not only in a cinematographic sense, but also in: reducing hazardous situations/training for workers and the military; teaching devices to help education institutions; and multiplayer/social situations.

The next step in virtual reality quickly appeared after the Sensorama. This step came from Ivan Sutherland in the form of the "Sword of Damocles" seen in Figure 1. This invention was the first known HMD that could rotate the user's virtual field of view in tandem with how the user is physically moving their head. Dr. Sutherland's work and accomplishments are vast and could take quite a few pages of this paper. Dr. Sutherland is credited as a pioneer of computer graphics. He received the Turing award for his PhD thesis, "Sketchpad", which was the first of its kind to use a complete graphical user interface (GUI). It also influenced, if not created, modern graphical user interfaces (GUIs) and object oriented programming. Dr. Sutherland went on to create many other notable technologies and influencing many other notable students. Dr. Sutherland created "Sword of Damocles" in 1968 with a few of his students at Harvard University. The most notable students are: Bob Sproull, the former director of Oracle Labs and current adjunct professor at the University of Massachusetts Amherst; and Danny Cohen who adopted the terminology "endianness" for computing and has been inducted into the Internet Hall of Fame. The Sword of Damocles itself actually only refers to the mechanical tracking system and not the head-mounted display itself. The Sword of Damocles is also considered the first augmented reality system as the system was somewhat translucent.

From 1970 to 1990 most VR was developed for medical simulations, flight simulations, and military training purposes. A few notable inventions did occur during this time, however. In 1979, Eric Howlett created the Large Expanse, Extra Perspective (LEEP) optical system. LEEP had a wide field of view and was added to NASA's Ames Research Center in 1985. Next, Jaron Lanier founded VPL Research in 1985 where VR peripherals were being created. The most notable VR peripherals from VPL Research were the "DataGlove" and "DataSuit". The "DataGlove", was one of the first examples of a wired glove, which acts as an input device for human-computer interaction. These gloves generally mirror what the user is doing with their hands in virtual environments, though there has been some use for wired gloves to have a robot mimic what a human wearing the gloves is doing. This "DataGlove" was then licensed to companies to make entertainment related technology, most notably the "Power Glove", which was used by Nintendo in their Nintendo Entertainment System. It didn't sell well and users notoriously had a hard time with its controls and imprecision. The "DataSuit", utilizes the "DataGloves" as well as a full body suit that is filled with sensors that can measure the movement of arms, legs, and the torso.

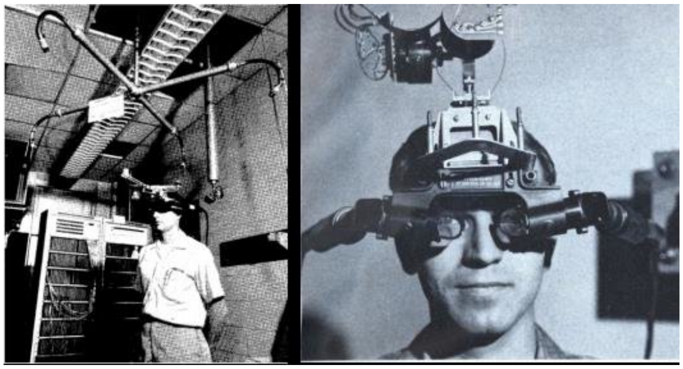


Figure 1: Ivan Sutherland's "Sword of Damocles" [28]

The 1990's saw some of the biggest changes to VR since it's inception and the seminal systems by Dr. Sutherland. One of the largest issues with VR before this point was its cost. None of these headsets were available for commercial use and they were all generally geared towards large institutions like military, medicine, or academia. In 1991 Sega announced Sega VR, which never made it to release. That same year, Virtuality launched the first mass-produced multiplayer VR systems. These systems were created for the use in VR arcades and had a cost of $73,000 per system. While not quite commercially available to the end user, this shows a distinct increase in the use of VR for entertainment and non-industry use. Again in 1991, the next large step in VR was taking place in academia.

The Cave Automatic Virtual Environment (CAVE) was a PhD thesis created by Carolina Cruz-Neira [7]. Daniel J. Sandin, a professor emeritus at the University of Illinois at Chicago, and Thomas A. DeFanti, a professor at the University of Illinois at Chicago, are also credited with creating the CAVE. The CAVE can come in quite a few forms, but the most complete CAVE is a six-sided fabric lined room using one projector and one mirror to light each side of the room, from the outside, with features from a simulated virtual environment. One example of a four-sided CAVE can be found in Figure 2.

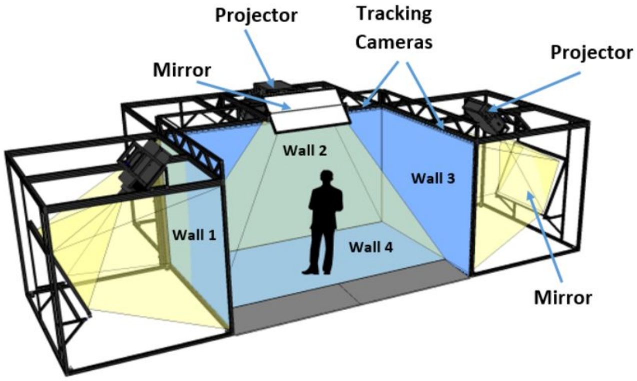


Figure 2: Annotated diagram of a four-sided CAVE system with mirrors from NASA's GRUVE Lab [19]

While this figure doesn't depict a complete six-sided CAVE, the same principles apply for any number of sides on a CAVE. This figure also shows the tracking cameras which allow the user inside of the CAVE to move around and interface with the virtual environment in various ways. Originally this technology used electromagnetic sensors to track movements, but has since moved to infrared cameras to eliminate interference common to electromagnetic sensors. This technology has been widely adopted and you can find CAVE systems, despite their high price and long setup times, at many universities and research facilities. This technology has also evolved, as of 2012, to produce the CAVE2 [8], which is based on LCD panels rather than full projection. The CAVE2 was produced by the Electronic Visualization Laboratory (EVL) at the University of Illinois, Chicago. The two most significant aspects of the CAVE2 system are that it's cost is considerably lower than the CAVE system, and it allows for a more spherical shape to the environment, which allows for much greater realism. The CAVE2 also boasts ten times the 3D resolution of the original CAVE.

Many other inventions happened after the CAVE to produce 3D graphics, but no real progress in VR devices happened until 2010, aside from large scale VR hardware in the use of theaters and amusement rides. In 2010, the prototype for the Oculus Rift was released. This headset boasted a 90-degree field of view, which wasn't previously seen before in HMD based VR. The Oculus Rift would then be shown off at E3 and Facebook, now Meta, ended up buying it for three billion USD. Meta and Oculus later got sued by Zenimax, over the Oculus on grounds of dissemination of company secrets, who won after Meta settled out of court. The next important step in HMD innovation was done by Valve who discovered, and freely shared, low-persistence displays which make smear-free HMDs for VR possible. This technology would then be adopted by all HMD manufacturers going forward. In 2014 Sony announces Playstation VR (code name Project Morpheus). In 2015 the HTC Vive would be announced and it would use tracking technology which utilized “Lighthouses” or “base stations” that use infrared light for position tracking of the VR headset and controllers. In 2015, Google announced Google Cardboard which would bring VR to a brand new audience by utilizing smartphones for VR. Going forward, almost every large tech company either had a VR HMD released or a VR/AR group at the company.

In the current era, each of these HMDs are starting to carve their own niches in the VR space. Despite these niches, most HMDs can be categorized based off of their tracking method and connection type. Tracking methods are either outside-in or inside-out. Outside-in tracking is where the HMD, and other peripherals, are being tracked by outside sensors. Some HMDs that offer this tracking are the Oculus Rift, Valve Index, HTC Vive Pro, HTC Vive Pro Eye, HTC Vive Pro 2, and the PS VR system. Inside-out tracking is where the HMD is tracked via integrated sensors that detect changes in the position of objects in the environment. This type of tracking can be done either with or without markers placed in the room. Some HMDs that offer this tracking are the HTC Vive Cosmos, Microsoft HoloLens, and Meta Quest 2. The HTC Vive Cosmos Elite is a particularly interesting VR headset due to the fact that it allows for both inside-out and outside-in tracking due it's replaceable face plates. Connection types for HMDs mean that the HMD is either connected to the PC to be able to work, or it has a standalone system that allows the headset to work without a PC attached. Most headsets are not standalone, some that are standalone are the: Meta Quest, Pico Neo 3 Link, HTC Vive Focus 3, and HTC Vive Flow. Currently, the product line with the most flexibility and diversity is done by HTC Vive, especially now that Oculus/Meta no longer produce any HMD besides the Meta Quest 2.

2.2 Current Research

Most current research is in the software and tools for VR since creating unique and usable headsets is expensive and is already being done by large manufacturers. There is quite a bit of research dedicated to reducing or better understanding virtual reality sickness (VR sickness). There are also quite sizable projects designing applications to train or educate individuals/groups.

2.2.1 VR Sickness and Health Risks. VR sickness, which is also called cybersickness, closely resembles motion sickness in terms of symptoms. This sickness comes with a few known symptoms: Nausea, balance disorder, and vomiting. VR sickness is generally understood to be caused by a disconnect between what our senses are telling us and what is actually happening. This is called sensory conflict theory and it has been used to understand motion sickness for many years, though there are many theories relating to this topic. An example of what sensory conflict theory is presenting: If a person is in a car and looks out a window, they may get motion sickness due to the fact that their eyes see a fast moving landscape, but they personally are not moving. This would explain why some people's motion sickness gets better when they look at objects in the environment that are further away, and are thus not moving as fast due to parallax. There is also a notable conflict in the literature between deciding if gender has a role to play in VR sickness with more recent research dictating that there is no significant difference [25, 34], while later research explains that women are more susceptible to VR sickness [12, 21]. It is not clear why this conflict in the literature exists, but due to the almost ten years of time difference between the studies it could be quite a number of things.

Though not currently being researched very heavily, it has been shown that VR can be problematic for a user's sense of presence and can even induce dissociation [1]. This work shows the symptoms of depersonalization and derealization had a significant increase (4.9% - 14.5%) in their thirty participants when exposed to a virtual environment in VR. These symptoms exist to some extent in every individual, but in both the cases of participants with high and low amounts of these symptoms/feelings already, there was a significant increase in these symptoms/feelings. Due to these findings: the more realistic a virtual environment, the more careful developers need to be in showing a user unsettling and graphic things as they could severely impact their users world view and sense of self outside of the real, objective, reality. Quite a bit more research should be put into this topic, in particular, if the dissociation issue should be listed as a part of VR sickness, meaning that they effect the same people in the same way. It would also be interesting to see if a number of factors change the amount of disassociation in the participants: the amount of realism, resolution of the environment, interactions available, multiplayer, ambient sound or background music, HMD differences, and/or locomotion techniques. It also isn't clear how long these symptoms last.

Lastly, there is another important health risk that users and developers of VR must be aware of. HMDs can be quite heavy at around 600g or 1.3lbs, so using them for extended duration and at a fast movement rate can be slightly dangerous. If the user's back, neck, or spine are sore do not continue to use the HMD. This disclaimer is brought about due to a very recent report of a German VR gamer breaking their C7 neck vertebra while playing VR [4]. This is the first ever case of a VR induced stress fracture, despite the sixty years of HMDs before this point. More cases are likely to pop up due to VR's growing proliferation, however.

2.2.2 Locomotion. Due to the fact that VR is a fairly new technology, a lot of work is being put into making it more usable and user friendly. One issue for VR is in typing, which is covered extensively in [14]. Another issue is in locomotion. Physical locomotion, which is the process of physically moving and having the VR system track and display the movement, by default isn't really that practical. Outside-in tracking causes this type of locomotion to be very limited due to the need to stay in range of a sensor. Inside-out tracking makes this locomotion a bit more plausible, but still relatively useless by itself as you're still tethered to a PC. The major downside to this locomotion technique not being usable is in the fact that physical bipedal walking comes the most naturally to humans and thus it makes VR sickness less likely. This is why redirected walking [23] was developed in 2001. Redirected walking rotates the environment around the user slowly and, almost, imperceptibly. The principle relies on the fact that humans will naturally auto-correct their movement in order to navigate through an environment. As they naturally auto-correct their physical rotation in order to direct themselves to an objective in their virtual environment, they create a curved physical path, thus increasing their available play space without even realizing it, as they only think they've walked in a straight path in the virtual environment. Many other publications have claimed they've improved on this concept using specific algorithms and machine learning [3, 13, 29], but the core concept remains the same.

Another physical locomotion technique is walking in place. This technique allows walking by having the user bring their legs up in a walking-like motion, but not actually move to anywhere physically. This technique is seldom used in favor of some of the other techniques listed below. There is an iteration of this technique that is used regularly, which is held-in-place walking or gait-negation. This technique utilizes a third party peripheral to hold the user in place as they walk or run. Two of these peripherals are the "Virtuix Omni-directional VR Treadmill" and the "Virtuix Omni One VR Treadmill". Both utilize a very slippery surface, to reduce friction of feet moving, and trackers attached to the user's shoes to detect movement. The way these two treadmills differ is in their holding mechanisms. The Omni-directional VR treadmill, uses a ring around the user that the user can lean against to move towards a virtual location. The ring itself is set to a specific height for each user before they get into the VR environment. As the user leans they slip and can walk or run forward towards that virtual location. The "Omni One" [32], as seen in Figure 3, holds the user by strapping them into a full vest that is suspended at a given height. This particular model reportedly allows the user to jump and crouch as well, something that the previous model did not. This treadmill is not currently out on the market yet, but they do have demos that make this treadmill seem very interesting for research going forward.

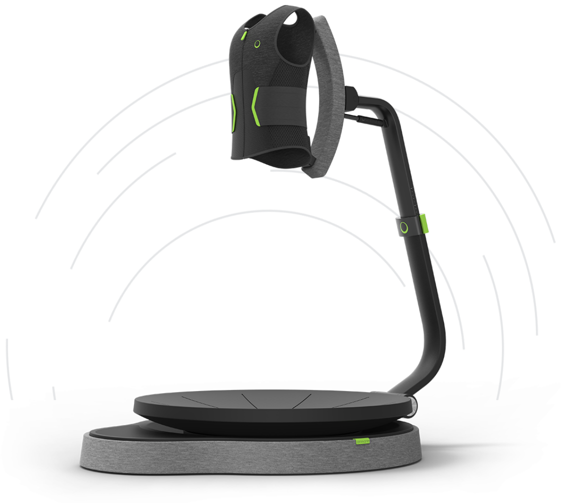


Figure 3: The new Virtuix Omni One VR treadmill [32]

There are quite a few other locomotion techniques. Joystick walking is where the user utilizes a joystick or a trackpad of some sort to move their avatar in a given direction. This approach can have six degrees of freedom due to the nature of the controllers. This approach also has a significant risk of VR sickness due to the fact that the user's perspective is moving, but the user themselves are not physically moving. This technique doesn't require the use of trackpads or joysticks, but some continuous actuator is required as well as some sort of vector. For example, the trigger on a controller could be the actuator and the rotation of the controller could give the angle of the vector, while the magnitude is fixed programmatically.

Teleportation based movement can come in quite a few forms. The three most prevalent forms are what we'll call: direct teleportation, preset teleportation, and avatar movement-based teleportation. Direct teleportation, in this case, refers to teleporting directly to the location that the user's cursor is. The cursor will generally be shot out from the tip of the user's controller and fall rapidly until it collides with the floor. This then creates a target on the ground that the user can then teleport to by pressing a button. This is by far the most common teleportation method and it is the default inside of the "SteamVR Home" [31] application (the default VR launcher on a PC). "SteamVR Home" also uses preset teleportation. This teleportation allows the user to teleport only to a specific location in the environment. These locations are either single points that allow the cursor to snap directly to them, or they are larger areas that utilize a small amount of direct teleportation. The combination of direct and preset teleportation results in areas that the user can teleport inside of, but are still defined by the developers. "SteamVR Home" utilizes all of these teleportation methods by having a preset area that the user can teleport inside of and specific points in that area that the user can teleport to in order to select and change specific aspects of the virtual environment. The last teleportation method, avatar movement-based teleportation, is quite interesting. This teleportation method is the least common, but it is a very novel approach. This approach aims to solve a social VR problem, where teleportation generally feels very off putting to the people around the user teleporting. This method animates the user's avatar and makes it walk to the location the user's cursor is focused, while keeping the user's camera fixed in their current position. When the avatar makes it to their desired location, the user can then teleport their camera to that location. This means that the user temporarily sees in third person and can watch their avatar walk to a location. This locomotion technique can be seen in the application, "VR Chat" [33], which is a social VR platform. It is important to note that none of the teleportation methods can move with the six degrees of freedom that the joystick allows, but they also have significantly less risk of VR sickness associated with them.

2.2.3 Education and Training. Education in VR has always been a key field where VR shines. In recent years, training in diversity, equity, and inclusion (DEI) have been large topics of discussion. Virtual reality has been used to further this type of training to make users more engaged and present rather than simply for compliance. Technology in this specific area has been focusing on social VR, 360° video, and speech recognition, to name a few. One DEI VR application [24] attempted this type of training and met with resounding success from participants. The users in this study responded to the training's questionnaire and the researches released the following data points: 90.8% felt moderately to completely engaged; 60.5% reported feeling somewhat present or very present during the VR experience; 94.7% of respondents agreed or strongly agreed that VR was an effective tool for enhancing empathy; 85.5% agreed that the VR experience enhanced their own empathy toward racial minorities; 18.4% reported discomfort in VR; and 67.1% stated that they believed the VR experience would change their approach to communication. These survey results are overwhelmingly positive, and they validate what corporations have started to do already, which is to train their employees in DEI using VR.

Training in VR not only has the benefit of generally being more immersive and causing trainees to feel more present in their training, it also has the benefit of allowing accurate and realistic simulations and training for dangerous situations virtually. From fire simulations based in a user's home town to training for mining and construction related difficulties, virtual reality training can help save many lives and prepare individuals more accurately than many other techniques. This is also true for training in VR for medicine. One paper, depicting the results of a randomized, double-blinded study on VR training for the operating room in gallbladder surgery [26], found that VR training was 29% faster than a traditional approach and that non-VR-trained residents were nine times more likely to transiently fail to make progress. Non-VR-trained residents were also five times more likely to injure the gallbladder or burn non-target tissue. Mean errors were also six times less likely to occur for the VR-trained group.

Indeed, the effectiveness of VR training and education can be very worthwhile for quite a few fields. However, this doesn't mean that VR is correct for every situation and field. To give more understanding as to when to use VR for training, a paper titled, "Reasons to Use Virtual Reality in Education and Training Courses and a Model to Determine When to Use Virtual Reality" [20] shows insight into this problem. This paper advocates for the use or consideration of VR when: Simulations could be used; teaching or training using the real thing is dangerous, impossible, inconvenient, or difficult; a model of an environment will teach or train as well as the real thing; interacting with a model is at least as motivating as with the real thing; Travel, cost, and/or logistics of gathering the class are too high compared to using VR; shared experiences and environments are important; the creation of the environment or model is part of the learning objective; information visualization is needed; a situation needs to be made to feel real; improving perception on specific objects; developing participatory environments and activities that can't exist in the real world; teaching tasks that involve dexterity or movement; a want to make learning more interesting and fun; accessibility for disabled people; or where mistakes in the real world would be devastating or demoralizing. This paper also describes the reasons not to use virtual reality for training/education: no substitution is possible; interactions with real objects is necessary; using a VE could be physically or emotionally damaging; using a VE can result in the confusion between reality and the VE; or VR is too expensive given the learning outcome.

3 Typing in VR

3.1 Introduction

Typing is critical in any modern computer interface. Typing is also a daily occurrence for most humans on the planet today. We type in both work and leisure environments. For some people, it is their primary form of communication. For something as critical as this, it is quite shocking that modern VR applications, generally, have little to no typing involved. When typing is involved, it is generally a slow process, especially when compared to a modern keyboard. Typing in VR often involves individually selecting letters on a virtual keyboard interface. This is a much slower process than simply typing using modern keyboards.

This work aims to speed up the typing process in VR using modern approaches and provide a methodology for reviewing input methods inside of VR. The largest addition in this project, compared to modern keyboards, is in using dictation as a primary form of input. Two input methods are created in this text. The first is using an edited form of dictation. This edited form allows the user to input any letter or, generally, any character found on a modern keyboard, as well as write whole words and sentences at a time. The second input method is a combination of an edited "drum-like keyboard" [6] and dictation. This edited form of the "drum-like keyboard" displays special characters first and allows the user to swap to an alphabetized keyboard. The reason for starting the "drum-like keyboard" with special characters selected is due to the expected use case being that the keyboard would be used to enhance dictation, not to type full sentences. The keyboard should be used to spell a single word that is hard to pronounce or hard for dictation to process. The authors created this distinction to, hopefully, facilitate understanding of when and why users might want to use the alphabetized keyboard opposed to spelling out words or acronyms using dictation.

The user study associated with this work tested for: words per minute (WPM); characters per minute (CPM); and errors per minute (EPM). The user study was broken up into four main typing challenges. These typing challenges are: URLs, Email addresses, sentences, and paragraphs. The reason these were chosen is due to the fact that they encompass almost everything the modern user of a computer might type regularly. The user study implemented a pre-test survey which collected demographic and experience data, and a post-test survey which collected general feedback and ideas for improvement. Lastly, the user study issued the System Usability Scale (SUS) for each input method, to test what the users thought about the usability of the input methods.

The rest of this section is structured as follows: Section 3.2 describes a general background for typing in VR with various input methods and user studies; Section 3.3 details the implementation of the typing methods, requirements for the application, and the organization of the experiment/user study; Section 3.5 details the results obtained from the user study including WPM, EPM, SUS scores, and user responses from the post-survey; Section 3.6 discusses the results found in Section 3.5; Section 3.7 finalizes what information this work has found and gives the opinion that the way we currently measure typing speed needs to be expanded on; and Section 3.8 details expansions to this work that would further research, allow researchers to obtain better data, or would generally allow this work to expand.

3.2 Background and Review of Literature

3.2.1 Input Methods. The classic example of VR typing methods is the "point and select" or "raycasting" method. This is a method of selecting keys on a virtual keyboard with a VR controller. The VR controller in this example sends out a raycast to the keyboard and displays itself to the user. The user would then move the controller so that the raycast hovers over a specific key and then they would press the select button to have that key input into the system. One application that uses this method is Google Earth VR [9].

Another example of VR typing methods is the "punch typing" method [35]. This is a method of selecting keys on a 3D keyboard with a VR controller. This method accomplishes typing using a collision based system. The user takes their controller, or another object that can facilitate collisions, and virtually hit the keys. The collision between the keys and the object/controller causes the key to be pressed. Keys in this method can be arrayed into a myriad of positions and can often be manually moved by the users.

The "split keyboard" or "two-thumb touchpad" [27] method is an input method for typing in VR with both controllers at the same time. Most other input methods could use both controllers at the same time, but would generally be quite uncomfortable for the user or they couldn't, generally, be used effectively. This input method aims to make use of both controllers by allowing each controller to operate distinct cursors on the virtual keyboard. This method is somewhat comparable to the "point and select" method in that they both use virtual keyboards and an, almost, cursor. However, this method is much more user friendly and much more precise. The "point and select" method is susceptible to shakes, jitters, and tremors from the user. This makes every character the user inputs a bit more of a struggle than just moving over the cursor like in the "two-thumb touchpad" method. The "point and select" method also doesn't work as well with both controllers being used to select keys as both hands would need to be extended, causing more shakes and tremors.

The "drum-like keyboard" [6] input method is almost a combination of the "point and select" method, the "punch typing" method, and the "two-thumb touchpad" method. The "drum-like keyboard" uses the VR controllers and extends a sort of drumstick (a stick with a ball on the end) out from the tips of the controllers. The 3D keyboard is displayed in front of the user where the user can then hit the keys with the drumstick, causing the key to be pressed. This allows the user to press keys with both controllers at the same time and allows for relatively quick input. The keys on the keyboard are usually at slightly different Y-coordinates based off of their row, with rows towards the user being lower than rows in the back. The keyboard itself is generally rotated towards the user as well, to allow every key to be seen at any time.

There are also a variety of input methods that do not involve the use of controllers at all. The first of these input methods is called, "dwell typing" [10]. This typing method, while not exclusive to VR, has been integrated into VR by using the middle of the HMD as the cursor that allows the user to wait/“dwell” over any key on a 2D keyboard to select that key. These approaches tend to be designed to help those with disabilities in allowing them to type without any additional and/or expensive equipment.

“Gaze typing” allows the user to focus their gaze on any key in a 2D keyboard and dwell until the key is selected. “Gaze typing” is an evolution of “dwell typing”. The most difficult part in designing these dwell based input methods is in deciding how long the user is required to dwell before making a selection. Make the dwell time too long and the WPM will go down, make the dwell time too short and the user can select incorrect keys on accident. There is quite a bit of research into what a correct dwell time is, but most research finds that it is related to the experience a user has with these systems [16]. Another interesting approach in this area is within the use of detecting the next most likely key to be pushed, and decreasing it’s specific dwell time while increasing other key dwell times [18].

3.2.2 User Studies and Comparisons. VR research is steadily increasing it’s use of user studies and comparison studies. As VR is becoming more of a main stay in both commercial and public use, the research surrounding its use is moving away from pure implementation details and moving towards user studies that test unique and helpful features.

A very relevant comparison study to this work is [5], which is a comparison study between four controller-based VR text-input techniques. These four input techniques are: Raycasting; drum-like keyboard; head-directed input; and split keyboard. Raycasting is the technique most commonly used in VR applications for text input and is analogous to the previously discussed “Point and Select” method. It involves raycasting from the tip of the VR controllers to hover over a 2D floating keyboard. The user would, most often, press the trigger button to input whatever key is being hovered over by the raycast. The drum-like keyboard is the same technique that this work is using, except its primary focus is on alphabetical keys in a QWERTY fashion. Head-directed input uses the rotation of the head to select specific keys and is analogous to the previously discussed “dwell typing” method. The cursor is in the middle of the user’s field of view and the user would hover over the key to select it. Split keyboard input uses both controller’s thumb pads to move around a cursor for each controller in a 2D keyboard that is split in half. This method is analogous to the previously discussed “two-thumb touchpad” input technique. This work found the mean WPM for each input method did not exceed twenty-one, with the drum-like keyboard being the fastest min and max WPM range. Due to this paper’s findings, this work is using the drum-like keyboard as a part of the tested input methods.

One publication that details a user study that tests a unique and helpful feature is [22]. This work details a system for continuous identification and authentication of users in VR systems. This system uses a variety of body relation and movements to allow for a somewhat accurate identification system in VR based off of very simple tasks in VR. These tasks include: pointing, grabbing, walking, and typing. It trains a random forest classifier to create the identification process. They found an accuracy for each individual task and reached a highest accuracy rate of about 40%, but it would be interesting to see what the identification accuracy rate would be if they combined all stages and all tasks for an overall combined accuracy. As headsets and other VR capture devices get more accurate and allow for further granularity of motion capture, this could be a very accurate way to authenticate users.

3.3 Implementation

3.3.1 Software Engineering. The identified functional requirements for this project are found in Table 1. The identified non-functional requirements are found in Table 2.

3.3.2 Technology. This work uses Unity [30] as it’s main development platform. We also used an HTC Vive Pro Eye [11] as the main HMD. Unity’s built-in “DictationRecognizer” was used for basic dictation. Unity’s “DictationRecognizer” is built on top of the Microsoft speech recognizer [17]. The reason the authors didn’t use Microsoft Azure’s Cognitive Speech Services SDK, despite Microsoft explaining it as generally better, is due to the financial trade-off for slightly better results. It was also identified that the free Unity “DictationRecognizer” would work for our purposes. The authors also used Unity’s XR Interaction toolkit, version 2.0.0, for basic VR setup and OpenXR support.

3.3.3 Dictation. Dictation using Unity’s “DictationRecognizer” required quite a few additions/changes to their algorithm in order for it to be used for our purposes. The goal for the dictation input method was to allow any text to be input at a user’s discretion. The way this dictation recognizer works is by generating a preliminary “DictationHypothesis” and then finalizing it into a “DictationResult” when the user is done talking. The “DictationResult” had a few complications, however.

The “DictationResult” doesn’t have contextualized output in most cases. The result would allow the user to say contractions (I’ve, Haven’t, Wasn’t, etc.), which shows contextualized output of special characters. For most special characters; however, the dictation algorithm would either not allow the special character to be placed or the special character would be placed, but the word would not. For example, if the user were to try to dictate “I care, period.” by saying, “I care comma period period”, the “DictationResult” would output as, “I care,..”. This shows that the “DictationResult” changes what was verbally said into a special character. This example also shows that there would be no way to dictate the example sentence, as any instance of the word “period” would change into the special character. This works in the same way for commas. In contrast to this, the “DictationResult” would not accept other special characters in the same way. For example, if the user were to try to dictate “#Hashtag” by saying, “hashtag hashtag” or “sharp hashtag”, or any other naming variation, the “DictationResult” would output exactly what the user said and not replace anything with a special character.

Table 1: The identified functional requirements

FR #	Functional Requirement Description	Priority
FR01	User can type any character found on a traditional keyboard	1
FR02	User can dictate any character found on a traditional keyboard	1
FR03	User can delete any character	1
FR04	Key presses should be clearly shown	1
FR05	Any input should be output to the same UI	1
FR06	Any input should be checked for validity	1
FR07	When a user finishes a stage the next stage should appear	1
FR08	The system will seamlessly switch between stages	1
FR09	When a new stage appears all input and sample text should be replaced	1
FR10	User can select and move to the testing scene	1
FR11	User can select and move to the main scene	1
FR12	User performance data should automatically be collected for each stage	1
FR13	User performance data should automatically output for each stage	1
FR14	When dictation is used the dictation hypothesis should be displayed	2
FR15	User can use dictation hypothesis as input text	2
FR16	Incorrect typing or dictation should be clearly shown	2

Table 2: The identified non-functional requirements

NFR #	Non-Functional Requirement Description	Priority
NFR01	User can use the controllers to collide with and type a key	1
NFR02	The system shall be well documented	1
NFR03	The system will have sample text to be used in each stage	1
NFR04	Keys should move down and then back up when pressed	1
NFR05	All input text will be managed by an intermediary writer	1
NFR06	The intermediary writer will put all input into the same UI	1
NFR07	The intermediary writer will validate input text against the sample text	1
NFR08	The intermediary writer will transmit to the system if input text is incorrect	1
NFR09	The system will support user interface interaction using the point and click method	1
NFR10	The system will use select-able buttons to move the user between scenes	1
NFR11	User performance data will be written to a file titled the start time for the program	1
NFR12	Every user interaction and dictation will be written to a full log with timestamps	2
NFR13	User can press a button to use dictation hypothesis as input text	2
NFR14	Input text and keys should turn red if an incorrect character has been entered	2

The “DictationResult” has another problem. Single characters can’t be input for things like acronyms, to spell out a particularly difficult word to pronounce, or a word that might not be a part of the dictation dictionary. For example, if the user were to try to dictate “ABC” the “DictationResult” would output, “hey be see”. Thus, not only does the “DictationResult” not take into account single letters, but it places spaces in between words by default.

To correct these issues, we created something called a “command phrase”. This “command phrase” consists of a “CommandWord” followed by a “statement”. This “CommandWord” is a word that dictation can recognize and is set at run-time by the user. The “CommandWord” for the user study was set to the word “command” for all users in order for the study to take less of the user’s time. The “CommandWord” can be said before any character to dictate an intended “statement”, which is just a character the user wishes to input. The “statement” can be any letter in the English alphabet or any special character found on a modern QWERTY layout keyboard. This implementation means that the “DictationResult” needed to be modified to remove the automatic swapping of periods and commas with their special character. For example of both changes, if the user were to try and dictate “I care, period.”, the user would need to dictate, “I care “CommandWord” comma period “CommandWord” period”. The “command phrase” in this example sentence is both ““CommandWord” comma” and ““CommandWord” period”. This example also shows that at any point in dictation, the user can input a “command phrase” along with the rest of the dictation. This same approach works with individual letters and other special characters. For cases with multiple synonyms, for example the special character “#”, which can be called a sharp,

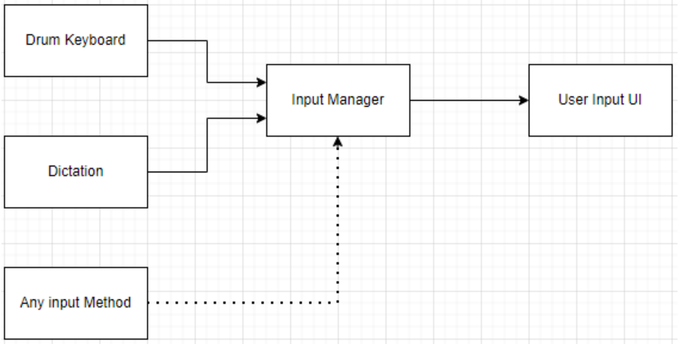


Figure 4: A block diagram detailing the basics for input method setup

pound, or hashtag, can be called by any of those synonyms and dictation will produce the special character when led by the “CommandWord”. These synonyms and wording for each letter are put into a text file that is read in at the start of the program that equates the special character with the synonym. While this approach does complicate typing, it allows a VR user to type anything they wish, albeit with a bit of practice. The largest issue with this approach is if the “DictationResult” does not output a known synonym or wording for any character. For example, if the “DictationResult” output “sea” instead of “see” for the letter “c”, and we didn’t include that as a possible synonym, then the model would not understand and just output, “ “CommandWord” sea” rather than replacing the text with “c”.

To correct the automatic spacing after each word, we implemented a variant of the “command phrase”. The variance comes from the “statement” portion of the “command phrase”, where instead of identifying a character a user would issue a “general command”. A “general command” is a word that is associated with a specific function of the input or dictation system. For example, we created a “general command” using the word “spaces”. When the user issues this “general command”, the dictation system no longer inserts automatic spaces between words or at the end of the “DictationResult”. There are a large amount of “general command”s that could be created, but we have created only what was necessary to allow the user to use dictation for any input they’d traditionally be able to use. We have created “general command”s that allow the user to backspace, insert a space, toggle capitalization of individual letters, toggle automatic spacing, and issue a full-backspace.

3.3.4 Input Methods. As seen in Figure 4, the basic setup for input methods is quite simple. Any input method that we want to include in a comparison study or a general user study sends any input it receives into an input manager. This input manager itself has three jobs. The first is to capture any and all input received and to update the user’s input text accordingly. This includes backspaces. The second job is to allow the authors to implement rather unique features that you won’t find by default on a modern physical keyboard. The example for this function is the whole word deletion function, which we call a “full backspace”. This is an input that we created in both input methods to allow the user to delete a full word. This is particularly useful with voice dictation as it can sometimes detect completely incorrect words. The third job of the input manager is to detect if the input is correct or not and then send out appropriate commands to other objects. This is used in the instance where the user makes a mistake when typing, or corrects a mistake when typing. If the user makes a mistake, the input text and their virtual keyboard (if they are using a keyboard) turn red. If the user goes back and corrects their mistake, the input text and their keyboard turn back to their normal color.

The edited “drum-like keyboard” (drum keyboard) displays special characters by default, but it is designed to be fully adaptable to any key set. A key set is the set containing the characters to be displayed, and used as input, for each key in order. When the application starts, the “key manager” sets every key in the drum keyboard to match a specific key set. The key manager can also be told at run-time to change the active key set. This feature is used to swap between special characters in a custom format, lowercase QWERTY-layout characters, and uppercase QWERTY-layout characters, though it could be used just as easily with keyboards for other languages besides English, as well as different keyboard layouts like Dvorak and Colemak.

3.3.5 Experiment Organization. The experimental organization falls into four different stages. One stage for each input type. Each stage is run through twice, once with each input method. These stages are picked at random to allow the data gathered to not be influenced by the familiarity of the input techniques and allows a less biased look at the data.

The application also has three distinct scenes for the experiment. The first scene that users see is the “Menu Scene”. This scene gives the user the choice to select between the remaining two scenes in a user interface. The mechanism for interacting with this user interface is similar to the “point and click” or “raycasting” input method described earlier in this work, except not used on a 2D keyboard. Once selected, the application will take the user into that scene. This scene can be viewed in Figure 5.

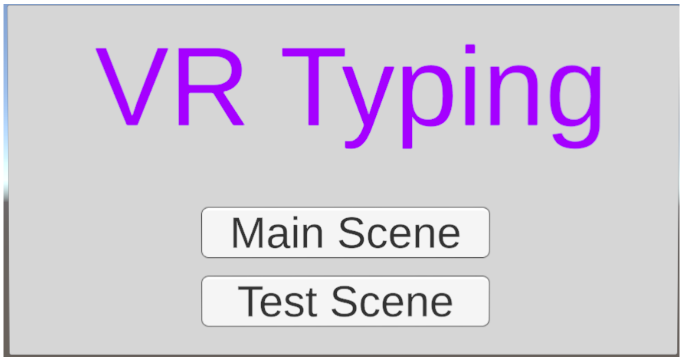


Figure 5: The menu scene

The second scene is the “Test Scene” which allows the user to use both the dictation and the drum-like keyboard input methods

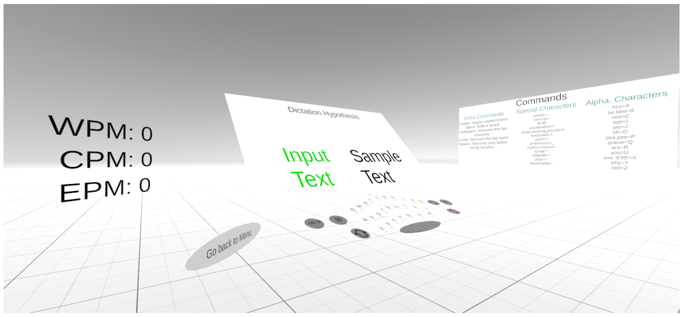


Figure 6: The testing scene

freely. This scene also displays to the user what their current words per minute (WPM), characters per minute (CPM), and errors per minute (EPM) are. Lastly, this scene displays a rotating list of sample texts that are not included in the actual experiment. This scene is to allow the user to test and play around with the input methods to get used to them before the experiment. This lowers the amount of learning that has to happen inside of the actual experiment and will allow those who aren't familiar with VR to get accustomed to it as well as the input methods. To allow the user to start the experiment, there is also a button that will take the user back to the "Menu Scene". The mechanism for pressing this button is the same as in the "drum-like keyboard". This scene can be viewed in Figure 6.

All WPM in this work is calculated using the following formula [15]:

$$WPM = \frac{|T|-1}{S} \times 60 \times \frac{1}{5}$$

where, $|T|$ is the overall resulting length of an input string, S is the number of seconds between the first and last input. $|T|$ is subtracted by 1 due to the fact that time does not start until the first input. The formula multiplies by 60 to transfer from words per second to words per minute, and it divides by 5 to transfer from characters per minute to words per minute, since a general word is considered to be 5 characters.

The third scene is the "Main Scene" this is the scene where the actual experiment takes place. This scene is very similar to the "Test Scene", except it has three main differences. The first difference is that the WPM, CPM, and EPM readings are not shown to the user and are instead averaged together per stage. Each time the user finishes a stage with an input method, this average gets output to a file and reset for the next stage/input method. This was done as a way of getting the user to try their best without any thoughts or anxiety about not performing as well as they did during the test scene. The second difference is that the return to menu button doesn't exist, so that the user can't accidentally hit it. The final difference is that the input methods rotate and the sample text is taken from a wider list of possible texts that do not include any sample text from the test scene. This scene can be viewed in Figure 7.

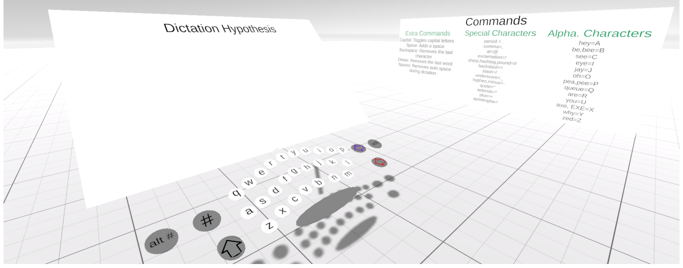


Figure 7: The main scene

3.4 Typing Text

The text that the participants had to copy involved four types of input text. This text was made out of four categories: URLs, email addresses, sentences, and paragraphs. The reasoning behind using URLs is the fact that they are very common when using a 2D typing interface, and most people are familiar with them. They also lend an interesting problem set to this process. URLs tend to need special characters, non-words, and specific abbreviations to function. This includes abbreviations like www, com, net, org, and edu. Special characters exist in URLs as well, ".", "/", and, ":". Emails also contain special characters, non-words, and abbreviations. They are also very common in the use of 2D typing interfaces. Sentences and paragraphs can contain a wider set of special characters than emails and URLs and can also test the effectiveness of typing methods for longer periods than emails or URLs. This means that both the verbal and physical input methods have to account for most/all special characters, non-words, and abbreviations. With these four input types, we believe that every use case for typing in VR and in 2D interfaces can be tested.

The sample text that is being pulled into the experiment are generated using online tools to verify consistent, or almost consistent, lengths and complexity. This text is placed into separate files to be pulled in at run-time when the user reaches any specific stage or completes the stage with one of the input methods. The input type and it's associated average character length and standard deviation is found in Table 3.

Table 3: Each input type with their associated average character length and standard deviation

Input Type	Character Length Mean (SD)
URL	10.2 (1.7)
Email	17.2 (2.2)
Sentence	37.5 (7.8)
Paragraph	305 (14.2)

3.5 Results

As seen in Table 4, both input methods implemented in this work have quite different words per minute (WPM) averages for each type of input. Paragraphs and sentences for both

Table 4: The mean WPM, with standard deviation, and WPM Range for each of the input methods in each input type

Input Method	Input Type	WPM Mean (SD)	WPM Range
Dictation	URL	7.83 (4.31)	2.47 - 17.26
	Email	5.00(3.79)	0.99 - 14.08
	Sentence	12.17(6.02)	2.414 - 19.5
	Paragraph	28.08(20.15)	10.38 - 69.11
Dictation + Drum-like keyboard	URL	7.70(4.71)	5.03 - 18.42
	Email	5.40(3.49)	2.34 - 14.45
	Sentence	18.83(13.35)	3.556 - 41.8
	Paragraph	31.69(19.28)	13.66 - 66.79

Table 5: The mean EPM, with standard deviation, and EPM Range for each of the input methods in each input type

Input Method	Input Type	EPM Mean (SD)	EPM Range
Dictation	URL	2.09 (2.02)	0.21 - 6.6
	Email	2.22(1.51)	0.6 - 5.14
	Sentence	2.41(1.78)	0.21 - 5.97
	Paragraph	3.22(2.51)	0.76 - 8.75
Dictation + Drum-like keyboard	URL	4.00(4.05)	0.22 - 13.45
	Email	3.40(3.15)	0.33 - 10.00
	Sentence	4.32(4.14)	0.55 - 10.06
	Paragraph	5.72(3.87)	1.86 - 16.38

input methods have larger WPM averages and larger standard deviations, drastically larger in the case of the dictation + drum-like keyboard input method. The email and URL input types had very similar results for both input methods. As seen in Table 5, the mean errors per minute (EPM) found for each input type are very similar to each other within each individual input method. The dictation + drum-like keyboard input method does have an average error rate and standard deviation of about twice than found with just dictation. The maximum errors per minute for this input method are also about double that of the dictation input method. The minimums are a bit different. The minimum errors per minute for the sentence and paragraph input types share the doubling found with the maximum, mean, and standard deviation; however, the email input type for the dictation + drum-like keyboard input method is about half that found in the dictation input method. The URL minimums are about the same.

The System Usability Scale (SUS) questionnaire is a measure of the usability of each system. The mean SUS score can be found in Table 6. These scores range from 0 to 100. The SUS was given to each participant twice. Once for the dictation input method and once for the dictation + drum-like keyboard input method. This table showcases that, on average, users found the combination input method of dictation + drum-like keyboard more usable. To give a better grasp on the data collected, Figure 8 and Figure 9 are included. Figure 8 showcases the SUS scores for the dictation input method. Figure 9 showcases the SUS scores for the dictation and drum-like keyboard input method.

Table 6: The mean SUS scores and SUS range

Input Methods	SUS Mean (SD)	SUS Range
Dictation	54.63 (26.54)	12.5-100
Dictation + Drum-like Keyboard	68.75 (16.81)	37.5-95

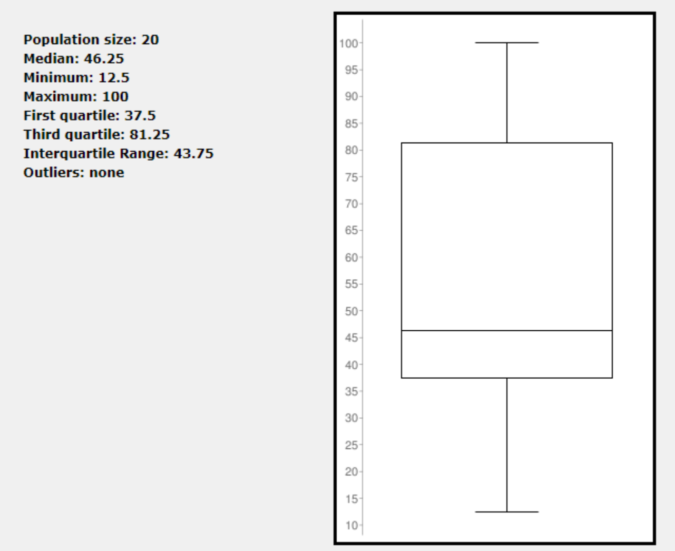


Figure 8: The box and whisker plot of SUS scores for the dictation input method

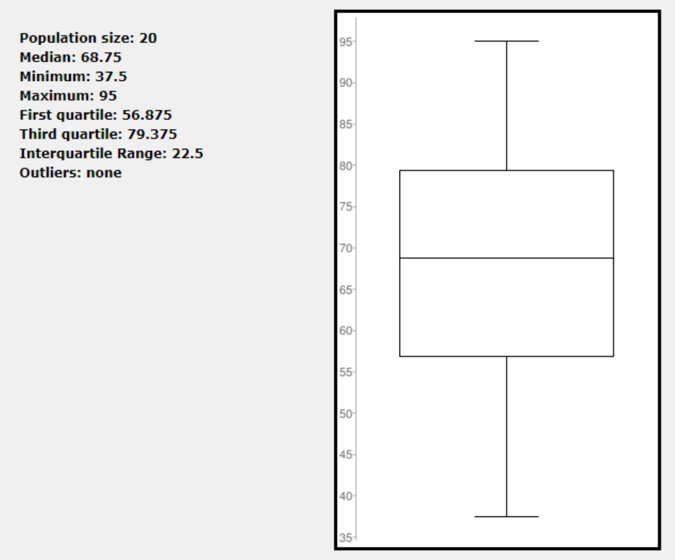
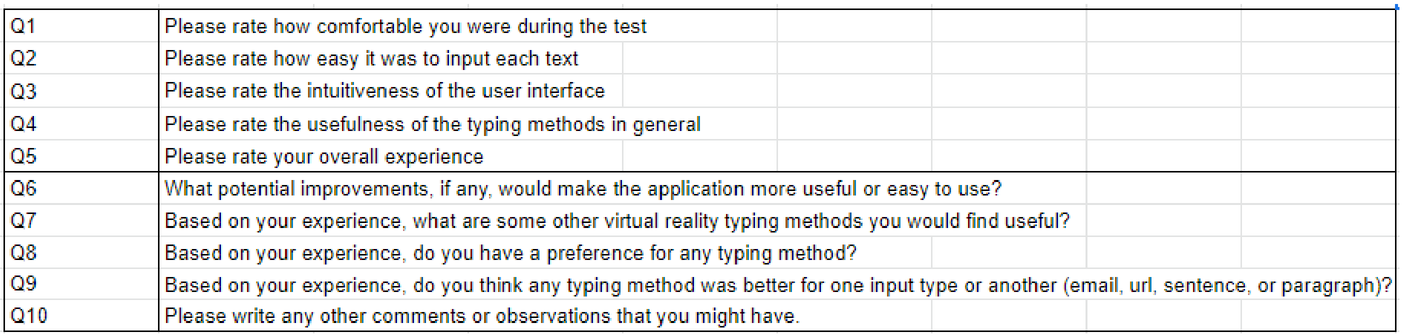


Figure 9: The box and whisker plot of SUS scores for the dictation + drum-like keyboard input method

The participants were also given a post-survey. This post-survey consisted of ten questions, which can be seen in Table 7. The first five questions were on a Likert scale from one to five. Participant responses to these five questions can be found in

Table 7: The questions asked during the post-survey



Q1	Please rate how comfortable you were during the test
Q2	Please rate how easy it was to input each text
Q3	Please rate the intuitiveness of the user interface
Q4	Please rate the usefulness of the typing methods in general
Q5	Please rate your overall experience
Q6	What potential improvements, if any, would make the application more useful or easy to use?
Q7	Based on your experience, what are some other virtual reality typing methods you would find useful?
Q8	Based on your experience, do you have a preference for any typing method?
Q9	Based on your experience, do you think any typing method was better for one input type or another (email, url, sentence, or paragraph)?
Q10	Please write any other comments or observations that you might have.

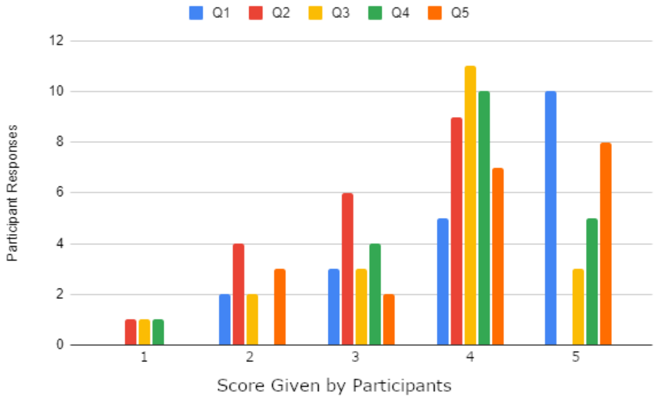


Figure 10: Participant responses for questions one through five of the post-survey

Figure 10. Users reported mostly five out of five for question one, comfortability during the test. Users reported mostly four out of five for question two, ease of text input. Users reported mostly four out of five for question three, intuitiveness of the user interface. Users reported mostly four out of five for question four, usefulness of the typing methods. Finally, users reported mostly five out of five for question five, overall experience.

Questions six to ten were free response and were then analyzed for significant statements and positive/negative sentiments for each input method and for each input method on each input type. Dictation had a total of fifteen positive statements made about it and nine negative statements. There were also eleven statements about the need to clarify or indicate some aspects of the system. The dictation method was also described as good for sentences and paragraphs, with thirteen and seventeen statements respectively. URLs and Emails received three and four positive statements respectively, for the dictation method. The drum keyboard had a total of thirteen positive statements made about it and six negative statements. There were also five statements asking for clarification or improvements in the system. The drum keyboard was described as useful for URLs and Emails, with eleven and twelve responses respectively. The drum keyboard also received two positive responses for sentences and three positive responses for paragraphs. Many of the statements in questions six to ten were responses that described the outcomes above, and many other statements gave feedback as to what needed to change and what other input methods they'd like to see.

Lastly, participants were given the Simulator Sickness Questionnaire (SSQ) to determine if typing in VR using these methods gave any excess simulator sickness. It was determined that this application does not give any excess simulator sickness, with the most common and severe symptom being eye strain and averaging to be less than mild eye strain, which is an average of less than one out of four.

3.6 Discussion

3.6.1 Demographics. Demographic data was taken directly from the pre-survey given to the participants before they entered into the VR typing application. The total number of participants was twenty. The median, and mean, age of the participants was thirty-one years of age. A total of fifteen males and five females took part in the study. The majority of participants have completed a bachelors degree. On a scale of one to five, the majority of participants were somewhat familiar with VR and responded with a three out of five or a four out of five. The exact values for this question can be seen in Figure 11. The participants were also asked, on a one to five scale, if they were familiar with motion tracking. The responses were more scattered, but most people rated a one out of five. This response shows that most participants didn't have much familiarity with typing in VR, as most VR typing methods rely on motion tracking. Lastly, the participants were asked if they were familiar with electronic entertainment and most participants rated a four, or higher, out of five, with the large majority answering five out of five.

3.6.2 Performance of Each Method. Each method performed extremely differently when comparing across input types. Many participants reported that dictation really helped with longer sentences and paragraphs, as long as they were pretty simple. The largest factor for speed in sentences and paragraphs, as well as error rate, was whether or not the dictation algorithm understood what the participant was saying. Many participants experienced unintended results due to the

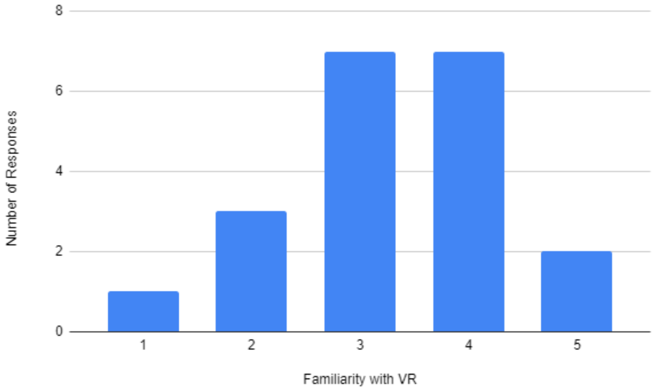


Figure 11: Participant responses for familiarity with VR

dictation results not being accurate. A large factor that hindered speed in all input types is the need to specify commands to change how the system worked. This includes commands like: "Command Spaces" and "Command Capital". A few participants reported during their post-survey that using the commands felt cumbersome, and that inputting commands fast caused the dictation result to be less accurate due to them no longer overpronouncing their dictation. Many participants also responded with the need for auto-capitalization in sentences and paragraphs. While this doesn't solve all of the problems that dictation has, it would improve accuracy, speed, and usability to a significant degree.

Participants often responded very well to the dictation and keyboard combined method. Interestingly, despite the increased usability scores and the specific mentions of enjoyment found in the post-survey, this combined input method resulted in worse errors per minute, but still faster average words per minute than the dictation method.

Another comment found in the post-survey was in the need to have more time with the systems. In particular, users found that they didn't have a large grasp of what words the dictation algorithm would insert easily and which words would require large amounts of overpronouncing. This is a common issue found with dictation algorithms as all of them, and conversational agents as a whole, can't directly tell you what they're good at. It requires a great deal of practice and trial and error to fully understand the use cases and abilities for each dictation algorithm.

Each participant was put into the test scene and were told to arbitrarily leave it when they found that they had practiced enough and had a good grasp of the input methods. This resulted in many users spending more time in the test scene than the main scene. Despite this practice; however, there was found to be no statistical correlation between time spent in the test scene and performance. For completeness, Table 8 shows the time spent in both the test and main scenes per participant, as well as average and standard deviation for time spent in both scenes.

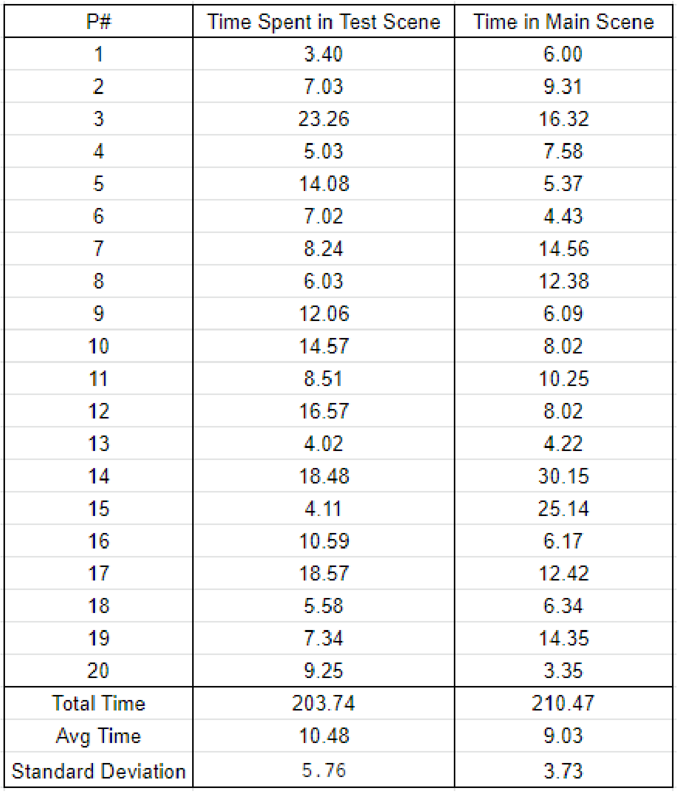
3.7 Conclusions

The dictation and drum-like keyboard input method was overall faster and more prone to errors than the dictation input method. Many users found the drum-like keyboard fun to use in short bursts, but over a larger use period the method was found to be cumbersome. This was due to the large amounts of movement required in typing anything lengthy.

Differing input types were found to change the average speed of both methods considerably. Dictation was determined by users to be better for longer strings of real words, like sentences and paragraphs. The drum-like keyboard was determined by users to be better for precise phrases or special characters.

Due to the large discrepancy found in speeds and error rates for each input type, it is our recommendation that future work regarding text input in VR should include not only sentences or phrases, as commonly found in current research, but also; paragraphs to test long form and endurance writing, and email addresses or URLs to test short input and special character insertion. More input types can be added to fully encompass everything a user might type with a traditional keyboard. Overall, the addition of more varieties of text in researching typing methods is incredibly important as VR transitions from a novelty entertainment and scientific games platform to an interface that many people can use daily in their work.

Table 8: The time (in minutes) spent in each scene per participant



P#	Time Spent in Test Scene	Time in Main Scene
1	3.40	6.00
2	7.03	9.31
3	23.26	16.32
4	5.03	7.58
5	14.08	5.37
6	7.02	4.43
7	8.24	14.56
8	6.03	12.38
9	12.06	6.09
10	14.57	8.02
11	8.51	10.25
12	16.57	8.02
13	4.02	4.22
14	18.48	30.15
15	4.11	25.14
16	10.59	6.17
17	18.57	12.42
18	5.58	6.34
19	7.34	14.35
20	9.25	3.35
Total Time	203.74	210.47
Avg Time	10.48	9.03
Standard Deviation	5.76	3.73

3.8 Future Work

Typing in VR has always been a rather slow and non-portable process. As part of the research to fix these issues, we would like to create a generalized interface system. This system would allow the user to move around the keyboard, or any other object associated with it, in 3D space complete with six degrees of freedom. This system would also allow the user to scale and disable the interface at will. This system, therefore, would be extremely useful in applications and other research that involves both locomotion and typing intermittently.

We would also like to incorporate many input methods into this research to allow a more complete observation of input methods, particularly in the case of the SUS. With a full view of the available input methods, we believe that participants of the user study would rate each method differently on the SUS.

A longitudinal study incorporating all of the input methods would indicate what the average speed of each input method actually is. Users in this study, for instance, displayed a need to use the input methods more to fully understand the methods. Some users reported that they didn't fully understand the input methods until some of the longer input types, or towards the very end of the study. Typing in 3D is not something that most users have experienced, so letting them get used to the systems is imperative.

Compound commands are when a participant would aim to give two commands at the same time to the dictation system. This was a somewhat common mistake users would feel natural in doing. These compound commands often were caused by the want/need to have uppercase or lowercase letters. The compound commands would take the form of "'CommandWord' Capital letter", where letter is a letter they wanted to input. The proper syntax in the current system to accomplish the same thing would be, "'CommandWord' Capital" + "'CommandWord' letter". The new type of capitalization system is almost analogous to a shift key opposed to the current system which is analogous to a caps lock toggle. Implementing the compound command system would severely increase spelling speed and severely lower error rates in dictation.

With the addition of input methods, we would like to incorporate all of them into a typing game/test. This VR typing game would be complete with a virtual environment, background music, locomotion, leaderboards for each input method, multiplayer competitive scenarios, and non-player character battles.

We also feel that there can be more creative ways to allow the user to type and change input types. A few examples of extra features to incorporate into the typing for the drum-like keyboard are: using the radial touch pad on the VIVE controllers to change between keysets; colliding both controllers on a single key to change the input without completely changing the keyset; allowing the user to change, create, and save keysets dynamically; and allowing the user to use the SWYPE feature, commonly found on mobile devices, using the drum keyboard as to minimize the amount of movement per word and increase user endurance. There are also creative ways to add commands to dictation that might shorten the amount of typing necessary. A good example of this is adapting the dictation algorithm to automatically structure if statements or other programming structures, thus allowing for less key strokes and faster implementation.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under grant numbers OIA-1301726, OIA-2019609, and OIA-2148788 Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Frederick Aardema, Kieron O'Connor, Sophie Côté, and Annie Taillon. "Virtual Reality Induces Dissociation and Lowers Sense of Presence in Objective Reality". In "Cyberpsychology, Behavior, and Social Networking", 13:429-435, 2010. doi:10.1089/cyber.2009.0164. PMID: 20712501.

[2] ARtillery Intelligence. "VR Usage & Consumer Attitudes, Wave VI". [online]. `https://artillry.co/artillry-intelligence/vr-usage-consumer-attitudes-wave-vi/`, Last Accessed (January 10, 2023).

[3] Mahdi Azmandian, Timofey Grechkin, and Evan Suma Rosenberg. "An Evaluation of Strategies for Two-User Redirected Walking in Shared Physical Spaces". In "2017 IEEE Virtual Reality (VR)", pp. 91-98, 2017. doi:10.1109/VR.2017.7892235.

[4] D. Baur, C. Pfeifle, and C. E. Heyde. "Cervical Spine Injury after Virtual Reality Gaming: A Case Report". *Journal of Medical Case Reports*. 15, 312. May 2021. doi:10.1186/s13256-021-02880-9.

[5] Costas Boletsis and Stian Kongsvik. "Controller-based Text-input Techniques for Virtual Reality: An Empirical Comparison". *International Journal of Virtual Reality*. 19(3):2-15, August 2019. doi:10.20870/IJVR.2019.19.3.2917.

[6] Costas Boletsis and Stian Kongsvik. "Text Input in Virtual Reality: A Preliminary Evaluation of the Drum-Like VR Keyboard". *Technologies*, 7(2). 2019. doi:10.3390/technologies7020031.

[7] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. "Surround-Screen Projection-Based Virtual Reality". In "Proceedings of the 20th annual conference on Computer graphics and interactive

techniques - SIGGRAPH '93", ACM Press. 1993. doi:10.1145/166117.166134.

[8] University of Illinois Chicago Electronic Visualization Laboratory. "CAVE2: Next-Generation Virtual-Reality and Visualization Hybrid Environment for Immersive Simulation and Information Analysis". https://www.evl.uic.edu/research/2016, Last Accessed (January 10, 2023).

[9] Google. "Google Earth VR". [online]. https://arvr.google.com/earth/, Last Accessed (January 10, 2023).

[10] John Paulin Hansen, Anders Sewerin Johansen, Dan Witzner Hansen, Kenji Ito, and Satoru Mashino. "Command Without a Click: Dwell Time Typing by Mouse and Gaze Selections". In "Interact", Citeseer. 3:121-128, 2003.

[11] HTC Corporation. "VIVE Pro Eye Specs". [online]. https://www.vive.com/us/product/vive-pro-eye/specs/, Last Accessed (January 10, 2023).

[12] R. S. Kennedy, M. G. Lilienthal, K. S. Berbaum, D. R. Baltzley, and M. E. McCauley. "Simulator Sickness in U.S. Navy Flight Simulators". *Aviat Space Environ Med*, 60:10–16. January 1989.

[13] Dong-Yong Lee, Yong-Hun Cho, and In-Kwon Lee. "Real-time Optimal Planning for Redirected Walking Using Deep Q-Learning". In "2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)", pp. 63-71, 2019, doi:10.1109/VR.2019.8798121.

[14] Christopher John Lewis. *Virtual Reality Applications and Development*. Master's Thesis, University of Nevada, Reno, Reno, NV 89557. https://www.cse.unr.edu/~fredh/papers/thesis/084-lewis/thesis.pdf Last accessed: (January 10, 2023). August 2022.

[15] I Scott MacKenzie and Kumiko Tanaka-Ishii. *Text entry systems*. Morgan Kaufmann, Oxford, England. ISBN:978-0123735911. Mar. 2007.

[16] Päivi Majaranta, Ulla-Kaija Ahola, and Oleg Špakov. "Fast Gaze Typing with an Adjustable Dwell Time". In "Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Boston, MA, USA", Association for Computing Machinery, New York, NY, USA, CHI '09. pp. 357-360, 2009. doi:10.1145/1518701.1518758.

[17] Microsoft. "Voice Input in Unity". [online]. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/voice-input-in-unity, Last Accessed: (January 10, 2023). May 2021.

[18] Martez E. Mott, Shane Williams, Jacob O. Wobbrock, and Meredith Ringel Morris. "Improving Dwell-Based Gaze Typing with Dynamic, Cascading Dwell Times". In "Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, Colorado, USA", ACM, New York, NY, USA, CHI '17. pp. 2558-2570, 2017. doi=10.1145/3025453.3025517.

[19] NASA. "GVIS - GRUVE Lab - Glenn Research Center". [online]. https://www1.grc.nasa.gov/facilities/gvis/gruve-lab/, Last Accessed (08/03/2022).

[20] Veronica S Pantelidis. "Reasons to Use Virtual Reality in Education and Training Courses and a Model to Determine When to Use Virtual Reality". *Themes in Science and Technology Education*, 2(1-2):59–70. 2010.

[21] George D. Park, R. Wade Allen, Dary Fiorentino, Theodore J. Rosenthal, and Marcia L. Cook. "Simulator Sickness Scores According to Symptom Susceptibility, Age, and Gender for an Older Driver Assessment Study". In "Proceedings of the Human Factors and Ergonomics Society Annual Meeting", SAGE Publications. 50:2702-2706, October 2006. doi:10.1177/154193120605002607.

[22] Ken Pfeuffer, Matthias J. Geiger, Sarah Prange, Lukas Mecke, Daniel Buschek, and Florian Alt. "Behavioural Biometrics in VR". In "Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems", ACM. May 2019. doi:10.1145/3290605.3300340.

[23] Sharif Razzaque, Zachariah Kohn, and Mary C. Whitton. "Redirected Walking". In "Eurographics 2001 - Short Presentations", Eurographics Association. 2001. doi:10.2312/egs.20011036.

[24] Robert O. Roswell, Courtney D. Cogburn, Jack Tocco, Johanna Martinez, Catherine Bangeranye, Jeremy N. Bailenson, Michael Wright, Jennifer H. Mieres, and Lawrence Smith. "Cultivating Empathy Through Virtual Reality: Advancing Conversations About Racism, Inequity, and Climate in Medicine". *Academic Medicine*. 95:1882-1886, July 2020. doi:10.1097/acm.0000000000003615.

[25] Dimitrios Saredakis, Ancret Szpak, Brandon Birckhead, Hannah A Keage, Albert Rizzo, and Tobias Loetscher. "Factors Associated with Virtual Reality Sickness in Head-Mounted Displays: A Systematic Review and Meta-Analysis". *Frontiers in Human Neuroscience*. December, 2019. doi:10.3389/fnhum.2020.00096.

[26] Neal E. Seymour, Anthony G. Gallagher, Sanziana A. Roman, Michael K. O'Brien, Vipin K. Bansal, Dana K. Andersen, and Richard M. Satava. "Virtual Reality Training Improves Operating Room Performance". *Annals of Surgery*. 236:458-464, October 2002. doi:10.1097/00000658-200210000-00008.

[27] Jeongmin Son, Sunggeun Ahn, Sunbum Kim, and Geehyuk Lee. "Improving Two-Thumb Touchpad Typing in Virtual Reality". In "Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, Scotland UK", Association for Computing Machinery, New York, NY, USA, CHI EA '19. pp. 1-6, 2019. doi:10.1145/3290607.3312926.

[28] Ivan E. Sutherland. "A Head-Mounted Three Dimensional Display". In "Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)", ACM Press. 1968. doi:10.1145/1476589.1476686.

[29] Jerald Thomas and Evan Suma Rosenberg. "A General Reactive Algorithm for Redirected Walking Using Artificial Potential Functions". In "2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)", pp. 56-62, 2019. doi:10.1109/VR.2019.8797983.

[30] Unity Technologies. "Unity Real-Time Development Platform: 3D, 2D VR & AR Engine". [online]. `https://unity.com/`, Last Accessed (January 10, 2023).

[31] Valve Corporation. "SteamVR Unity Plugin". [online]. `https://valvesoftware.github.io/steamvr_unity_plugin/`, Last Accessed (January 10, 2023).

[32] Virtuix. "Omni One — The Future of Gaming". [online]. `https://omni.virtuix.com/`, Last Accessed (January 10, 2023).

[33] VRChat Inc. "VRChat Hompage". [online]. `https://hello.vrchat.com/`, Last Accessed (January 10, 2023).

[34] Michael L. Wilson and Amelia J. Kinsela. "Absence of Gender Differences in Actual Induced HMD Motion Sickness vs. Pretrial Susceptibility Ratings". *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. 61(1):1313-1316, 2017. doi:10.1177/1541931213601810.

[35] Powen Yao, Vangelis Lympouridis, Tian Zhu, Michael Zyda, and Ruoxi Jia. "Punch Typing: Alternative Method for Text Entry in Virtual Reality". In "Symposium on Spatial User Interaction, Virtual Event, Canada", Association for Computing Machinery, New York, NY, USA, SUI '20. 2020. doi:10.1145/3385959.3421722.

Christopher Lewis received his BS and MS degrees in Computer Science and Engineering from the University of Nevada, Reno in 2020 and 2022 respectively. During this time he specialized in Virtual Reality as a researcher in the High Performance Computation and Visualization Lab. Since graduating, he has went on to work as an engineer for Moth + Flame, a company making high quality VR training experiences in both hard and soft skills. He has plans to return to academia for a PhD in a few years.

Frederick C. Harris, Jr. received his BS and MS degrees in Mathematics and Educational Administration from Bob Jones University, Greenville, SC, USA in 1986 and 1988 respectively. He then went on and received his MS and Ph.D. degrees in Computer Science from Clemson University, Clemson, SC, USA in 1991 and 1994 respectively.

He is currently the Associate Dean for Research in the College of Engineering and a Foundation Professor in the Department of Computer Science and Engineering and the Director of the High Performance Computation and Visualization Lab at the University of Nevada, Reno. Since joining UNR, he has worked on research projects funded by federal agencies (NSF, NASA, DARPA, ONR, DoD) as well as industry. He is also the Nevada State EPSCoR Director and the Project Director for Nevada NSF EPSCoR. He has published more than 300 peer-reviewed journal and conference papers along with several book chapters and has edited or co-edited 14 books. He has had 14 PhD students and 81 MS Thesis students finish under his supervision. His research interests are in parallel computation, simulation, computer graphics, and virtual reality. He is also a Senior Member of the ACM, and a Senior Member of the International Society for Computers and their Applications (ISCA).

Optimal Control Frequencies for Large Number of Virtual Agents in Augmented Reality Applications

Bradford A. Towle Jr.*
Florida Polytechnic University, Lakeland, FL 33805

Abstract

With the rise of augmented reality (AR) applications, more devices of different computation capabilities are employing this technology. Currently, AR applications are limited to a small number of virtual agents, but in the future, this will change. A virtual agent is any entity within the program that must periodically run logic and has some graphical effect. This journal article explores the optimal control frequencies for virtual agents across three common AR platforms and compares the results. This experiment uses a stair-step stress test and records the framerate at a 10Hz cycle. Special care has been taken to reduce extenuating factors that may consume CPU cycles, isolating the only change to the increase in virtual agents. These tests were run five times for six different frequencies on the HoloLens 1, HoloLens 2, and the Android Note 8.

Key Words: Augmented reality, AR, framerate, MRTK, unity3D, HoloLens.

1 Introduction

Augmented Reality (AR) is a promising new technology making large gains in how people interact with computers and mobile devices. Augmented reality allows a computer program to combine computer-generated and real-world content [12]. This combination is usually accomplished by adding to an image or video taken in the real world from the user's perspective. The technology is expanding rapidly with subtle integration into everyday uses, especially in mobile applications. Three main reasons for this expansion are emphasis on reality, mobile device CPU improvement, and unique visualization capabilities.

The most crucial distinction of AR is that it does not occlude or replace the user's environment. Virtual reality (VR) will enclose the user in a completely virtual environment and capture all their visual/audio senses. Augmented reality, enhances the real world, allowing the user easier interaction with people and their surroundings. This feature is most noted with social applications as people begin focusing on real-life interactions and moving away from the computer screen. Another element fueling the AR enhancement is the increase in mobile computing capabilities. Smartphones now have enough processing power to run AR applications without noticeable delays. This means every new mobile device can now be used as a platform for AR. App developers have noticed this trend and begun incorporating AR elements into their programs, thus creating a subtle shift toward the widespread incorporation of augmented reality. The last reason for AR expanding popularity is the unique capability of integrating real-world items into the program.

While not all devices are capable of this yet, many AR devices can scan the surrounding objects, allowing them to affect the program's outcome. This scanning capability, coupled with the vast visualization potential of AR, has made it popular in the medical field for training, simulation, and a visual reference for doctors.

Since AR has become popular due to the aforementioned points, large game engines, including Unity 3D and Unreal, have championed its development. While this software is well optimized, the programmers that use them only sometimes employ maximum efficiency within their code, especially compared to hand-coding a device driver from scratch. This phenomenon is especially true for control scripts of virtual agents. Virtual agents are any virtual entity that must be updated regularly and have some graphical effect on the application. This definition can include characters, user panels, or visual effects.

To date, this has not been a pressing issue, as most AR applications employ small numbers of virtual agents. However, as the field of AR expands, the number of virtual agents will also scale. This increase in virtual agents could be problematic since larger CPU usage corresponds to lower framerates; lower framerates correspond to motion and simulator sickness in both AR and VR. This paper outlines a stair-step test to determine the optimal frequency for controlling virtual agents across three different AR platforms. The paper is broken down into the following sections: related work, platform description, testing methodology, problems encountered, results, future work, and conclusion.

2 Related Work

This paper considers virtual agent to mean any projected visualization in an augmented reality application that must change, move, or adapt to outside stimuli, thus, requiring a control script to function. The term virtual agent conjures

* 4700 Research Way: ARC 2230. Email: btowle@floridapoly.edu.

mental images of brightly colored cartoon characters running around. While such virtual agents have been used in AR applications to assist with user interaction [3, 4, 6, 7, 11], not all virtual agents are 3D game characters. Many virtual agents will be informational elements that change and adapt. For example, a virtual agent may be nothing more than a label or text message that appears identifying a desired product in the grocery store [1]. Another example in the realm of health applications is visualizing different organs during surgery or education [2, 5, 10]. This visualization may need to change, update, or provide some form of response due to the user's action requiring some form of control script. Another important field in AR is bridging the gap between robotics and humans [8, 9]. These applications will employ virtual agents to represent robots or robot-human scenarios and goals. Again, these informational virtual agents will need a control script to update, move, and adapt to different input from the user.

Currently many of the augmented reality applications only focus on one or two virtual agents at once. Therefore, processing power for their control scripts is not a large concern; however, as the field grows expanding the scale of these applications, it is foreseeable that an AR application may have hundreds of virtual agents running simultaneously. Due to this expectation, the experiment described in the next section seeks to determine the best frequency to control virtual agents.

3 Platforms

Three different platforms were chosen for the stair-step agent test:

- The Android Smart Phone
- HoloLens 1
- HoloLens 2

3.1 Android Mobile Device

Mobile devices and smartphones have become ubiquitous within society. The devices can now run augmented reality applications, and game engines can build applications for these platforms. This capability means that the general public now owns a device capable of AR; therefore, its performance with a large number of virtual agents should be tested. The Android phone tested was the Samsung Galaxy Note 8. This was purposefully done to represent a newer phone, but not the newest one on the market. Any performance issues observed with this phone would indicate a potential systemic problem for AR applications on modern phones.

3.2 HoloLens 1 – (PC Architecture)

The HoloLens 1 was one of the first head-mounted-display (HMD) AR platforms. It was also one of the first to have spatial awareness. A powerful feature where the device can map the physical environment and use that to occlude virtual objects. This device uses a typical x86 PC architecture and provides the next generation of AR functionality, such as gesture recognition, spatial awareness/mapping, and accurate localization for user position. The device is fully supported by Unity 3D game engine and is a solid baseline to compare against newer HMDs.

3.3 HoloLens 2 – (ARM Architecture)

Arguably one of the most advanced HMDs available and has a generational improvement in capability compared to the HoloLens 1. It is one of the best technologies available for AR applications and uses the ARM architecture instead of the standard x86 PC architecture. Any problems observed with the Hololens 2 would indicate that current hardware is not capable of running a large number of virtual agents in an AR setting.

4 Testing Methodology

4.1 The Experiment

The experiment outlined in this paper was run on all three platforms. The program used in this experiment was written with Unity 2020.3 and used the Mixed Reality Tool Kit (MRTK 2.0). This program would create a new virtual agent at a frequency of 2 Hz for five seconds and then wait for five seconds to determine if the system was stable. The above sequence of spawning and waiting would repeat until there were 100 agents, during which the frame rate was recorded to a file at the frequency of 10 Hz. The frame rate was calculated using the unscaled delta time property to provide the most accurate values possible (Equation 1).

UnscaledDeltTime is an independent interval in seconds from the last frame to the current [13].

$$FrameRate = \frac{1}{UnscaledDeltaTime}$$

Equation 1: Equation used to calculate framerate

The program recorded the framerate every tenth of a second and kept the file writer open to minimize computational overhead. The program would only append information, never delete, or search through the file. This limitation with the file handler was explicitly done to minimize its computational load on the hardware.

The virtual agents were programmed by employing best practices with Unity 3D; however, no other optimization was done. This programming style imitated a typical game programmer and not necessarily a researcher in computer science. The reason for imitation is to ensure the test script represents a typical program written for this platform.

When a virtual agent was created, it was given a team: red or blue. The agent invoked a control function called handle update, which provided the control algorithm.

Each agent ran the same function to control themselves. However, the frequency this function invoked varied throughout the experiment, and the resulting framerates were compared. Five individual tests were administered for each following frequency:

- Update (once per frame)
- Fixed Update (20 Hz)
- 20 Hz coroutine
- 10 Hz coroutine
- 5 Hz coroutine
- 2 Hz coroutine

The control function performed the following tasks:

1. Control the nav-mesh agent.
2. Fire projectiles at the enemy team.
3. Orient and update the score panel.

The test had a large arena where there were sixteen predetermined points the virtual agents could move. If the virtual agents were within 6 centimeters of the goal, it would then randomly choose a new goal and start navigating toward it.

The nav-mesh system in Unity was used as it is a common and popular tool amongst developers. This nav-mesh path-finding system is well optimized and would likely be chosen over building a path-finding algorithm from scratch. Please note, even though the logic was updated at different frequencies, the agents still moved continuously due to the nav-mesh.

Initially, the nav-mesh agent would be the only logic the virtual agents performed. However, it is unlikely that a typical application would only have navigation for a virtual agent being the only overhead. Therefore, logic was added to determine if there was a virtual agent on the opposite team within 50 centimeters in front of it. If there were, the agent would then fire a projectile in the same direction it was facing. If the projectile struck the other virtual agent, then the score of the first would increase by one. These projectiles also had a timer on them so that they would be destroyed after one second. The rate of fire was controlled by an additional co-routine that would wait for .3 seconds before allowing the virtual agent to fire again.

Each virtual agent had a small canvas above itself in world space. This canvas displayed the current score for each virtual agent and was used to simulate a visualization load that may be required for an AR application. The control function would rotate the canvas to make the visualizations more user-friendly to ensure it was facing the camera regardless of what direction the virtual agent was moving. Typically, this would be done in the update function, but it was added to the control function to keep all tests consistent.

4.2 Testing Procedures and Data Cleanup

The testing procedure took five individual tests of the frequencies mentioned above. The user would start each test and disable the default profiler, to keep things consistent, then move outside the arena and sit down. The user's action would be constrained to look around the arena as the different virtual elements were spawned and performed their control logic. This reduction in physical movement is essential as fast or erratic movements by the user will cause the system to do extra computation to keep the virtual environment aligned with the real environment. This experiment did not intend to put the AR application under stress from user movement. After the number of agents reached 100, the test was stopped, and the framerate was collected in a comma-delimited file. The raw data was very noisy, as shown in Figure 1.

Five tests were run for each frequency and then averaged together to reduce the noise. The results were still noisy; therefore, a moving average with a sliding window of size ten was used to improve the results further (Equation 2).

$$AverageRecord_r = \frac{\sum_{Test=1}^{5} Value_{r,Test}}{5}$$

$$\forall ma \text{ where } ma = \{10 \ldots Number\ of Records\}.$$

$$MovingAverage_{ma} = \frac{\sum_{n=ma-10}^{ma} AverageRecord_n}{10}$$

Equation 2: Calculation for Moving Average

The improvement can be seen by comparing the above graph with Figure 2. Notice the noise is significantly reduced.

5 Problems Encountered

The most significant problem encountered was getting the program to deploy to the HoloLens 2 correctly. Much time was spent configuring the project and libraries to ensure the program ran successfully on the HoloLens 2. The resolution to this problem was using the older Windows Mixed Reality plugin instead of, the newer recommended OpenXR plugin. More time is needed to determine why the newer plugin did not work correctly. Once the configuration issues had been resolved, no real problems were encountered. Due to the cross-platform capabilities of Unity and MRTK, deploying on a HoloLens 1 and Android were almost seamless.

6 Results

The six frequencies were tested over Android, HoloLens 1, and HoloLens 2. Each platform mapped the average framerate per number of agents onto a graph to compare the best results. From that information, two additional tables were created. One of the tables was the device's highest number of agents at that specific frequency while remaining above 50 fps. The other table provided the last framerate recorded in the tests.

6.1 Android

The Android platform took a different philosophy than the HoloLens 1 or 2. The performance philosophy for Android was consistency. All frequencies hovered around 30 fps regardless of the number of agents. This artificial throttling of the framerate meant that up to 100 agents, the control frequency had almost no measurable impact on the device's performance. The

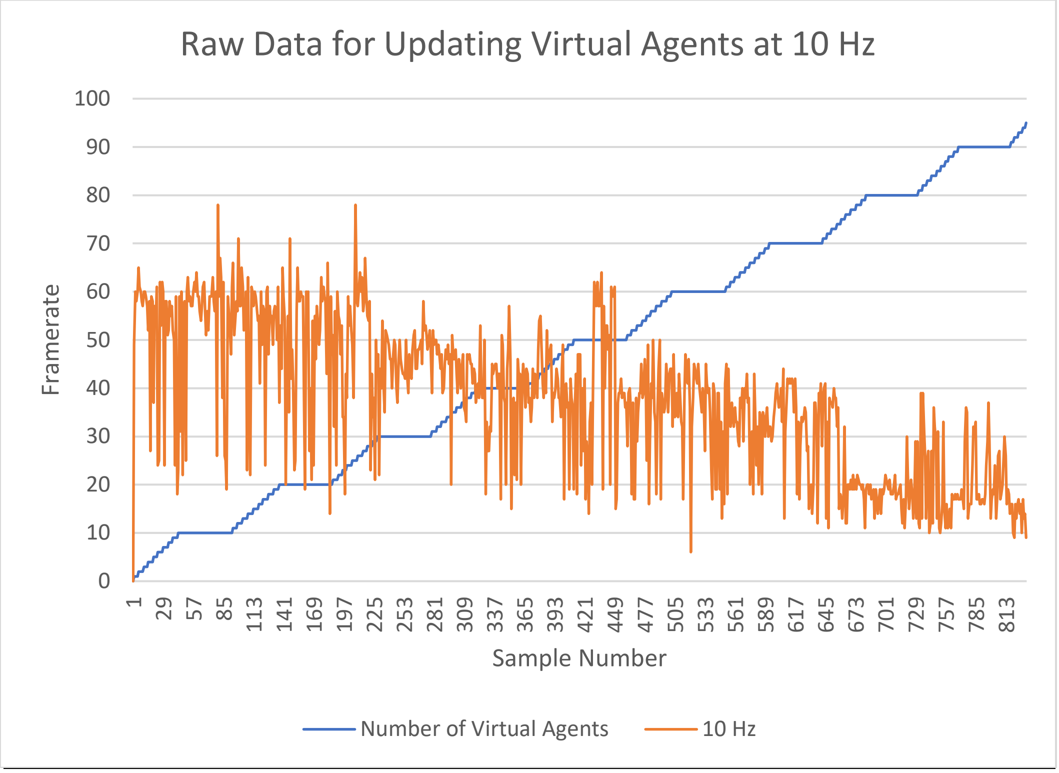


Figure 1: Raw data from 10 Hz virtual agent update experiment

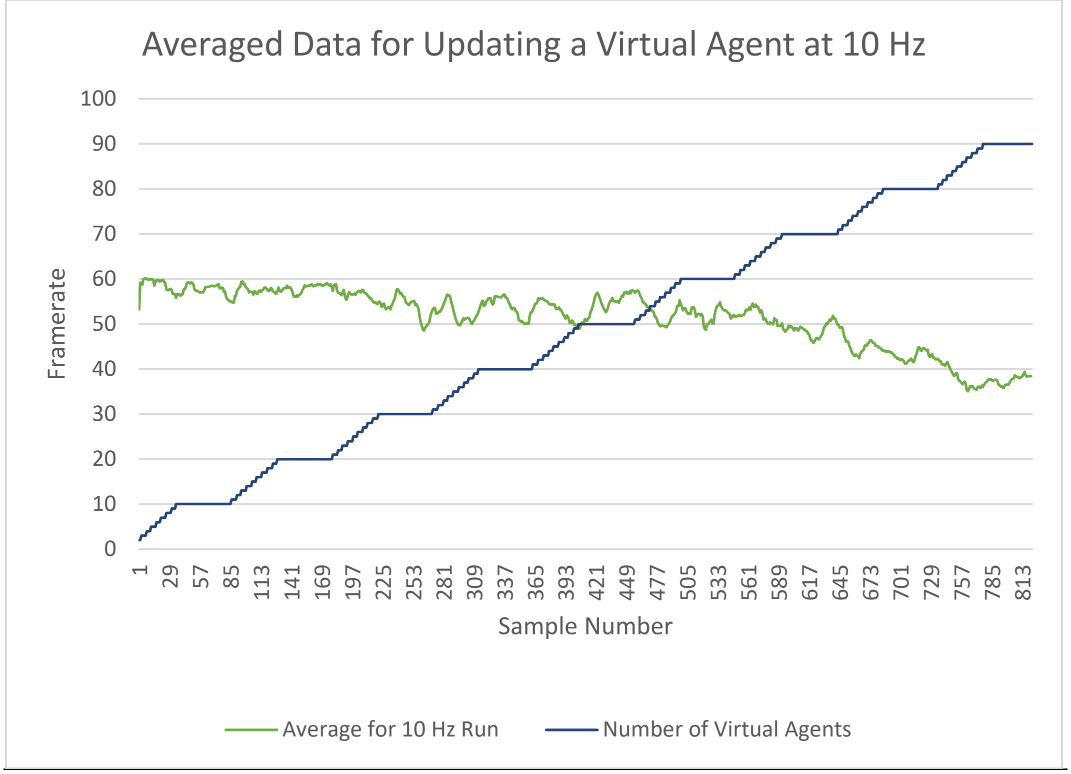


Figure 2: Averaged data for updating a virtual agent at 10 Hz

frame rate is being throttled specifically to 30 fps as the performance was the same for ten agents and 98 agents. This fact reveals two critical design considerations when building for Android. First, the platform can handle a higher number of agents before suffering from frame loss. Secondly, a developer should remember they are never going to achieve higher than 30 fps with the platform (Figure 3).

Since the frame rate never exceeded 30 fps, it was impossible to create the table reflecting the most significant number of virtual agents before falling below 50 fps. It was possible to gather the last recorded frame rate for the android. Since these values reflect a running average over multiple runs, it does indicate which frequency would be able to maintain 30 fps for the longest. Fixed update and 20Hz both had 34 fps. This result was surprising as conventional wisdom with the Unity Game Engine would suggest fixed updates would have the worst performance. Once again, this is probably due to the philosophical design approach to limit everything to 30 fps. Standard frame update and 2 Hz performed the worst (Figure 4).

6.2 HoloLens 1

The results for the HoloLens 1 revealed some noteworthy differences between itself and its subsequent version, the HoloLens 2. The HoloLens 1 held 60 fps until about 40 agents. Even more interesting is that there was very little noise (Figure 5). The HoloLens 1 held 60 fps as tightly as the Android held the 30-fps rate. The HoloLens 2 contained much more noise, and only some of the frequencies held 60 fps for controlling the first 40 virtual agents (Figure 8). However, after 40 virtual agents, the HoloLens 1 showed significant performance failure for all frequencies.

The HoloLens 1 performed the best with 10Hz (Figure 6). This result was unexpected as 2Hz would intuitively cost less CPU. As explained later, these phenomena would also extend to the HoloLens 2. In the following table (Figure 6), the 10 Hz performed the best, reaching 53 agents before falling below 50 fps. Apart from this anomaly, the results were as expected. The higher the frequency, the lower the number of virtual agents. Fixed update performing the worst due to its real-time constraint.

The table containing the final framerate recorded at each frequency is shown below (Figure 7). Once again, 10 Hz outperformed the other frequencies at a smaller margin.

Another note-worthy observation was that Unity's normal update function outperformed 20 and 2 Hz. Further analysis determined that this was due to an unintended feedback loop. As the frames-per-second drop, the number of times an update is called per second drops, thus reducing the total load.

6.2 HoloLens 2

The results for the HoloLens 2 demonstrated an iterative improvement from its predecessor. However, there were some unexpected results. The performance was stronger than the HoloLens 1 but much noisier. This noise in the performance was unexpected as the hardware is significantly more powerful. The only element that could have influenced this was the fact

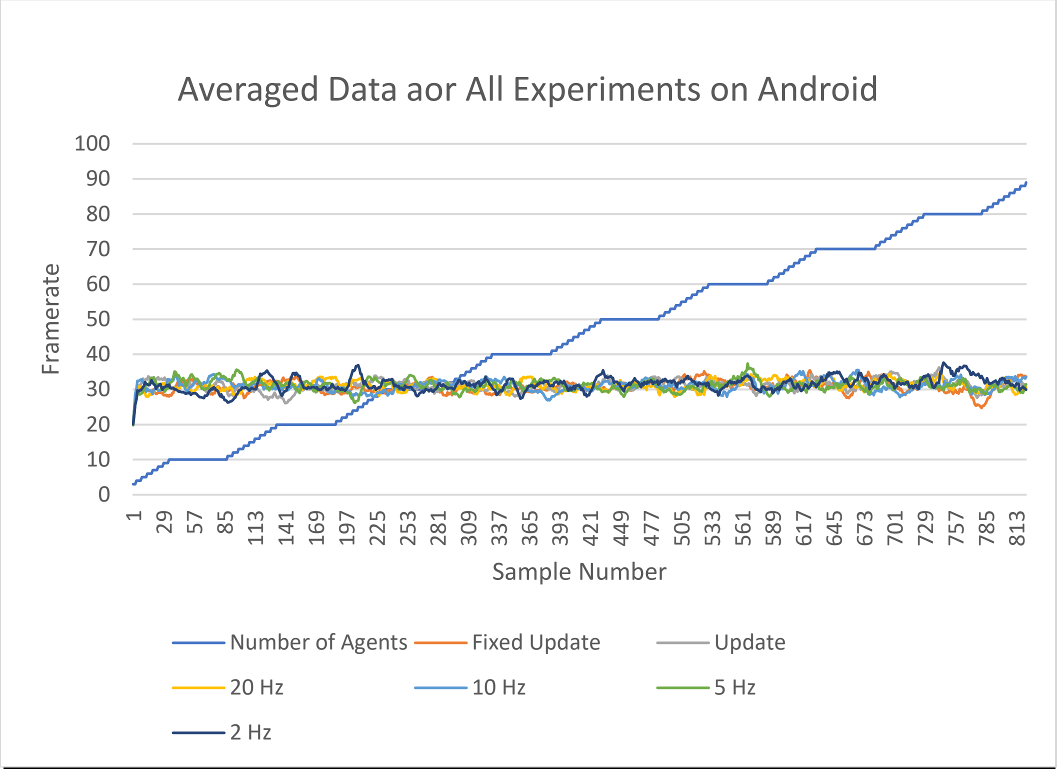


Figure 3: Averaged Data for All Experiments on Android

Control Frequency (From Best to Worst Performance)	Last Framerate for Android
Fixed Update	34
20 Hz	34
10 Hz	33
5 Hz	31
Update	30
2 Hz	30

Figure 4: Last Frame Rate for Android Platform per Control Frequency

that the HoloLens 2 uses an ARM processor, whereas the HoloLens 1 used a standard PC architecture processor (Figure 8).

Comparing the highest number of virtual agents while maintaining 50 FPS, 10 Hz significantly outperformed other frequencies. This result was an interesting trend where 10 Hz outperformed 5 and 2 Hz. The performance spread was much larger than the HoloLens 1. 2 Hz, and the Fixed update could only maintain 50 virtual agents at 50 fps. From this test, it was concluded that 10 Hz was optimal for programming control agents for the HoloLens family (Figure 9).

The last framerate recorded showed all the frequencies in the same order, except for the normal update, which surpassed 20 Hz. Again, this is most likely due to the feedback loop associated specifically with the update function (Figure 10).

7 Analysis

To conclude this research, the optimal frequencies were compared and graphed. The results below reflect both the throttled philosophy of the Android platform and the increased performance of the HoloLens 2. It is worth noting that the HoloLens 1 did maintain 60 frames-per-second longer than the HoloLens 2. However, its performance decay was quicker than the HoloLens 2 (Figure 11).

Since the Android platform did not have a higher framerate than 30 fps, the last virtual count before falling below 50 fps was compared between the HoloLens 1 and HoloLens 2. Here both tests show that 10 Hz performed the best. However, the HoloLens 1 has the smaller frequencies performing better after this, whereas the HoloLens 2, 2 Hz, is one of the lowest performing frequencies. This discrepancy in performance indicates the hardware change between the two devices (Figure 12).

The final analysis compared the final framerate for all frequencies across all platforms. On the HoloLens 1 and 2, the 10 Hz performed the best. The android fixed update and 20 Hz tied for the highest framerate. It is interesting to note that despite the hardware difference, the Android's performance at the end of the test is comparable with both HoloLens 1 and HoloLens 2.

‘

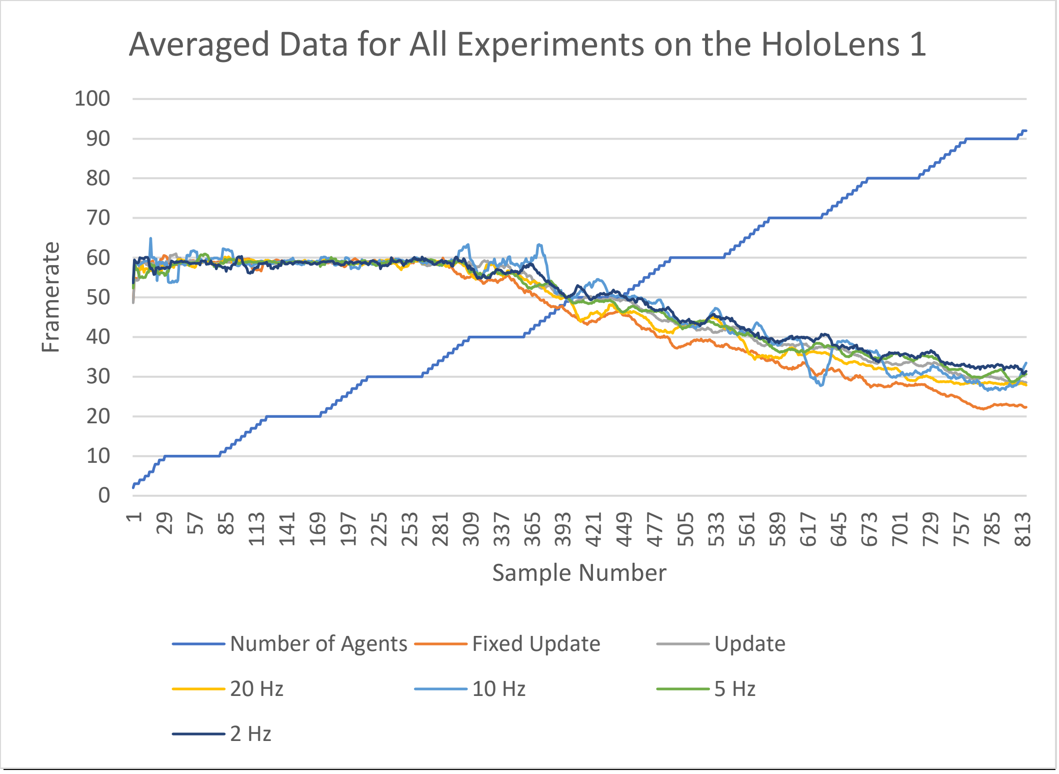


Figure 5: Averaged data for all experiments on the HoloLens 1

Frequency (From Best to Worst Performance)	Highest Number of Virtual Agents before Dropping Below 50 fps. (HoloLens 1)
10 Hz	53
2 Hz	51
5 Hz	49
20 Hz	49
Update	49
Fixed Updated	43

Figure 6: Highest number of virtual agents before falling below 50 fps

Frequency (From Best to Worst Performance)	**Last Framerate for the HoloLens 1**
10 Hz	38
5 Hz	36
Update	34
20 Hz	33
2 Hz	30
Fixed Update	26

Figure 7: Last frame rate for HoloLens 1 per control frequency

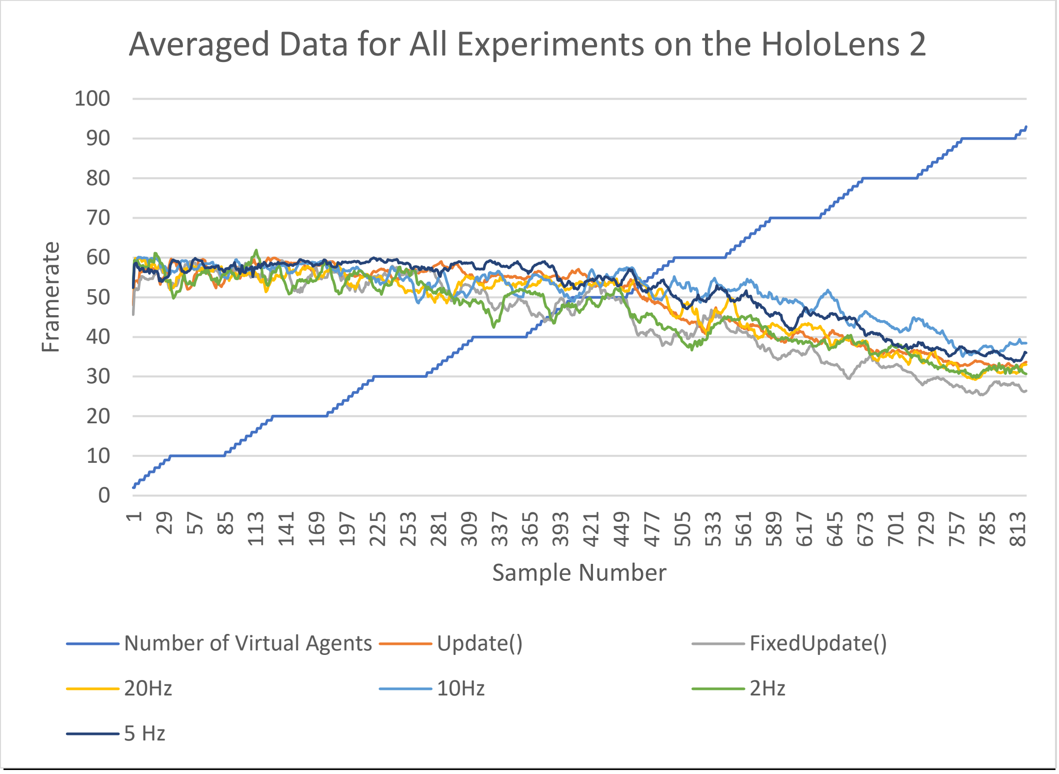


Figure 8: Averaged data for all experiments on the HoloLens 2

Frequency (From Best to Worst Performance)	**Highest Number of Virtual Agents before Dropping Below 50 FPS. (HoloLens 2)**
10 Hz	71
5 Hz	66
20 Hz	58
Update	54
2 Hz	50
Fixed Update	50

Figure 9: Highest number of virtual agents before falling below 50 fps (HoloLens 2)

Frequency (From Best to Worst Performance)	**Last Framerate for the HoloLens 2**
10 Hz	38
5 Hz	36
Update	34
20 Hz	33
2 Hz	30
Fixed Update	26

Figure 10: Last frame rate for HoloLens 1 per control frequency (HoloLens 2)

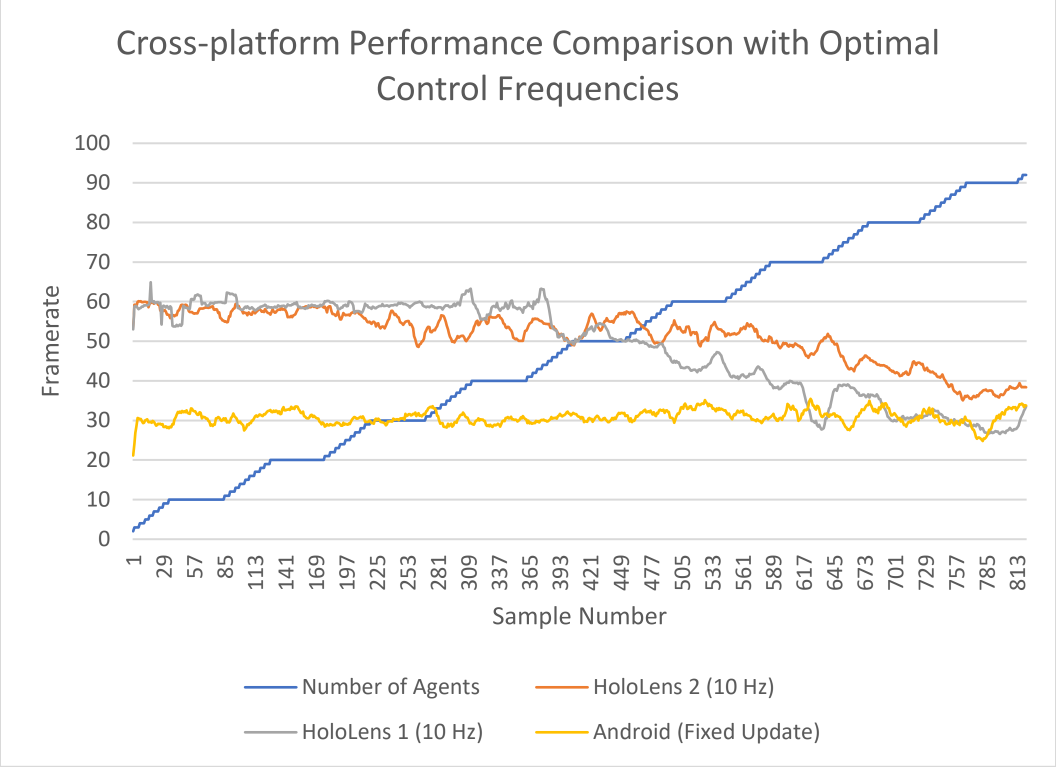


Figure 11: Comparing the optimal frequencies for all three platforms

Comparison of Highest Virtual Agent Count Before Dropping Below 50 fps		
Frequency	**HoloLens 1**	**HoloLens 2**
2 Hz	51	50
5 Hz	49	66
10 Hz	53	71
20 Hz	49	58
Update	49	54
Fixed Update	43	50

Figure 12: Highest number of agents before dropping below 50Hz

Comparison of the Platforms and the Final Framerate per Each Frequency			
Frequency	**HoloLens 2**	**HoloLens 1**	**Android**
2 Hz	30	31	30
5 Hz	36	31	31
10 Hz	38	34	33
20 Hz	33	28	34
Update	34	29	30
Fixed Update	26	22	34

Figure 13: Comparison of the platforms and the final framerate per each frequency

8 Future Work

This research created a foundation for the computational power expected from common AR devices. Several projects will benefit from this research and the knowledge of the optimal control frequencies. One research area is localizing multiple users and virtual objects without object tracking. The AR device must map the environment and combine maps from other devices to create a shared coordinate system. The computational requirement for this is immense. Another project that could benefit from this research is dynamically loading content into an AR environment. This capability would also require computational power and bandwidth to import, load, and visualize new objects that were not natively part of the application.

9 Conclusion

In conclusion, this paper presented computation stair-step tests to determine the optimal control frequency for three platforms: HoloLens 1, HoloLens 2, and an Android Phone. Surprisingly the lowest frequency did not perform the best. 10 Hz performed the best for HoloLens 1 and 2, while 20 Hz or Fixed Update performed the best on the Android. This paper also discovered a philosophical difference between the platforms. The HoloLens family would give the highest framerate possible, while the Android system kept a consistent 30 frames per second regardless of the computational load. These results will be helpful when designing AR applications in the future when considering platform constraints.

References

[1] J. Ahn, J. Williamson, M. Gartrell, R. Han, Q. Lv, and S. Mishra, "Supporting Healthy Grocery Shopping via Mobile Augmented Reality," ACM Trans Multimedia Comput. Commun. Appl., 12:16:1-16:24, 2015, doi: 10.1145/2808207.

[2] B. Garrett, J. Anthony, and C. Jackson, "Using Mobile Augmented Reality to Enhance Health Professional Practice Education," Current Issues in Emerging eLearning 4, 2018.

[3] A. Hartholt, S. Mozgai, E. Fast, M. Liewer, A. Reilly, W. Whitcup, and A. S. Rizzo, "Virtual Humans in Augmented Reality: A First Step Towards Real-World Embedded Virtual Roleplayers," Proceedings of the 7th International Conference on Human-Agent Interaction, pp. 205-207, 2019.

[4] K. Kim, L. Boelling, S. Haesler, J. Bailenson, G. Bruder, and G. F. Welch, "Does a Digital Assistant Need a Body? The Influence of Visual Embodiment and Social Behavior on the Perception of Intelligent Virtual Agents in AR," 2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), IEEE, pp. 105-114, 2018.

[5] C. Moro, Z. Štromberga, A. Raikos, and A. Stirling, "The Effectiveness of Virtual and Augmented Reality in Health Sciences and Medical Anatomy," *Anatomical Sciences Education* 10:549-559, 2017, doi: 10.1002/ase.1696.

[6] M. Obaid, I. Damian, F. Kistler, B. Endrass, J. Wagner, and E. André, "Cultural Behaviors of Virtual Agents in an Augmented Reality Environment," International Conference on Intelligent Virtual Agents, Springer, pp 412–418, 2012.

[7] M. Obaid, R. Niewiadomski, and C. Pelachaud, "Perception of Spatial Relations and of Coexistence with Virtual Agents," International Workshop on Intelligent Virtual Agents. Springer, pp 363-369, 2011.

[8] P. Parashar, L. M. Sanneman, J. A. Shah, and H. I. Christensen, "A Taxonomy for Characterizing Modes of Interactions in Goal-driven, Human-Robot Teams," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2213-2220, 2019.

[9] S. Saeedi, B. Bodin, H. Wagstaff, A. Nisbet, L. Nardi, J. Mawer, N. Melot, O. Palomar, E. Vespa, T. Spink, C. Gorgovan, A. Webb, J. Clarkson, E. Tomusk, T. Debrunner, K. Kaszyk, P. Gonzalez-De-Aledo, A. Rodchenko, G. Riley, C. Kotselidis, B. Franke, M. F. P. O'Boyle, A. J. Davison, P. H. J. Kelly, M. Luján, and S. Furber, "Navigating the Landscape for Real-Time Localization and Mapping for Robotics and Virtual and Augmented Reality," Proceedings of the IEEE 106:2020-2039, 2018, doi: 10.1109/JPROC.2018.2856739.

[10] I. C. S. da Silva, G. Klein, and D. M. Brandão, "Segmented and Detailed Visualization of Anatomical Structures based on Augmented Reality for Health Education and Knowledge Discovery," Adv. Sci. Technol. Eng. Syst J, 2:469-478, 2017, doi: 10.25046/aj020360.

[11] I. Wang, J. Smith, and J. Ruiz, "Exploring Virtual Agents for Augmented Reality," Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, ACM, Glasgow Scotland Uk, pp. 1-12, 2019.

[12] Augmented Reality - Wikipedia, https://en.wikipedia.org/wiki/Augmented_reality, Accessed 19 Nov 2022.

[13] Unity - Scripting API: Time.unscaledDeltaTime, https://docs.unity3d.com/ScriptReference/Time-unscaledDeltaTime.html, Accessed 5 Apr 2022.

Bradford A. Towle, Jr. is an Assistant Professor at Florida Polytechnic University. He has designed and coordinated the Game Design and Development concentration within the Computer Science Department since 2016. His primary research topics include augmented reality applications, autonomous robotic control architectures, and human-computer interaction. Dr. Towle researches individually and with undergraduates, hoping to foster future generations of researchers within Computer Science. He has successfully advised three graduate students helping them achieve a Master's degree in Computer Science. He is actively working to build an international reputation for his augmented reality research and has formed a student research group at Florida Polytechnic.

uMuVR: A Multiuser Virtual Reality and Body Presence Framework for Unity

Joshua Dahl*, Erik Marsh*, Christopher Lewis*, and Frederick C. Harris Jr.*
University of Nevada, Reno, Nevada, USA

Abstract

Due to the rapidly evolving nature of the Virtual Reality field, many frameworks for multiuser interaction have become outdated, with few, if any, designed to support mixed virtual and non-virtual interactions. We have developed a framework that lays an extensible and forward-looking foundation for the development of mixed interactions based upon a novel method of ensuring that inputs, visuals, and networking can all communicate without needing to understand the others' internals. This framework also provides utilities for representing user avatars in a physicalized manner while supporting a range of different input methods. We tested this framework in the development of several applications and show that it can easily be adapted to support application requirements it was not originally designed for.

Key Words: Graphics, virtual reality, body presence, multiplayer, networking, neural networks, physics simulation, boneworks

1 Introduction

Currently, there are very few multiuser Virtual Reality (VR) frameworks available in the literature. Likewise, much of the formalized work on interactions between multiple VR and non-VR users remains in its infancy, while very few people are investigating ways of portraying a user's entire body in VR without the need for additional trackers. Novotny et al. [35] is a notable exception to the first issue, providing a framework for multiuser VR development. However, the VR field is rapidly evolving and many previous works have become obsolete as newer standards and methodologies emerge. uMuVR [12] (pronounced You-Mover) serves as a total overhaul of the framework designed by Novotny et al. with major emphasis put on supporting the OpenXR [44] standard as well as the newly emerging UltimateXR [50] framework and reworking the core concepts behind Novotny et al. to be easily extendable, with an eye toward future support for non-VR users as well. Towards this aim, we have developed a novel method of ensuring that inputs, visuals, and networking can all communicate without needing to understand each other thus allowing each of the before-mentioned systems to be replaced without disrupting the others. We have also developed some innovative techniques for representing users entire bodies (including their feet) in virtual reality in a physically interactable fashion. The literature lacks not only in facilities for supporting multiuser VR interactions, but also in predicting the position of their feet and legs without explicit sensor data.

The rest of this paper is structured as follows: Section 2 reviews some of the existing literature and details several of the frameworks we are using. Section 3 then dives into library choices that we made, with particular emphasis placed upon comparisons to existing networking and compression libraries. Section 4 details the design and implementation decisions of the networking aspects of uMuVR. Section 5 explores our novel foot prediction algorithm and physicalized interaction system as well as several of the more traditional techniques we used to portray the users' upper bodies. Section 6 details two applications we implemented with uMuVR, showing how easily the framework can be extended; and finally Section 7 wraps up the paper with conclusions and plans for future work.

2 Background

There are very few multiuser VR frameworks available in the literature. This fact exists in stark contrast to the fact that many simulations [3], educational experiences [24], and entertainment experiences [40] are now being developed for VR. Thus facilities to ease this development would be beneficial.

Alternatively, much work has been done in the field of Virtual Body Presence. Currently, this work is based heavily upon Skeleton based Forward Kinematics and Inverse Kinematics (IK) techniques; Lewis et al. [30] and Aristidou et al. [5] respectively provide overviews of these two foundational topics.

*Department of Computer Science and Engineering. Emails: joshuadahl@nevada.unr.edu, erik.i.marsh@gmail.com, christopher_le1@nevada.unr.edu, fred.harris@cse.unr.edu

Currently, the literature suggests that being in a virtual environment where we can perceive the full bodies of others to lead us becoming more accepting of a complete body of our own [29]. Although there is a bit of contention to this theory [20], the literature generally seems to agree on a statement similar to the one above. However, it has also been theorized that in action-packed experiences (or types likely to invoke a "flow state") we are prone to focus more on the enemy or obstacle presented and less on ourselves [31].

Counter-arguments aside, little work has been done on analyzing the effects self interactions have on body presence, despite the fact that work has been put into non-contact-based interactions; work which seems to indicate that the presence of a full emotive body enhances facial expressiveness [28]. However, for all of these, either the legs were ignored [20, 31, 28] or tracked using additional sensors [29] using methods similar to the ones described by Caserman et al. [6]. The literature generally seems to indicate that more tracking leads to better acceptance, and yet the entire lower body is either ignored or handled with sensor arrays the average consumer may not have.

As far as the present authors are aware, there has been minimal, if any, work done by academia on predicting points on the body such as the feet and legs without the need for additional trackers the average consumer may not have. That being said, preliminary work is being done in this area outside of academia [9, 43], but the field as a whole is still in its infancy.

While leg and foot prediction may be in its infancy, physicalized interaction has largely been driven by the video game studio STRESS LEVEL ZERO [43] and their games BONEWORKS and BONELABS. While in academia work has been done on physicalized hand interaction by a chain of authors starting with Nasime and Kim [33] and (currently) ending with Delrieu et al. [13]. Additionally, libraries providing support for a system similar to the one proposed by Nasime and Kim are available for Unity [18], which additionally provide proprietary extensions to the whole body. However, there is not currently a unified framework tying all of these technologies together.

2.1 Unity

Unity [47] is a commercial game engine with extendable scripting in the C# language. It utilizes the Object-Oriented Composition paradigm [19] that was common in game engine architecture from the last decade, where component classes implement various types of encapsulated behavior that can then be attached to container objects. This extensible model is interfaced with using a visual environment editor with the ability to visually change properties and create references to other objects in the environment; thus using code to walk the engine's object hierarchy to find references is uncommon. Since uMuVR is tightly coupled to the Unity ecosystem, migrating it to another game engine would prove to be nontrivial.

Most proprietary libraries for Unity are distributed as precompiled shared objects. Unity supports a wide variety of platforms, many of which are not supported by the shared objects provided by this classification of libraries. This is of extra significance due to the recent trend towards standalone VR headsets, many of which run the Android [36] operating system which provides a very different set of facilities when compared to a standard desktop environment. Thus we have striven to avoid such resources as much as possible, instead utilizing free and open source alternatives that we can compile to any platform ourselves.

2.2 VR Frameworks

The main criteria used for choosing between the many available VR frameworks was the number of platforms they support. Additionally, we only considered frameworks that were open source or included with Unity. Historically, the two leading VR frameworks used with Unity were SteamVR and the Oculus SDK. The standard SteamVR implements, named OpenVR, has been deprecated in favor of OpenXR [44], and the Oculus SDK only supports hardware created by Meta. While both platforms provide a range of useful utilities such as an extensive interactables library (providing buttons, levers, etc...) and positioning appropriate controllers to indicate the location of your hands, neither SteamVR nor the OculusSDK are ideal considering the current rapid proliferation of VR devices.

OpenXR is an open standard from the Khronos group (the same group that maintains the OpenGL and Vulkan standards) that acts as an abstraction layer between applications and a large selection of popular eXtended Reality (XR) devices (an umbrella term used to describe both Virtual and Augmented Reality devices). However, accessing functionality inherent to a single device or manufacturer using OpenXR is difficult.

Unity provides the XR Interaction Toolkit [48] (XRIT), a toolkit that provides a platform-agnostic framework for implementing interactions in VR, however, the framework does not provide as large of a selection of precreated interactables as its competitors. When combined OpenXR and XRIT provide a widely supported device-agnostic method of interacting with VR environments.

Late into the development of our original paper VRMADA released an experimental competitor to the OpenXR and XRIT stack they call UltimateXR [50]. UltimateXR itself acts as a middleware between most of the major VR device manufacturers and provides a fallback mechanism to use OpenXR if none of the platforms it has support for are available. This method was chosen since the OpenXR standard does not provide support for several common VR features, the most notable being haptic hand feedback. Additionally, UltimateXR provides not only hand presence utilities (and one of the most feature rich hand presence facilities available no less) but an IK based solution for estimating the position of a user's entire upper body. Finally, as a cherry on top, the framework also provides a utility which will blank the user's screen if their head happens to go inside of a solid object.

The combination of OpenXR and XRIT was chosen as

Table 1: VR framework comparison summary

Framework	Supported Platforms	Interactables Library	Hand Presence	Deprecated	Experimental
UltimateXR	HTC, Valve, Oculus Microsoft, and Varjo	Yes	Yes+	No	Yes
OpenXR + XRIT	HTC, Valve, Oculus Microsoft, and Varjo	Basic*	Basic†	No	No
Steam VR	HTC and Valve	Yes	Yes	Yes	No
Oculus SDK	Oculus	Yes	Yes	No	No

+ Includes support for not only hand presence, but pose estimation for the entire upper body.

* Includes support for basic grabbable objects that follow the user's hand.

† Includes support for representing the location of the user's hand, however it may not properly represent the type of controller the user is using.

the foundation for uMuVR originally. However, UltimateXR provides an enticing set of features but it is experimental and could be abandoned at any point by its developer. Thus we decided to provide support for both frameworks. Table 1 summarizes the differences between the discussed VR frameworks.

3 Benchmarks

While the choice of VR framework was fairly straightforward given our goals and SteamVR's recent deprecation, choosing a networking solution proved more difficult, many of the frameworks aim to be approximately equivalent with the primary distinguishing factor being their performance. There also exists a multitude of high throughput compression libraries that all serve the same role (namely making data smaller) with different throughput-to-compression ratios.

3.1 Networking Frameworks

We began by examining several different Unity networking frameworks. Several of these frameworks require that the code for clients and servers be written in different projects; all of these frameworks were rejected since providing a unified framework with a fragmented codebase produces extra complexity that we decided it would be best to avoid.

The first framework we considered was Photon's Fusion [14] framework. This was the framework that sparked the platform support requirement, since difficulties were encountered when using their provided DLLs. Additionally, Fusion's documentation is lacking, making development with the framework a frustrating experience. Alternatively, Fusion is one of the two considered frameworks (along with Novotny et al.) to provide a matchmaking system for players to discover each other without requiring an IP address. All of these factors paired with the existence of a price tag on these services led us to search for other options.

Fish-Networking [16], Mirror [49], and Unity's Netcode for GameObjects [45] (NCGO) were all considered next. All of these frameworks are open source and have similar designs, with minor differences in usability between the three, but nothing major enough to strongly influence a decision. Mirror supports a purely peer-to-peer-based architecture while Fish-Networking and NCGO support a client-server architecture where the server can either be dedicated or hosted on one of the clients. In a peer-to-peer architecture like Mirror, data is sent from every connected user to every other connected user, without the aid of a central authority. Alternatively, in a client-server architecture, users send their data to a central server which is then responsible for either rejecting it or forwarding it to the other users.

After identifying the several candidate frameworks, a performance benchmark was performed; the results of which along with several other comparison details are explained in Table 2. The performance benchmarks were conducted utilizing the methodology outlined in Fish-Networking's documentation [15] except: the tick rate for every framework was set to 60 ticks per second, the server was run from within Unity's editor, and thirty separate client executables were launched, all on a single machine[1]. All of the code utilized for these benchmarks can be found in uMuVR's Git repository [11] spread across several branches whose names all start with "benchmark/". Thirty clients were chosen since more would result in GPU throttling. Bandwidth information was captured using Wireshark's [8] Protocol Hierarchy statistics, filtered to only scan relevant ports that captured the number of bytes transferred which were then divided by the timestamp of the last packet scanned to find the average bandwidth. Since data was captured on a single machine, the bandwidth statistics represent both sent and received data. All data was captured and averaged over a period of five minutes.

Once the benchmarks had been performed, Fish-Networking proved to perform better than its competition, with similar frame rates and significantly reduced bandwidth overhead; and thus reduced bandwidth utilization indicating that more information can be exchanged before network infrastructure becomes overloaded. This leads to a direct increase in the

[1] The machine used to run the benchmarks is custom built with an Intel i7-12700k, EVGA GeForce RTX 3090 with 24GB of dedicated RAM, 32GB 2133MHz Corsair RAM, and a Samsung 980 Pro NVME SSD, running Unity 2021.3.5f1 set to build executables with the IL2CPP backend.

Table 2: Networking framework comparison over several performance metrics

Framework	Estimated Max Concurrent Users	Average FPS	Average Bandwidth	Cost	Supported Architectures	Matchmaking	Voice
Fish-Networking	500+	60.19	0.94 MB/s	Open Source	Hosted/ Dedicated	No	No
Mirror	200+[†]	60.23	2.15 MB/s[‡]	Open Source	P2P	No	No
Netcode for Game-Objects	Not Published	59.97	3.42 MB/s[‡]	Open Source	Hosted/ Dedicated	No	Yes[§]
Photon Fusion (Shared Topology)	2000	25.08	1.82 MB/s	Per User	Hosted/ Dedicated/ P2P	Yes	Yes[¶]

[†] Old stress test demos have shown Mirror supporting 480 concurrent users, however this has not been tested in practice.
[‡] Due to how Mirror and Netcode for GameObjects calculate their tick rate, these numbers are not based on exactly 60 ticks per second (we found it was between 55 and 60) whereas the other frameworks are.
[§] Provided by separate subscription priced Vivox package.
[¶] Provided by separate subscription priced Photon Voice package.

number of users that can be connected at once or the amount of network synchronized objects that can be in a scene while still utilizing the same amount of bandwidth. The benchmark utilized appears to be designed to fairly showcase Fish-Networking's performance superiority with a minimal amount of bias; that being said, we acknowledge that there is an unlikely potential of biasing in the results that we missed. With all factors considered, including several of Fish-Networking's nicer usability features, Fish-Networking became the clear choice to base uMuVR upon.

3.2 Compression Libraries

Data compression is a relatively hot field with many competing algorithms. It is worth noting that each considered library is a C# port of the original (usually C) algorithm. We began by considering the LZ4 [27] compression algorithm. This implementation along with the LZF [37] implementation provides an interface that allows the same buffer to be reused, reducing the burden on the C# Garbage Collector. LZF also has the advantage of being implemented as a single file and providing a tweakable compression ratio, we tested both the default high compression ratio, a faster compression ratio more suited to our needs, and a version of the implementation [22] tuned for Unity which we quickly discarded after we discovered its poor performance relative to the more general implementation.

Additionally Snappy [1], Zstandard [41], and Bontli [51] where all considered. Neither the Snappy nor Bontli implementations support buffer reuse and the Zstandard implementation requires build processors that prevent it from being used within Unity. Additionally, Bontli is a slower algorithm, thus we primarily included it as a reference for high-end compression ratios.

Since we are going to be compressing vocal audio from users' microphones, we performed this benchmark on a similar sample of audio. All of the code utilized for these benchmarks can be found in FishyVoice's Git repository [10] spread across several branches whose names all start with "benchmark/". Our voice networking library delivers segments of audio with a size of approximately 1600 bytes, thus we take similarly sized chunks of audio (55 chunks per second from our clip resulting in samples 1603 bytes in size). Since the sample clip is about 5 seconds long we collect 1375 looping samples representing five passes over the clip. For each sample, we determine how long it takes to serialize and then deserialize the packet (extra data is randomized using a fixed seed so that all algorithms are given the same data) then once these values are averaged we subtract the average time taken when no compression is used to find the extra time taken. We also record the size of both the compressed and uncompressed packet which we use to calculate the compression ratio. We discard the first sample taken from this analysis since there is a noticeable increase in time needed to initialize the serialization system, the time this extra initialization takes averaged over five startups along with the rest of the data mentioned above is presented in Table 3. All results were collected on the same machine used in the networking benchmarks, but Unity 2021.3.15f was utilized and the results were written to a CSV file by the benchmark itself.

LZF with an HLOG of 11 has the best trade off of performance to compression ratio. Slightly higher compression ratios can be achieved by the LZF algorithm at the cost of significantly more performance degradation. Thus the LZF implementation with this set of parameters was chosen as our audio compression library.

4 Networking Design and Implementation

4.1 Ownership

In most applications, when networking provides a latency spike, it is an annoying irritant that is usually ignored, however, in VR even a minor latency spike is substantially more noticeable. The increased immersion makes many people more sensitive to issues with the simulation, and a momentary lack of movement due to a latency spike increases Transport Delay

Table 3: Compression library comparison over several performance metrics

Library	Extra Compression Time	Extra Decompression Time	Average Compression Ratio	Average Initialization Time
Uncompressed	$0\mu s$	$0\mu s$	1	$13480.4\mu s$
LZ4	$29.0859\mu s$	$10.1048\mu s$	3.2446	$12488.2\mu s$
Snappy	$73.9032\mu s$	$30.5655\mu s$	3.7622	$5402.0\mu s$
LZF (hlog10)	$24.75\mu s$	$14.33\mu s$	3.7877	$3355.0\mu s$
LZF (hlog11)	$21.42\mu s$	$12.18\mu s$	3.8535	$3463.8\mu s$
LZF (hlog13)	$24.89\mu s$	$12.72\mu s$	3.8536	$2794.0\mu s$
LZF (hlog16)	$54.10\mu s$	$14.01\mu s$	3.8571	$2954.6\mu s$
Bontli	$469.51\mu s$	$102.81\mu s$	7.7192	$23269.0\mu s$

as described by Stoner et al. [42], which could easily invoke a bout of simulator sickness. To account for this discrepancy, all physics simulations and other interactions in uMuVR are performed client authoritatively (Meaning they are performed on the local client's machine, with the results relayed to other clients through the server. This model exists in contrast to a server authoritative model where all of the simulations are performed on the server and then propagated to the clients). This client authoritative structure poses an important question: For any given object, who should simulate it?

Fish-Networking, and every other considered networking framework, support the concept of object ownership. One particular user owns the object, and thus is responsible for simulating its behavior. However, these implementations typically only allow for ownership to be transferred between users upon object creation or some other manually invoked event. uMuVR elaborates upon this feature by allowing ownership to be assigned to certain volumes of space and automatically transferred upon interaction.

4.1.1 Ownership Management. Ownership management in uMuVR is orchestrated by a Unity component appropriately named `OwnershipManager`. Since ownership management is facilitated by a component, it is an opt-in feature, thus certain objects (notably the `UserAvatars` discussed in the next section) can simply belong to a single user without any possibility of transfer. The `OwnershipManager` has two main responsibilities: first, it is assumed that if someone is actively interacting with an object, they own it and are responsible for its simulation. Second, we facilitate a simplified version of Kawano and Yonekura's Allocated Topographical Zone [25] (AtoZ) algorithm that we call `OwnershipVolumes`.

4.1.2 Ownership Volumes. `OwnershipVolumes`, unlike AtoZ's regions which dynamically morph based on users' positions, are predefined fixed regions of space. This allows for more fine-grained control over exactly where ownership transfers will occur. `OwnershipVolumes` have several methods of deciding who owns them. The two primary methods, oldest user and newest user, rely on collisions to detect when users enter or exit the volume. Additional methods are provided where ownership is assigned to the objects creator and ownership is not managed by the component but instead managed manually, similar to how ownership is managed by default in Fish-Networking.

Early implementations of `OwnershipVolumes` were afflicted with an interesting bug where the small amount of jitter present when a physics simulation changes owners would cause an object to re-enter the volume it just departed, which often resulted in a jittery loop of repeated ownership transfers that could last for up to several seconds. We solved this issue by adding a short window of time lasting ten ticks, approximately 78 milliseconds, after an ownership transfer occurs during which another ownership transfer can not occur.

4.2 Clear Separation of Inputs, Visuals, and Networking

One of uMuVR's major priorities is laying a foundation for integrating mixed VR and non-VR users into the same shared environment. For this to be possible there needs to be a separation between the visuals displayed to users, which are then synchronized over the network, and the inputs driving those visuals. To facilitate this, we have developed a novel slot-based system where pose data (three-dimensional positions and rotations) can be stored in named slots. Originally, this Pose-Slot System featured 12 slots, head and pelvis, along with left and right shoulders, elbows, wrists, knees, and ankles that should be capable of representing the majority of human poses (without regard to finger positions). We then discovered that for some applications some of these points are unnecessary and several additional points would be useful, thus we generalized to a system where a variable number of named slots can be associated with pose data.

These pose-slots act as a unified layer of glue code as defined by Hummel and Atkinson [23]. Various input methods can store pose data and then a single visual representation can use either straightforward pose copying techniques or more complex techniques which are elaborated on in Section 5 to position the visuals (although there is nothing preventing a developer from creating their own additional techniques). In uMuVR's current design the visuals are responsible for synchronizing their state across the network, thus the Network Layer need not have any awareness of the Input Layer.

4.2.1 UserAvatar. The `UserAvatar` serves two major purposes. First, it tracks the object's current owner and if the local user is also the current owner it creates appropriate input controls for them. Whenever the ownership of an object changes, we reperform this check, always ensuring that only the current owner has input control. This system is designed to support multiple types of input depending on the medium of the user.

Second, it acts as the storage location for pose-slots. The Pose-Slot System is implemented using a dictionary mapping strings to referable pose data. Behind the scenes, links to this dictionary are converted to direct references to the associated pose data so that no runtime performance is lost using this system.

Arbitrary property storage can easily be added through inheritance with convenient access to an event function that is called after input is spawned, which is useful if there is any additional non-generalizable glue code that needs to be executed. Combined with the utilities provided by `SyncPoses`, the `UserAvatar` provides a powerful, generalized, and extendable input storage utility.

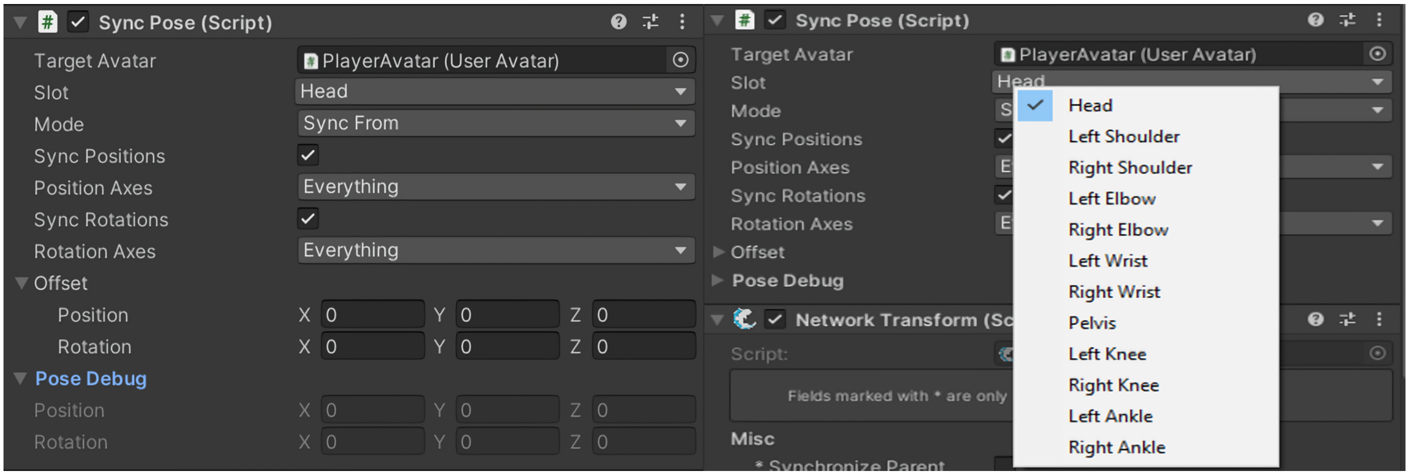
4.2.2 SyncPose. `SyncPose` is also a Unity component that is used to load or store data from or to the `UserAvatar`, possibly with an offset. `SyncPoses` attached to objects in the Input Layer read the location of objects with local input control and then store them in one of the `UserAvatar`'s pose-slots. Similarly, `SyncPoses` attached to objects in the Visual Layer load the location of objects from the `UserAvatar` and apply that location to objects in the visual representation. To facilitate this, `SyncPoses` take a reference to a prefab (Unity's word for a prepackaged and reusable set of entities and with components preattached) version of the `UserAvatar` they are synchronizing with and provide a custom graphical interface as shown in Figure 1 to make selecting pose-slots simple.

Additionally, `SyncPoses` support locking each axis of the position and each Euler axis of the rotation so that they are not copied. This provides the capability of synchronizing from the position of one object and then copying one of the rotational axis of another object into the same pose-slot.

4.2.3 Network Synchronization. After input data has been transferred to the Visual Layer, positional information must be propagated to other clients on the network. Fish-Networking provides a `NetworkTransform` component that synchronizes position, rotation, and scale across the network; however, for objects not based upon the `UserAvatar` model, Fish-Networking does not provide any client authoritative method of synchronizing physics properties. Thus, we implemented a `NetworkRigidbody` component that synchronizes the physics properties (velocity, angular velocity, gravity, and drag) utilized by Unity's physics simulation system. Due to the Client Authoritative nature of uMuVR, these properties are only necessary when an ownership transfer occurs; however, we discovered that the delay encountered when synchronizing these properties during such an event, paired with the nondeterministic nature of Unity's physics system would produce unpredictable changes in the object's trajectory after an ownership transfer.

Likewise, we discovered that utilizing an unreliable method of delivery produced similar unpredictable changes. Fish-Networking is built upon LiteNetLib [38], an unreliable UDP-based [34] C# transport that provides its own optional reliability layer with a design very similar to TCP [34] but without dedicated congestion control. Thus, we use LiteNetLib's reliability mode to synchronize velocity and angular velocity to all clients every tick while other less frequently changed properties are reliably synchronized whenever a change is detected.

4.2.4 Post Processing. We discovered through some of our testings with Neural Network driven inputs, that it might be useful to provide developers a method of processing poses stored in a `UserAvatar` after the fact. For some of our early tests, we made use of a `PostProcessingUserAvatar` which runs a First Order Exponential Averaging Low Pass Filter, the notation for which is detailed in Equation 1, over the Neural



Sync Pose (Script)

Target Avatar: PlayerAvatar (User Avatar)
Slot: Head
Mode: Sync From
Sync Positions: ✓
Position Axes: Everything
Sync Rotations: ✓
Rotation Axes: Everything
Offset
Position: X 0 Y 0 Z 0
Rotation: X 0 Y 0 Z 0
Pose Debug
Position: X 0 Y 0 Z 0
Rotation: X 0 Y 0 Z 0

Sync Pose (Script)

Target Avatar: PlayerAvatar (User Avatar)
Slot: Head
Mode: S
Sync Positions: ✓
Position Axes: E
Sync Rotations: ✓
Rotation Axes: E
Offset
Pose Debug

Network Transform (Sc
Script:
Fields marked with * are only
Misc
* Synchronize Parent

Figure 1: The interface provided for `SyncPoses` (left) and customized dropdown that makes selecting pose-slots simple (right)

Network generated poses to smooth some jittery behavior.

$$y(n) = a * x(n) + (1 - a) * y(n - 1) \quad (1)$$

To facilitate this we provide a separate `PostProcessingUserAvatar` type that provides a separate set of pose slots which represent the unprocessed pose data; the names of which we synchronize with the developer-defined names. `SyncPoses` are aware of this separation and will properly store and load information to and from the proper slots. Every frame we iterate over every slot, and for each slot we call a developer definable processing function that performs whatever arbitrary calculation they would like on the old pose data given the new pose data. For cases where there is no interdependence on the states of other poses, we provide an option to run these iterations using Unity's C# Job System [46]. This allows the processing to occur on background threads.

The `UserAvatar`-based design at uMuVR's core has many moving parts, however, most of them are not unique. Figure 2, provides a visual summary of how these components interact. The figure's color-coded lines clearly illustrate how each layer morphs its input data into a form the next layer can understand without needing to have any awareness of the original form. It is worth noting that the Pose-Slot System is a lossy approximation, thus certain rare and extreme poses are not representable using this system.

4.3 Voice

The final aspect of uMuVR's networking design worth mentioning is its voice communication implementation. We wanted a voice implementation that was simple and did not rely on any costly third-party subscription services. UniVoice [2] met all of these requirements, however, its non-Unity idiomaticity and entirely separate networking stack were less than desirable.

Before we can discuss the adjustments we made to UniVoice, a basic understanding of its architecture is required. UniVoice is based on four main classes: an `IChatroomNetwork` that is responsible for transporting voice data, an `IAudioSource` that is responsible for acquiring voice data, an `IAudioOutputFactory` that is responsible for creating an audio output for every relevant peer, and finally a `ChatroomAgent` that manages the previous three. The `ChatroomAgent` supports separating users into rooms, limiting the pool of other relevant users to only the users within the same room.

The first change we made was implementing a Fish-Networking-based `IChatroomNetwork` we call `VoiceNetwork`. It utilizes Fish-Networking's remote

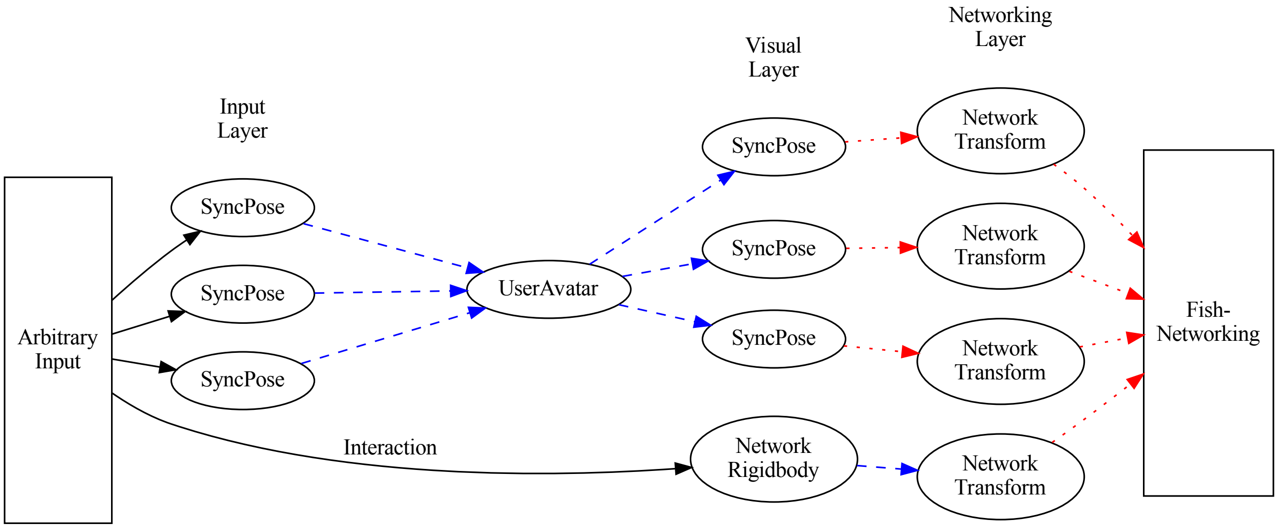


Figure 2: A visual overview of the components used to link inputs, visuals, and networking. Inputs (depicted in solid black) are applied to various objects in the environment. `SyncPoses` then transfer modified position information (depicted in dashed blue) from the Input Layer to the Visual Layer via the `UserAvatar`. Finally, global position data (depicted in dotted red) is sent through the network and used to set the position of this user as seen by other users. Simultaneously, interactions adjust properties of the physics simulation which are propagated to the rest of the network via a `NetworkRigidbody` and `NetworkTransform`

procedure calls to transfer data between users, invoking subscribable events at either end so that other interested objects can listen to the data. Furthermore, a dictionary is kept in sync between all of the connected users that maps room names to lists of connections currently within those rooms. Behind the scenes, UniVoice ensures that audio is only sent to users within the same room as the sender.

In terms of improving Unity idiomaticity, we implemented a component that allows easy selection of audio input sources from within the Unity editor. Additionally, the `VoiceNetwork` is implemented as a component that can be easily referenced using Unity's graphical editor and provides several extra convenience functions that set up an agent which utilizes the `VoiceNetwork`; all referenceable using Unity's editor.

Additionally, we added two new features on top of the UniVoice stack: automatic positional audio and disableable voice compression. The audio packets which we transfer through Fish-Networking include an additional variable that encodes the position of the speaker. A `PlayerAudioPositionReference` component is used to determine where the user should be positioned from the perspective of other users. Finally, a `PositionalAudioOutput` factory is provided which creates a specialized audio source for each user. The default `VoiceNetwork` can detect the presence or absence of positional audio components and will automatically adapt as needed.

We compress audio using the LZF [37] compression library. A discussion of why this library was chosen can be found in Section 3.2. Our audio packets have a custom Fish-Networking serializer and deserializer pair which convert the packet into a byte array which is then passed through LZF. A C# define is provided so that from the Unity script control page you can easily disable or enable compression.

5 Body Presence Design and Implementation

uMuVR's current body presence facilities can be separated into two categories: the half which lives in the Input Layer and the half which lives in the Visual Layer. Everything, including the physics simulation, has been designed in such a fashion that it can simply be disabled on remote clients who will rely on data coming from the network to position remote avatars in their scenes.

5.1 Input Layer

The input layer half has a very simple goal, namely, calculate the position and rotation of all 12 original pose-slots of the user avatar. Actually, at present the Body Presence system is using 42 pose-slots (the 12 original and a slot for every finger joint on both hands). We hope to simplify this in the future but that is a story for another paper.

5.1.1 UltimateXR. Everything from the hips up is simple from uMuVR's perspective. UltimateXR [50] provides Cyclic Coordinate Descent IK [26] based utilities for estimating elbow, spine, hip, and neck positions based on the position of the user's controllers and head. Unfortunately, we do not currently have a solution for accurately predicting shoulder positions, thus if shrugs are desired they will need to be externally tracked.

Additionally, UltimateXR provides a system for defining grab points and associated poses on interactable objects. This system is used to define the position of every finger joint. We can thus simply extract the relevant points from their implementation and pass the hip position off to a Phase-Functioned Neural Network controller.

5.1.2 PFNN. Phase-Functioned Neural Networks [21] (PFNN) provide a system for cheaply and procedurally generating the next pose of a walking cycle based on a constantly progressing phase variable. This system is capable of creating animations for convincingly traversing complex environments; give or take the fact that sometimes the feet do not find themselves correctly planted on the ground, a flaw which can be corrected with a secondary foot placement pass which we shall discuss in Section 5.1.3.

The PFNN network takes as input a target direction, the relative speed it should approach its target, and the pose last frame. It then generates a new pose. Unfortunately, this process is very sensitive to deviations from the information it was trained on. For instance, the publicly available weights utilized by Holden et al. [21] assume that poses will be requested at a constant rate of 60 times per second. Thus we have a controller sitting over the model which only requests a new pose if a 60th of a second has elapsed since the last pose request.

Additionally, to keep the legs under the body, our controller will define directions and velocities in such a way that the legs will be positioned under the hips when everything is said and done. However, care needs to be taken with the network's target position, once it has "reached" its target position it will stop rotating to match the user, but if it never "reaches" its target the simulation will destabilize. Thus our controller goes through a rather complicated set of procedures to dynamically adjust the threshold at which the network is considered to "reach" its target based on the hip's velocity and the difference in angle between the hips and feet.

This whole process also fails if the experience being developed uses a teleportation-based locomotion system. The legs will be left behind as the upper body teleports and then wander over to reattach themselves. While we are sure certain experiences could use this as a hilarious gimmick, for the average case this behavior is undesirable. Thus the controller also detects when the upper and lower body's positions have diverged too much and will reset the leg simulation to match the upper body in this case.

Much like the shoulders, this simulation can only predict "normal" movements such as standing, walking, and crouching. It will encounter issues with more exotic motions such as laying down or performing a back flip. External trackers will still be needed if these sorts of motions will be commonly necessary for the experience.

5.1.3 Slope Aware Foot Placement. Once PFNN has generated a leg and foot placement for us, we then need to ensure that the feet rest squarely on the ground. We begin by projecting the toes straight down onto the ground. Foot placement is then based on the height of the ankle above the toes which we consider a configurable constant, named Δ. We then fall into two cases based on the height of the ankle above the ground minus Δ, which we call H. If H is greater than Δ the ground is steep and we don't modify the foot placement, if H is less than or equal to Δ we move the ankle straight down to a point Δ above the ground. These two cases are represented graphically in Figure 3. This procedure mimics human behavior where we stand on our toes on steep inclines and our heels on more gradual inclines.

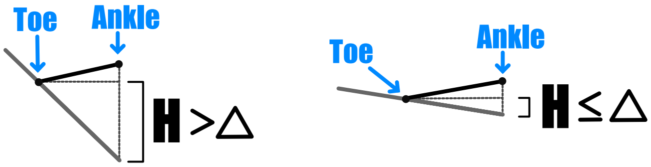


Figure 3: A representation of the Ankle and Toe points, and how they would relate to the ground in the Steep (left) and Not Steep (right) cases

Behind the scenes, all of the points from UltimateXR and PFNN are combined into a hidden `UserAvatar`. The foot placement algorithm depends on this abstraction in uMuVR's design to simplify the foot placement problem to just sliding points. We calculate how much we had to move the ankle down, and gradually spread decreases in height through the rest of the points to compensate. When combined with the IK system we run in the visual layer, this results in very convincing movements.

Additionally, the phase variable from PFNN is used as a weight factor to describe how strongly the foot placement should be applied. In parts of the phase that correspond to a raised foot in the walk cycle, we completely ignore the foot placement, while in phases where the foot should be planted, we apply it with a weight of 100% and blend between the two as necessary.

5.2 Visual Layer

The half of the system which lives in the Visual Layer is inspired from two primary sources: PhysIK [7] and HPTK [18]. PhysIK is a system for expressing IK like animations using a physics simulation. This allows for the "animations" to interact with themselves, for instance an arm can not be pushed through its body. While HPTK is a Unity library providing physics based hand interaction, instead of needing to press a button on your controller to pick up a box, it allows the user to simply use their hands to pick up the box.

HPTK's influence is simpler than that of PhysIK. They provide a proxy hand which represents the input data received from the user's controllers. Similarly we present a proxy hand to the user in the Input Layer, thus it only exists for them.

While a physics based system seems to work wonderfully for the upper body, we encountered numerous issues integrating such a system into the avatar's feet and legs. Thus the lower body is animated using the more traditional FABRIK [4] IK algorithm which is used to determine the rotations necessary for the feet, knees, and hips to all be properly positioned.

We originally were using FABRIK based IK for the arms and spine, as well as the legs. But perusing alternative implementations during development showed some of the interesting benefits of using a physics simulation; for instance, a hand properly interacting with the opposite out-stretched arm. A behavior which a properly tuned physics simulation simply provides that would be a tremendous amount of effort to emulate using traditional IK techniques. Thus we began implementing a version of PhysIK that would work in Unity. Due to the nature of Unity's physics system, this means that almost everything from PhysIK, except the ideas, was abandoned.

More specifically two main physics "constraints" have been adapted. A constraint which pulls a joint towards a location and a constraint which rotates a joint to match a rotation which, when both visualized, leave the model looking a lot like a puppet on strings as depicted in Figure 4.

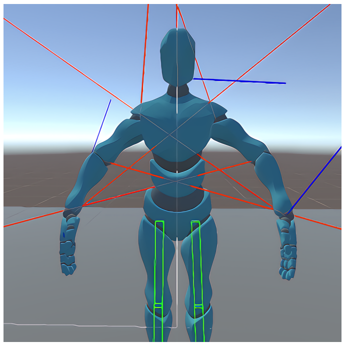


Figure 4: A visualization of the forces (red) and torques (blue) acting on an example avatar

A puppet on a string is an excellent explanation of how the location constraint works. It simply applies a force towards the targeted point which grows stronger the further away from the target the joint is; almost as if it is pulled by an invisible string. Figure 5 lists the code, executed every physics tick, to implement this functionality.

Unfortunately, there is not a nice analogy for how the rotation-matching constraint functions. It simply calculates the quaternion needed to convert the current world space rotation into the target world space rotation and then converts that quaternion to the angular velocity form that Unity expects when applying torques. There is a slight wrinkle; when applied to the center of mass, this rotation looks odd, so instead, the center of

```
// Every physics tick...
protected void FixedUpdate() {
    // We are applying a force from our
    //  current position towards the
    //  target
    var force = (target.position
        - transform.position)
        * springConstant;
    // Cap the force
    force = force.normalized
        * Mathf.Min(
            force.magnitude,
            maxForce
        );
    // Apply the force
    rigidbody.AddForce(force);
}
```

Figure 5: The code for the constraint which pulls a joint towards a target

mass has to be moved to the origin of the joint's local coordinate space since Unity always applies torques relative to the center of mass. The code for this constraint is listed in Figure 6.

6 Applications

Once we developed our basic implementation, we needed to test it in an actual application. We began by implementing a Ping-Pong-like game with only a subset of the full game's rules. We then integrated our framework into an existing project [32], performed as a joint partnership with the University of Nevada, Reno's Mining & Metallurgical Engineering Department, that needed multiuser functionality.

6.1 Ping-Pong

The main goal of the Ping-Pong application was to stress test the physics and ownership transfer aspects of uMuVR. The framework allowed this application to be implemented almost entirely without additional code; only two simple scripts: one to manage scoring and balls and another to position the scoreboard; combined only accounting for approximately 100 lines of code, where required.

Ping-Pong provides the quintessential application of `OwnershipVolumes`: a volume is positioned on either side of the net and each player owns half of the table. Figure 7 depicts this configuration. Additionally, the fast-paced nature of Ping-Pong illuminated many of the issues inherent in the behavior of the physics simulation surrounding an ownership transfer, providing us a test-bed for experimenting to reduce the jitter.

```
// Every physics tick...
protected void FixedUpdate() {
    // Reset the center of mass so that
    //  torque is properly applied
    rb.centerOfMass = Vector3.zero;
    rb.inertiaTensor = Vector3.one;

    // Calculate how much we need to
    //  rotate to match the object
    var diff = offset
        * target.rotation.Diff(
            transform.rotation
        );
    diff.ToAngleAxis(
        out var angle,
        out var axis
    );
    // Apply a torque to make this change
    //  occur
    rigidbody.maxAngularVelocity
        = springConstant;
    rigidbody.AddTorque(
        axis * (angle * Mathf.Deg2Rad)
            * springConstant,
        ForceMode.VelocityChange
    );
}
```

Figure 6: The code for the constraint which rotates a joint to match another joint's rotation. The center of mass needs to be reset so that torque is applied to the object's origin and not its center of mass

6.2 Mining Application

The addition of multiuser functionality allowed us to collaborate with Mining researchers on a training simulation [32]. This simulation is designed to teach mining truck drivers how to utilize a new proximity warning system to avoid collisions with equipment and personnel they can not see. This integration greatly challenged our original pose-slot implementation.

Instead of only needing to translate input for a human body, we also had to synchronize several properties associated with a mining dump truck whose Visual and Input Layers are depicted in Figure 8. This caused us to realize the inadequacies of our original fixed pose-slot mapping and led to its replacement with the current dynamic system.

In addition to extra poses that now needed to be tracked, we also had a car controller that needed direct access to wheel meshes for both physics and animation purposes. Similarly, several audio and haptic feedback utilities needed to be synchronized in both the Input and Visual Layers. Instead

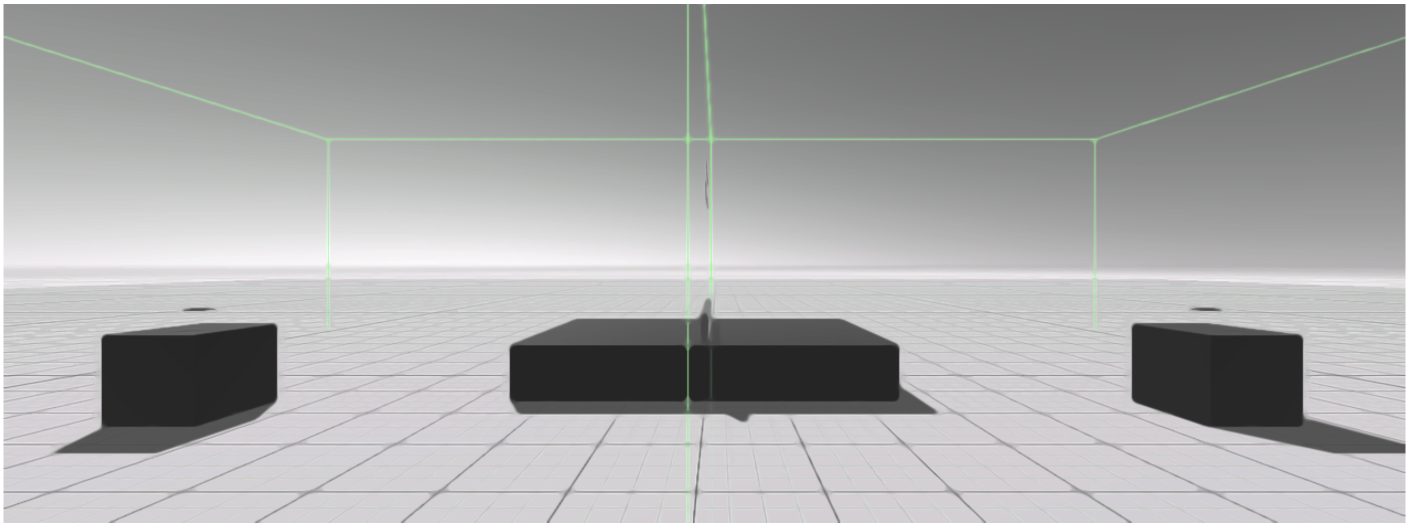


Figure 7: The Ping-Pong table with its two `OwnershipVolumes` outlined in green

of rewriting all of this functionality ourselves, we used the `UserAvatar`'s flexible extension ability to link these properties for us. Figure 9 lists all of the code necessary to facilitate this additional linkage.

7 Future Work and Conclusion

Looking to the future several aspects of uMuVR could be improved. The most notable one being how the physics simulation handles ownership transfers. While we have been able to greatly reduce the jitter this entails, we have not been able to eliminate it. Perhaps a Predictive Behavioral Model [17] could be used to ease this issue.

Furthermore, while a foundation has been laid for work on mixed VR and non-VR interactions, we have not even scratched the surface of the work that must be done in this field. Factors such as compelling interactions in both VR and non-VR still need to be developed.

Currently, the physics simulations in the body presence system takes a lot of manual tweaking to achieve acceptable results. Ways to automate or at least partially automate this process would be of great utility. Additionally, interactions between the simulated hands and other objects in the scene are inconsistent. Research has been done on this problem [13] and thus implementing at least part of these solutions would be beneficial. On the topic of hands, a lot of slots within the current `UserAvatars` are being devoted to storing the poses of every finger joint. It seems likely that it should be possible to simplify this representation to simply the knuckle's rotation and a single value representing how open or closed the rest of the finger is.

The publicized weights for PFNN lead to an avatar that tends to shuffle its feet, which can be undesirable when the user is standing still. Training weights optimized for the movements users tend to make in VR or maybe upgrading to newer work done on the same problem could prove beneficial. In a similar vein, the ad hoc tensor implementation that we are

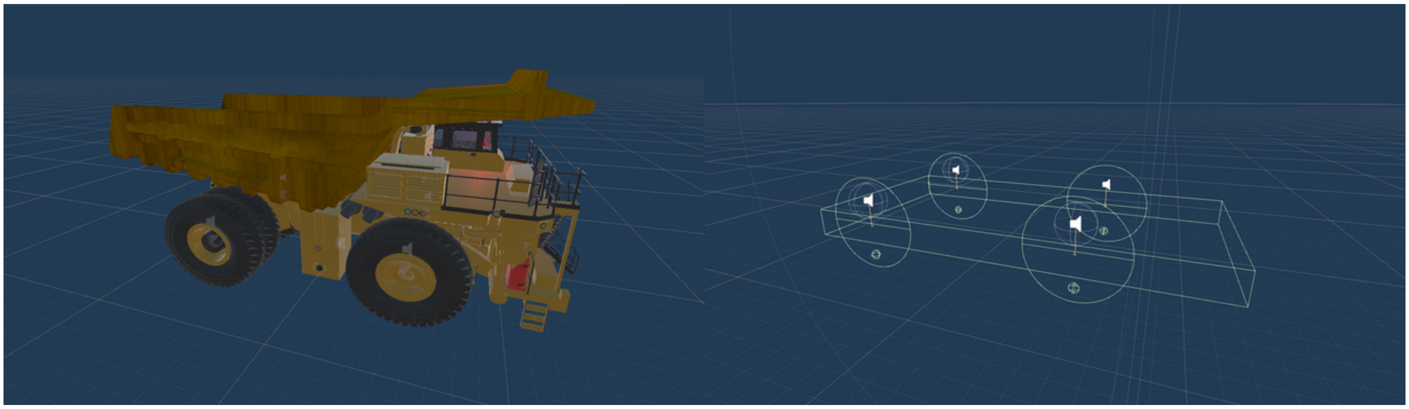


Figure 8: The mining dump truck's visual layer (left) and input layer (right)

```
using UnityEngine;
using UnityStandardAssets.Vehicles.Car;

public class TruckControlLinker
  : InputControlLinker {
    public CarController car;

    // Haptic feedback
    public Telemetry telemetry;
}

public class TruckAvatar : UserAvatar {
    public GameObject[] wheelMeshes;
    public Alarm alarm;
    public NetworkCarAudio carAudio;

    protected override void
      OnInputSpawned(GameObject input) {
        var linker =
          GetComponentInChildren
            <TruckControlLinker>();
        linker.car.m_WheelMeshes =
          wheelMeshes;
        linker.car.Start();

        carAudio.carController =
          linker.car;
        alarm.telemetry =
          linker.telemetry;
    }
}
```

Figure 9: The code used to link additional properties of the Mining Dump Truck. Every public attribute of these classes is set using Unity's visual editor. Note that this example is based upon a slightly older version of uMuVR which required a separate `InputControlLinker` component. This requirement has since been removed

currently using for PFNN is not hardware accelerated. If we wish to use additional neural networks for tasks such as voice transcription [39], unifying our tensor library in a manner that supports hardware acceleration would be desirable.

Finally, none of this implementation matters if either users or developers do not find the framework appealing. Thus several user studies examining various aspects of the framework's implementations and development usability will need to be conducted.

In conclusion, Novotny et al. created a useful framework for multiuser VR experiences. We have followed their example and expanded upon the foundation they laid. uMuVR has been designed to be simple and extendable. These claims have been tested and proved by using the framework for the development of several applications, one having requirements the framework was not originally designed to support. It additionally provides a unified framework for providing rich body presence facilities for users to experience. We hope this framework will prove to be a great boon to other developers and the VR field as a whole.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under grant numbers OIA-2019609, and OIA-2148788 Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] aloneguid (Ivan G). "GitHub - aloneguid/IronSnappy: .NET Managed Port of Google Snappy". `https://github.com/aloneguid/IronSnappy`, Last Accessed (1/11/2023).

[2] Vatsal Ambastha. "UniVoice: Voice chat/VoIP Solution for Unity. P2P Implementation Included". `https://github.com/adrenak/univoice`, Last Accessed (1/11/23).

[3] Kurt Andersen, Simone José Gaab, Javad Sattarvand, and Frederick C Harris. "METS VR: Mining Evacuation Training Simulator in Virtual Reality for Underground Mines". In Shahram Latifi, editor, "17th International Conference on Information Technology–New Generations (ITNG 2020)", Springer International Publishing. pp. 325-332, 2020. `doi:10.1007/978-3-030-43020-7_43`.

[4] Andreas Aristidou and Joan Lasenby. "FABRIK: A Fast, Iterative Solver for the Inverse Kinematics Problem". *Graphical Models*. 73(5):243-260, 2011. doi:10.1016/j.gmod.2011.05.003.

[5] Andreas Aristidou, Joan Lasenby, Yiorgos Chrysanthou, and Ariel Shamir. "Inverse Kinematics Techniques in Computer Graphics: A Survey". *Computer graphics forum*. 37(6):35-58, September 2018. doi:10.1111/cgf.13310.

[6] Polona Caserman, Philipp Achenbach, and Stefan Göbel. "Analysis of Inverse Kinematics Solutions for Full-Body Reconstruction in Virtual Reality". In "2019 IEEE 7th International Conference on Serious Games and Applications for Health (SeGAH)", pp. 1-8, 2019. doi:10.1109/SeGAH.2019.8882429.

[7] Frederick Choi. "PhysIK: Physics Based Inverse Kinematics for Character Posing and Animation". [online]. `https://www.cs.rpi.edu/~cutler/`

classes/advancedgraphics/S19/final_projects/fred.pdf, Last Accessed (1/11/2023), 2019.

[8] Gerald Combs. "Wireshark · Go Deep." https://www.wireshark.org/, Last Accessed (1/11/2023).

[9] John Coumerilh. "Standable: Full Body Estimation". https://www.standablevr.com/projects-8, Last Accessed (1/11/2023).

[10] Joshua Dahl. "Compression Benchmarks at Benchmark/Base". [online]. https://github.com/hpcvis/FishyVoice/tree/benchmark, Last Accessed (1/11/2023).

[11] Joshua Dahl. "Networking Benchmarks at Benchmark/Base". [online]. https://github.com/hpcvis/MuVR/tree/benchmark/base, Last Accessed (1/11/2023).

[12] Joshua Dahl, Erik Marsh, Christopher Lewis, and Frederick C. Harris. "GitHub: uMuVR-A Multiuser Virtual Reality and Body Presence Framework for Unity". [online]. https://github.com/hpcvis/MuVR/tree/uMuVR-AMultiuserVirtualRealityFrameworkforUnity Last Accessed (1/11/2023).

[13] Thibauld Delrieu, Vincent Weistroffer, and J. P. Gazeau. "Precise and realistic grasping and manipulation in Virtual Reality without force feedback". In "2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)", pp. 266-274, 2020. doi:10.1109/VR46266.2020.00046.

[14] Exit Games Inc. "Setting the Benchmark for Multiplayer Games. — Photon Engine". [online]. https://www.photonengine.com/en-US/Fusion, Last Accessed (1/11/2023).

[15] FirstGearGames. "Benchmark Setup". https://fish-networking.gitbook.io/docs/manual/general/performance/benchmark-setup, Last Accessed (1/11/23).

[16] FirstGearGames. "Introduction - Fish-Net: Networking Evolved". https://fish-networking.gitbook.io/docs/, Last Accessed (1/11/23).

[17] Chen Gao, Haifeng Shen, and M. Ali Babar. "Concealing Jitter in Multi-Player Online Games Through Predictive Behaviour Modeling". In "2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)", pp. 62-67, 2016. doi:10.1109/CSCWD.2016.7565964.

[18] Jorge Juan González. "FinalIK - HPTK". [online]. https://jorge-jgnz94.gitbook.io/hptk/integrations/finalik-1, Last Accessed (1/11/2023).

[19] Toni Härkönen. "Advantages and Implementation of Entity-Component-Systems". [online]. Bachelor of Science Thesis https://urn.fi/URN:NBN:fi:tty-201905231735, https://trepo.tuni.fi//handle/123456789/27593, Last Accessed (1/11/2023). April 2019.

[20] Fernanda Herrera, Soo Youn Oh, and Jeremy N. Bailenson. "Effect of Behavioral Realism on Social Interactions Inside Collaborative Virtual Environments". *Presence*. 27(2):163-182, 2020. doi:10.1162/pres_a_00324.

[21] Daniel Holden, Taku Komura, and Jun Saito. "Phase-Functioned Neural Networks for Character Control". *ACM Trans. Graph.* 36(4) article 42, July 2017, doi:10.1145/3072959.3073663. 2017.

[22] HouraiTeahouse. "Houraiteahouse/LZF: Simple C# LZF Compression Library Which Attempts to Minimize Memory Allocations". [online]. https://github.com/HouraiTeahouse/LZF, Last Accessed (1/11/2023).

[23] Oliver Hummel and Colin Atkinson. "The Managed Adapter Pattern: Facilitating Glue Code Generation for Component Reuse". In Stephen H. Edwards and Gregory Kulczycki, editors, "Formal Foundations of Reuse and Domain Engineering", Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 211-224, 2009.

[24] Dorota Kamińska, Tomasz Sapiński, Sławomir Wiak, Toomas Tikk, Rain Eric Haamer, Egils Avots, Ahmed Helmi, Cagri Ozcinar, and Gholamreza Anbarjafari. "Virtual reality and its Applications in Education: Survey". *Information*. 10(10):318, 2019. doi:10.3390/info10100318.

[25] Yoshihiro Kawano and Tatsuhiro Yonekura. "On a Serverless Networked Virtual Ball Game for Multi-Player". In "2005 International Conference on Cyberworlds (CW'05)", pp. 270-278, 2005. doi:10.1109/CW.2005.68.

[26] Ben Kenwright. "Inverse Kinematics – Cyclic Coordinate Descent (CCD)". *Journal of Graphics Tools*. 16(4):177-217, 2012. doi:10.1080/2165347X.2013.823362.

[27] Milosz Krajewski. "GitHub - MiloszKrajewski/K4os.Compression.LZ4: LZ4/LH4HC Compression for .NET Standard 1.6/2.0 (Formerly Known as LZ4NET)". [online]. https://github.com/MiloszKrajewski/K4os.Compression.LZ4, Last Accessed (1/11/2023).

[28] Catherine Oh Kruzic, David Kruzic, Fernanda Herrera, and Bailenson Jeremy. "Facial Expressions Contribute More than Body Movements to Conversational Outcomes in Avatar-Mediated Virtual Environments". *Scientific Reports (Nature Publisher Group)*. 10(1):1-23, 2020. doi:10.1038/s41598-020-76672-4.

[29] Marc Erich Latoschik, Daniel Roth, Dominik Gall, Jascha Achenbach, Thomas Waltemate, and Mario Botsch. "The Effect of Avatar Realism in Immersive Social Virtual Realities". In "Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology", Association for Computing Machinery, New York, NY, USA, VRST '17. doi:10.1145/3139131.3139156. 2017.

[30] John P Lewis, Matt Cordner, and Nickson Fong. "Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation". In "Proceedings of the 27th annual conference on Computer graphics and interactive techniques", pp. 165-172, 2000. doi:10.1145/344779.344862.

[31] Jean-Luc Lugrin, Maximilian Ertl, Philipp Krop, Richard Klüpfel, Sebastian Stierstorfer, Bianka Weisz, Maximilian Rück, Johann Schmitt, Nina Schmidt, and Marc Erich Latoschik. "Any "Body" There? Avatar Visibility Effects in a Virtual Reality Game". In "2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)", pp. 17-24, 2018. doi:10.1109/VR.2018.8446229.

[32] Erik Marsh, Joshua Dahl, Alireza Kamran Pishhesari, Javad Sattarvand, and Frederick C. Harris. "A Virtual Reality Mining Training Simulator for Proximity Detection". In "20th International Conference on Information Technology: New Generations (ITNG 2023)", Springer International Publishing, Advances in Intelligent Systems and Computing. To Appear. 2023.

[33] Kiran Nasim and Young J. Kim. "Physics-Based Interactive Virtual Grasping". In "Proceedings of HCI Korea", Hanbit Media, Inc., Seoul, KOR, HCIK '16. pp. 114-120, 2016. doi:10.17210/hcik.2016.01.114.

[34] Network Working Group, Internet Engineering Task Force. "RFC1122: Requirements for Internet Hosts-Communication Layers". [online]. https://www.rfc-editor.org/rfc/rfc1122, Last Accessed (1/11/2023). 1989.

[35] Alexander Novotny, Rowan Gudmundsson, and Frederick C. Harris. "A Unity Framework for Multi-Player VR Applications". *International Journal of Computers and Their Applications*. 27(3)115-121, September 2020.

[36] Open Handset Alliance. "Android Secure & Reliable Mobile Operating System". https://www.android.com/, Last Accessed (1/11/23).

[37] Chase Pettit. "GitHub - Chaser324/LZF: Simple C# LZF Compression Library which Attempts to Minimize Memory Allocations". [online]. https://github.com/Chaser324/LZF, Last Accessed (1/11/2023).

[38] Ruslan Pyrch. "GitHub - RevenantX/LiteNetLib: Lite Reliable UDP library for Mono and .NET". [online]. https://github.com/RevenantX/LiteNetLib, Last Accessed (1/11/2023).

[39] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. "Robust Speech Recognition via Large-Scale Weak Supervision". doi:10.48550/ARXIV.2212.04356. 2022.

[40] Anastasia Rychkova, Alexey Korotkikh, Andrey Mironov, Artem Smolin, Nadezhda Maksimenko, and Mikhail Kurushkin. "Orbital Battleship: A Multiplayer Guessing Game in Immersive Virtual Reality". *Journal of Chemical Education*. 97(11):4184-4188, 2020. doi:10.1021/acs.jchemed.0c00866.

[41] Oleg Stepanischev. "GitHub - oleg-st/ZstdSharp: Port of ZSTD Compression Library to C#". [online]. https://github.com/oleg-st/ZstdSharp, Last Accessed (1/11/2023).

[42] Heather A Stoner, Donald L Fisher, and Michael Mollenhauer. "Simulator and Scenario Factors Influencing Simulator Sickness". In Donald L. Fisher, Matthew Rizzo, Jeffrey Caird, and John D. Lee, editors, "Handbook of Driving Simulation for Engineering, Medicine, and Psychology", CRC Press/Taylor & Francis, Boca Raton FL, pp. 220–243. 2011.

[43] Stress Level Zero. "Stress Level Zero". [online]. https://www.stresslevelzero.com/, Last Accessed (1/11/2023).

[44] The Khronos Group Inc. "OpenXR Overview - The Khronos Group Inc". [online]. https://www.khronos.org/openxr/, Last Accessed (1/11/2023).

[45] Unity Technologies. "About Netcode for GameObjects". https://docs-multiplayer.unity3d.com/netcode/current/about, Last Accessed (1/11/2023).

[46] Unity Technologies. "C# Job System". https://docs.unity3d.com/Manual/JobSystem.html, Last Accessed (1/11/2023).

[47] Unity Technologies. "Unity Real-Time Development Platform — 3D, 2D VR &; AR Engine". [online]. https://unity.com/, Last Accessed (1/11/2023).

[48] Unity Technologies. "XR Interaction Toolkit: 2.0.4". [online]. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/index.html, Last Accessed (1/11/2023).

[49] vis2k. "Mirror Networking, Documentation". [online]. https://mirror-networking.gitbook.io/docs/, Last Accessed (1/11/2023).

[50] VRMADA. "UltimateXR: The XR Framework and Toolkit". [online]. https://www.ultimatexr.io/, Last Accessed (1/11/2023).

[51] XieJJ99, James Hopper, and AzureGem. "GitHub - XieJJ99/brotli.net: The .Net Implementation for the Brotli Algorithm". [online]. `https://github.com/XieJJ99/brotli.net`, Last Accessed (1/11/2023).

Joshua Dahl is currently a student at the University of Nevada, Reno. He is pursuing a BS with a major in Computer Science and Engineering and a minor in Mathematics. When he graduates he is planning on pursuing a Ph.D. in Computer Science where he hopes to continue to make contributions to both the fields of computer graphics and programming languages.

Erik Marsh is currently pursuing a MS in Computer Science at the University of Nevada, Reno. He received a BS in Computer Science and Engineering with a minor in Mathematics from the University of Nevada, Reno in 2022. His research interests include input devices, real-time simulations, and data visualization.

Christopher Lewis received his BS and MS degrees in Computer Science and Engineering from the University of Nevada, Reno in 2020 and 2022 respectively. During this time he specialized in Virtual Reality as a researcher in the High Performance Computation and Visualization Lab. Since graduating, he has went on to work as an engineer for Moth + Flame, a company making high quality VR training experiences in both hard and soft skills. He has plans to return to academia for a PhD in a few years.

Frederick C. Harris, Jr. received his BS and MS degrees in Mathematics and Educational Administration from Bob Jones University, Greenville, SC, USA in 1986 and 1988 respectively. He then went on and received his MS and Ph.D. degrees in Computer Science from Clemson University, Clemson, SC, USA in 1991 and 1994 respectively.

He is currently the Associate Dean for Research in the College of Engineering, a Foundation Professor in the Department of Computer Science and Engineering, and the Director of the High Performance Computation and Visualization Lab at the University of Nevada, Reno. Since joining UNR, he has worked on research projects funded by federal agencies (NSF, NASA, DARPA, ONR, DoD) as well as industry. He is also the Nevada State EPSCoR Director and the Project Director for Nevada NSF EPSCoR. He has published more than 300 peer-reviewed journal and conference papers along with several book chapters and has edited or co-edited 14 books. He has had 14 PhD students and 81 MS Thesis students finish under his supervision. His research interests are in parallel computation, simulation, computer graphics, and virtual reality. He is also a Senior Member of the ACM, and a Senior Member of the International Society for Computers and their Applications (ISCA).

On Generalization of Residue Class Based Pyramid Tree P2P Network Architecture

Indranil Roy*
Southeast Missouri State University, Cape Girardeau, MO

Nick Rahimi†,
2 University of Southern Mississippi, Hattiesburg, MS

Ziping Liu*
Southeast Missouri State University, Cape Girardeau, MO

Bidyut Gupta‡
Southern Illinois University; Carbondale, IL

Narayan Debnath§
Eastern International University; VIETNAM

Abstract

In this paper, we have considered a recently reported 2-layer non-DHT-based structured P2P network. It is an interest-based system and consists of different clusters such that peers in a given cluster possess instances of a particular resource type. It offers efficient data look-up protocols with low latency. However, the architecture lacks in one very important aspect: it is assumed that no peer in any cluster can have more than one resource type and this could be a very hard restriction practically. Therefore, in the present work, we have addressed this issue of generalizing the architecture to overcome this restriction and have come up with effective solutions. We have modified appropriately our previously reported data look-up protocols wherever applicable in order to accommodate the idea of generalization while making sure that look-up latencies of these modified protocols remain the same.

Key Words: Structured P2P network; residue class, interest-based; non-DHT; complete and incomplete pyramid trees; virtual neighbors.

1 Introduction

Peer-to-Peer (P2P) overlay networks are widely used in distributed systems due to their ability to provide computational and data resource sharing capability in a scalable, self-organizing, distributed manner. There are two classes of P2P networks: unstructured and structured ones. In unstructured systems [3] peers are organized into arbitrary topology. It takes help of flooding for data look up. Problem arising due to frequent peer joining and leaving the system, also known as churn, is handled effectively in unstructured systems. However, it compromises with the efficiency of data query and lookups are not guaranteed. On the other hand, structured overlay networks provide deterministic bounds on data discovery. They provide scalable network overlays based on a distributed data structure which actually supports the deterministic behavior for data lookup. Recent trend in designing structured overlay architectures is the use of distributed hash tables (DHTs) [18, 20, 27]. Such overlay architectures can offer efficient, flexible, and robust service [14, 18, 20, 27, 29]. However, maintaining DHTs is a complex task and needs substantial amount of effort to handle the problem of churn. So, the major challenge facing such architectures is how to reduce this amount of effort while still providing an efficient data query service. In this direction, there exist several important works, which have considered designing DHT-based hybrid systems [7, 13, 16, 26, 30]; these works attempt to include the advantages of both structured and unstructured architectures. However, these works have their own pros and cons. Another design approach has attracted much attention; it is non-DHT based structured approach [4, 9, 17, 21, 24]. It offers advantages of DHT-based systems, while it attempts to reduce the complexity involved in churn handling. Authors in [21] have considered one such approach and have used an already existing architecture, known as Pyramid tree architecture originally applied to the research area of 'VLSI design for testability' [8, 19]. Our structured architecture is an interest-based peer-to-peer system [1, 5, 10, 11-12, 17, 21, 24-25, 28]. In such a system, peers with a common interest are clustered together. Its main focus is to improve the efficiency of data lookup protocols in that a query for an instance of a particular resource type is always directed to the cluster of peers which possess different instances of this resource type. So,

[*] Department of Computer Science.
[†] School of Computing Sciences & Computer Engineering.
[‡] School of Computing.
[§] School of Computing and Information Technology.

success or failure to get an answer for the query involves a search in that cluster only, instead of searching the whole overlay network as in the case of unstructured networks.

The overlay network considered in this paper is a 2-layer non DHT based architecture [21]. At layer-1, there exists a tree like structure, known as pyramid tree. It is not a conventional tree. A node *i* in this tree represents the cluster-head of a cluster of peers which possess instances of a particular resource type R_i (i.e., peers with a common interest). The cluster-head is the first among these peers to join the system. Layer 2 consists of the different clusters corresponding to the cluster-heads. Details of the architecture appears in the next section.

Related works and our Contribution: Before we state our present contributions, we briefly state now some of our recent related contributions. The pyramid tree architecture was initially used in the area of Testable Fault Tolerant Arrays of Processors [8, 19]. Later we realized its potential as a probable network architecture for P2P communication systems especially for interest-based systems. In [21] authors studied extensively what the architecture could offer from the viewpoints of data look-up efficiency and they identified several interesting architectural properties (stated in the Preliminaries) that finally led to the design of various simple yet very efficient data look-up protocols [15, 22-23]. It is a non-DHT-based two-level structured architecture and experimental results [23] have shown the superiority of the proposed various data look-up protocols when compared with the protocols used in some noted DHT-based structured networks from the viewpoints of search latency and complexity involved. It offers as well several advantages when compared with some noted works on interest-based architectures [1, 5, 10, 11-12, 25, 28]. Authors have extensively studied the effect of churn on the architecture [23]; besides another important contribution was the design of intra-capacity constrained broadcast and multicast protocols which take into consideration the real situation where peers most likely will be heterogeneous [22].

However, we believe that all these above-mentioned recent contributions on interest-based architectures still lack in one very important aspect: in these architectures, the underlying assumption is that no peer can have more than one resource type and this could be a very hard restriction practically. Therefore, in the present work, we have addressed this issue of generalizing pyramid tree based P2P architecture and have come up with effective solutions that allow a peer to possess multiple different resource types.

The organization of the paper is as follows. In Section 2, we talk briefly about some related preliminaries. Our contributions in the present paper appear in Sections 3 and 4. In Section 3, generalization of the architecture has been considered. In Section 4, effect of the generalization on the existing communication protocols [15, 22-23] has been considered. Section 5 draws the conclusion.

2 Preliminaries

In this section, we present some relevant results from our recent works on the Pyramid tree based P2P architecture [15, 21-23]. for interest-based peer-to-peer system. Residue Class based on modular arithmetic has been used to realize the overlay topology.

Definition 1. We define a resource as a tuple <R_i, V>, where R_i denotes the type of a resource and V is the value of the resource.

Note that a resource can have many values. For example, let R_i denote the resource type 'songs' and V' denote a particular singer. Thus <R_i, V'> represents songs (some or all) sung by a particular singer V'.

Definition 2. Let S be the set of all peers in a peer-to-peer system with n distinct resource types (i.e., n distinct common interests). Then S = {C_i}, $0 \le i \le n-1$, where C_i denotes the subset consisting of all peers with the same resource type R_i. In this work, we call this subset C_i as cluster i. Also, for each cluster C_i, we assume that C_i^h is the first peer among the peers in C_i to join the system. We call C_i^h as the cluster-head of cluster C_i.

2.1 Pyramid Tree

The following overlay architecture has been proposed in [21].

- The tree consists of n nodes. The ith node is the ith cluster head C_i^h. The tree forms the layer-1 and the clusters corresponding to the cluster-heads form the layer-2 of the architecture.
- Root of the tree is at level 1.
- Edges of the tree denote the logical link connections among the n cluster-heads. Note that edges are formed according to the pyramid tree structure [8].
- A cluster-head C_i^h represents the cluster C_i. Each cluster C_i is a completely connected network of peers possessing a common resource type R_i, resulting in the cluster diameter of 1.
- The tree is a complete one if at each level j, there are j number of nodes (i.e., j number of cluster-heads). It is an incomplete one if only at its leaf level, say k, there are less than k number of nodes.
- Any communication between a peer $p_i \epsilon C_i$ and a peer $p_j \epsilon C_j$ takes place only via the respective cluster-heads C_i^h and C_j^h and with the help of tree traversal wherever applicable.
- Joining of a new cluster always takes place at the leaf level.
- A node that does not reside either on the left branch or on the right branch of the root node is an internal node.
- Degree of an internal non-leaf node is 4.
- Degree of an internal leaf node is 2.

2.2 Residue Class

Modular arithmetic has been used to define the pyramid tree architecture of the P2P system.

Consider the set S_n of nonnegative integers less than n, given as $S_n = \{0, 1, 2, …. (n – 1)\}$. This is referred to as the set of residues, or residue classes (mod n). That is, each integer in S_n represents a residue class (RC). These residue classes can be labelled as [0], [1], [2], …, [n – 1], where [r] = {a: a is an integer, a ≡ r (mod n)}.

For example, for n = 3, the classes are:

$[0] = \{…., -6, -3, 0, 3, 6, …\}$

$[1] = \{…., -5, -2, 1, 4, 7, …\}$

$[2] = \{…., -4, -1, 2, 5, 8, …\}$

In the P2P architecture, we use the numbers belonging to different classes as the logical (overlay) addresses of the peers with a common interest (i.e., peers in the same cluster) and the number of residue classes is the number of distinct resource types; for the sake of simplicity, we shall use only the positive integer values.

Before we present the mechanism of logical address assignments, we state the following relevant property of residue class.

Lemma 1. Any two numbers of any class r of S_n are mutually congruent [15, 21].

2.3 Assignments of Overlay (Logical) Addresses

Assume that in an interest-based P2P system there are n distinct resource types. Note that n can be set to an extremely large value a priori to accommodate large number of distinct resource types. Consider the set of all peers in the system given as $S = \{C_i\}$, $0 \leq i \leq n\text{-}1$. Also, as mentioned earlier, for each subset C_i (i.e., cluster C_i) peer C_i^h is the first peer with resource type R_i to join the system and hence, it is the cluster-head of cluster C_i.

The assignment of overlay addresses to the peers in the clusters and the resources happens as follows:

1) The first cluster-head to join the system is assigned with the logical (overlay) address 0 and is denoted as C_0^h. It is also the root of the tree formed by newly arriving cluster-heads (see the example in Figure 1).
2) The $(i+1)^{th}$ newly arriving cluster-head possessing the resource type R_i is denoted as C_i^h and is assigned with the minimum nonnegative number (*i*) of *residue class i (mod n)* of the residue system S_n as its overlay address.
3) In this architecture, cluster-head C_i^h is assumed to join the system before the cluster-head C_{i+1}^h.
4) All peers having the same resource type R_i (i.e., 'common interest' defined by R_i) will form the cluster C_i. Each new peer joining cluster C_i is given the cluster membership address (i + j.n), for i = 0, 1, 2, …
5) Resource type R_i possessed by peers in C_i is assigned the code *i* which is also the logical address of the cluster-head C_i^h of cluster C_i.

Definition 3. Two peers of a cluster C_r are logically linked together if their assigned logical addresses are mutually congruent.

Lemma 2. Each cluster C_r forms a complete graph [15].

Observation 1. Any intra-cluster data look up communication needs only one overlay hop.

Observation 2. Search latency for inter-cluster data lookup algorithm is bounded by the diameter of the tree.

Scalability: It may be noted that number of distinct resource types is very small compared to the number of peers in any overlay network [15]. To avoid the possibility of redesigning the architecture as new clusters are formed, a very large value of n can be selected at the design phase to accommodate a very large number of possible resource types (if needed in the future). It means that if at the beginning number of resource types present is small, only the first few of the residue classes will be used initially for addressing; and as new clusters are formed in future with new resource types in the system, more residue classes in sequence will be available for their addressing. For example, say initially n is set at 1000; so, there are 1000 possible residue classes, starting from [0], [1], [2], [4],[5], …., [999]. If initially there are only three clusters of peers present with three distinct resource types, the residue classes [0], [1], [2] will be used for addressing the peers in the three respective clusters. If later two new clusters are formed with two new resource types, the residue classes [3] and [4] will be used for addressing the peers in the two new clusters in sequence of their joining the system. Moreover, as we see, there is no limit on the size of any cluster because any residue class can be used to address logically up to an infinite number of peers with a common interest. Therefore, the proposed architecture does not have any negative issue with scalability.

2.4 Virtual Neighbors [23]

An example of a complete pyramid tree of 5 levels is shown in Figure 1. It means that it has 15 nodes/clusters (clusters 0 to 14, corresponding to 15 distinct resource types owned by the 15 distinct clusters). It also means that residue class with *mod 15* has been used to build the tree. The nodes' respective logical addresses are from 0 to 14 based on their sequence of joining the P2P system.

Each link that connects directly two nodes on a branch of the tree is termed as a *segment.* In Figure 1, a bracketed integer on a segment denotes the difference of the logical addresses of the two nodes on the segment. It is termed as *increment* and is denoted as *Inc.* This increment can be used to get the logical address of a node from its immediate predecessor node along a branch. For example, let X and Y be two such nodes connected via a segment with increment *Inc*, such that node X is the immediate predecessor of node Y along a branch of a tree which is created using *residue class with mod n.* Then, *logical address of Y = (logical address of X + Inc) mod n.*

Thus, in the example of Figure 1, Logical address of the leftmost leaf node = (logical address of its immediate predecessor along the left branch of the root + Inc) mod 15 = (6

+ 4) mod 15 = 10.

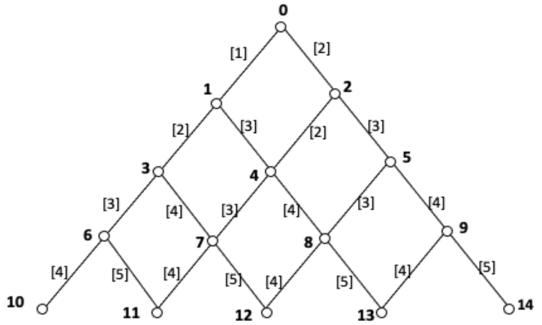


Figure 1: A complete pyramid tree with root 0

Also, note that a *left branch* originating at node 2 on the right branch of the root node is *2 → 4 → 7 → 11*. Similarly, we can identify all other left branches originating at the respective nodes on the right branch of the root node. In a similar way, we can identify as well all right branches originating at the respective nodes on the left branch of the root node as well.

Remark 1. The sequence of increments on the segments along the left branch of the root, appears to form an AP series with 1st term as 1 and common difference as 1.

Remark 2. The sequence of increments on the segments along the right branch of the root, appears to form an AP series with 1st term as 2 and common difference as 1.

Remark 3. Along the 1st left branch originating at node 2, the sequence of increments appears to form an AP series with 1st term as 2 and common difference as 1. Note that the 1st term is the increment on the segment 0 → 2.

Remark 4. Along the 2nd left branch originating at node 5, the sequence of increments is an AP series with 1st term as 3 and common difference as 1. Note that the 1st term is the increment on the segment 2 → 5.

Authors [21] have presented some important structural properties of the pyramid tree P2P system. According to the authors, no existing structured P2P system, either DHT or non-DHT based, possesses these properties. These are stated below.

Let S_Y be the set of logical links that connect a node Y to its neighbors in a complete pyramid tree T_R with root R. Assume that the tree has n nodes (i.e., n cluster heads / n clusters). Let another tree T'_R be created with the same n nodes but with a different root R'. Let S'_Y be the set of logical links connecting Y to its neighbors in the tree T'_R.

Property 1. $S_Y \neq S'_Y$

Property 2. Diameter of T_R = Diameter of T'_R

Property 3. Number of levels of T_R = Number of levels of T'_R

Property 4. Complexity of broadcasting in T_R with root R as the source of broadcast is the same for T'_R with root R'

Property 5. Both T_R and T'_R are complete pyramid trees.

An example: Consider the complete pyramid tree of 5 levels as shown in Figure 2. Note that the root of this tree is node 13, whereas root of the tree of Figure 1 is 0.

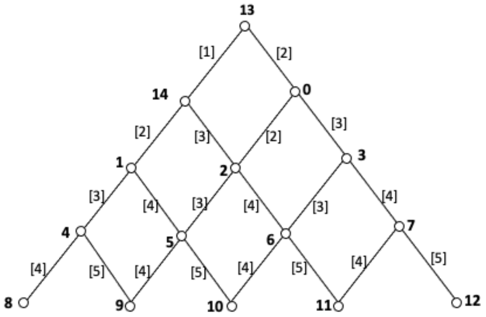


Figure 2: A complete pyramid tree with root 13

It is seen that S'_4 = {1,8,9} and S_4 = {1,2,7,8}. Therefore, Property 1 holds.

Diameters of both trees are the same; it is 8 in terms of number of overlay hops. Besides, both trees use the same 15 nodes and have the same total number of levels. Complexity of broadcasting from either root 0 in the tree of Figure 1 or from root 13 in the tree of Figure 2 is bounded by the number of levels of each of the trees (here it is 4 in each). Finally, both trees are complete pyramid trees. Thus, all properties as mentioned above hold.

Remark 5. Set of the neighbors of a given node Z may vary as the root of the tree varies. Hence, it is termed 'virtual'. However, time complexity of broadcasting remains same, i.e., it is O(d) where *d* denotes the number of levels of the tree.

3 Generalization of the Architecture

As mentioned earlier, in the architecture, it is assumed that no peer can have more than one resource type and this could be a very hard restriction practically. To overcome this restriction, we have come up with the concept of *Generalization* i.e., the architecture is generalized in such a way that a peer can have multiple resource types. Generalization of the Architecture needs to deal with two possible scenarios. Below we consider the two possible scenarios and state how to incorporate some necessary changes in the architecture in order to handle the two scenarios. Throughout our presentations, we shall interchangeably use the words 'node' and 'cluster-head'. So, a node on the tree is actually a cluster-head. These are all peers though. However, we strictly use the word 'peer' to represent members of a cluster only to avoid any possible confusion. In addition, we assume that 'resource with type k' and 'resource with code k' mean the same resource.

3.1 Peer with Multiple Existing Resource Types

Scenario 1: Let us consider a situation that in some cluster

C_i, its cluster-head C_i^h or a peer p in C_i wants data insertion of another existing resource type, say R_k in the network. Here data-insertion by a peer means the peer in question declares the possession of instances of another resource type that already exists in the system.

As mentioned earlier, peers in cluster C_k possess instances of the resource type R_k. Also, peer p in C_i already possesses some instances of the resource type R_i.

Solution*:* The solution for this scenario is as follows. The cluster-head C_i^h or peer p will now become a member of cluster C_k as well. So, it is understood that the IP address of C_i^h /p will be known to members of both the clusters C_i and C_k. It means that, in the overlay network, C_i^h /p will appear logically in both the clusters C_i and C_k, and will have direct logical connections to all member peers of clusters C_i and C_k. Therefore, it should be clear that our already reported intra- and inter-cluster data-lookup protocols [28] do not need any modification in this scenario. The same is true for broadcasting involving the cluster-heads in the tree. In addition, we have observed that the capacity-constrained broadcast and multicast protocols inside a cluster [20] in the tree need not be modified as well.

However, we observe that the existing inter-cluster data look-up protocol as well as the broadcast protocol involving all cluster-heads in the tree [15] will need some appropriate modifications to handle the second scenario. We shall present these in detail in the following sections. Before that we present the following solution to tackle the second scenario.

3.2 Existing Peers Declaring New Resource Types

Scenario 2: Consider a P2P interest-based pyramid tree system which has currently r distinct resource types, viz., R_0, R_1, R_2, … R_{r-1}. Assume that the respective resource codes are 0, 1, 2, …, (r-1). Without any loss of generality, let us assume a scenario where cluster-head C_i^h / a peer p in a cluster C_i wants a data insertion of a new resource type R_r currently not present in the network.

Solution*:* Solution lies in an appropriate modification of the table of information (*TOI*) maintained by each cluster-head. We know that in *TOI*, corresponding to each cluster-head there is an entry (tuple). For example, the tuple for some cluster-head C_i^h appears as <resource code (logical address) owned by peers in C_i^h, IP address of the cluster-head C_i^h >; note that in the architecture resource code and the logical address of a cluster-head are the same. That is, one denotes the other. It facilitates packet propagation in the tree. In short, we write the tuple as < Res. Code, IP (C_i^h) >. As new clusters are formed owing to peers joining with new resource types, the *TOI* grows dynamically and the newest joining cluster-head is assigned with the next largest logical address not yet used and hence its resource code also becomes the largest among all such existing codes. Therefore, this table remains sorted with respect to logical addresses of cluster-heads (i.e., with respect to the resource codes of the resources they possess).

Coming back to the second scenario, a new entry is made in the *TOI* corresponding to the new resource type R_r with resource code r. So currently this code r is the largest one present in the table. Based on if it is the cluster-head C_i^h / or a peer in cluster C_i that wants a data insertion of a new resource type R_r, in the newly entered tuple, the corresponding cluster-head will be either C_i^h or the peer p. That is, if it is C_i^h, it will now represent two different clusters corresponding to two different resource types i and r. So, it will have two different logical addresses i and r as well. Therefore, later any peer wishing to join with resource type r will join the cluster with logical address r. Effectively, C_i^h now will maintain two different clusters C_i and C_r, i.e., one with peers for resource code i and the other with peers with resource code r. It is clear that cluster-head in the second case with resource type r is now C_r^h which is actually C_i^h. In case it is the peer p in cluster C_i, peer p will maintain a cluster of peers with resource type r; thus, p will appear as a peer in Cluster C_i and will also appear as a cluster-head C_r^h with logical address r. Therefore, we have modified the *TOI* to include the relevant information of the new entry. Below we give an example to clear the idea further.

Observation 3. Generalization of the architecture may require some nodes of the tree represent multiple cluster-heads with the same IP address, but with different distinct resource types.

Example 1: Without any loss of generality let us consider a 3-level complete pyramid tree. Thus, the tree has six distinct resource types with respective resource codes as 0, 1, 2, 3, 4, 5. According to the structure of the tree node 0 is at level 1, nodes 1 and 2 at level 2, and nodes 3,4, and 5 are at level 3. Next, assume that cluster-head C_1^h declares that it has just possessed some instances of another new resource type with 6 as its code. Now, the tree becomes a 4-level incomplete tree with seven nodes (i.e., seven cluster-heads) with node 6 at level 4. Therefore, as explained above, *TOI* needs to be modified. Before and after the above declaration *TOI* appears as shown below in Figures 3a and 3b. We denote IP address of a node X as IP(X). ‘Res. Code’ is actually ‘Resource Code’.

Note that in Figure 3b cluster-head C_1^h has appeared twice: once it represents a cluster-head with logical address 1 and next with logical address 6, appearing as C_6^h. That is, C_1^h now represents virtually two different clusters of peers C_1 and C_6, one with instances of resource type with code 1 and the other one with code 6. In effect, the 2nd appearance of C_1^h as C_6^h makes the tree incomplete with 7 nodes.

Note that if instead of the cluster-head C_1^h some peer, say p* in cluster C_i declares that it has just possessed another new resource type with 6 as its code, the entry for resource code 6 will become <6, IP(p*)> in Figure 3b. Hence, the new cluster-head p*(i.e., C_6^h) forms a cluster with peers willing to join with instances of resource type 6.

It may be noted that any inter-cluster query for some instance of the resource type 6 will be directed at either C_1^h or p* depending on the tuple corresponding to resource code 6 (Figure 3b).

Res. Code	IP address
0	IP(C_0^h)
1	IP(C_1^h)
2	IP(C_2^h)
3	IP(C_3^h)
4	IP(C_4^h)
5	IP(C_5^h)

Figure 3a: *TOI* before declaration

Res. Code	IP address
0	IP(C_0^h)
1	IP(C_1^h)
2	IP(C_2^h)
3	IP(C_3^h)
4	IP(C_4^h)
5	IP(C_5^h)
6	IP(C_6^h)

Figure 3b: *TOI* after declaration: IP(C_1^h) = IP(C_6^h)

4 Modification of Existing Inter-Cluster Data Look-Up and Broadcast Protocols

As pointed out earlier the existing inter-cluster data look-up and broadcast protocols (involving all cluster-heads in the tree) will need some appropriate modifications to handle only the second scenario. In this section we deal with this. We again emphasize that none of the two scenarios have any effect on the existing capacity-constrained broadcast and multicast protocols inside a cluster [22] i.e., as long as the communication is inside a cluster only, no related existing protocols need be modified.

4.1 Modified Inter-Cluster Data Look-Up Protocol

In the generalized protocol stated below, codes from line 2 to line 3 are added to handle the second scenario. This section of the total code deals with the situation when a peer represents multiple cluster-heads with each cluster having distinct resource types. In the architecture, any communication between a peer $p_i \in C_i$ and a peer $p_m \in C_m$ takes place only via the respective cluster-heads C_i^h and C_m^h. Without any loss of generality let a peer p_i* ($\in C_i$) request for <R_m, V*>. Note that peer p_i* knows that $R_m \notin C_i$, because resource code used in cluster C_i is i. The protocol appears in Figure 4 below.

Protocol Generalized Inter-Data-Lookup

```
1. $p_i$* sends a data lookup request for <$R_j$, V*> to its cluster-head $C_i^h$
2. if  $IP(C_i^h) = IP(C_m^h)$      / $C_i^h$ checks in its TOI; same peer acts as cluster-heads for clusters $C_i$ and $C_m$
          if  $C_m^h$ possesses <$R_m$, V*>
               $C_m^h$ unicasts <$R_m$, V*> to $p_i$*
          else
               $C_m^h$ broadcasts the request for <$R_m$, V*> in $C_m$   / one hop communication
                    if ∃ $p_m$ ($\in C_m$) with <$R_m$, V*>
                         $p_m$ unicasts <$R_m$, V*> to $p_i$*
                    else
3.                  $C_m^h$ unicasts 'search fails' to $p_i$*
4.  else
          $C_i^h$ determines the cluster-head $C_m^h$'s IP address from its TOI using $C_m^h$'s resource code
                                        / logical address of $C_m^h$ = resource code of $R_m$ = m
          $C_i^h$ unicasts the request to $C_m^h$
          if  $C_m^h$ possesses <$R_m$,V*>
               $C_m^h$ unicasts <$R_m$,V*> to $p_i$*
          else
               $C_m^h$ broadcasts the request for <$R_m$,V*> in $C_m$   / one hop communication
                    if  ∃ $p_m$ ($\in C_m$) with <$R_m$,V*>
                         $p_m$ unicasts <$R_m$,V*> to $p_i$*
                    else
                         $C_m^h$ unicasts 'search fails' to $p_i$*
```

Figure 4: Modified generalized inter-cluster data-lookup protocol

As in Observation 2 earlier, search latency for modified inter-cluster data look-up approach remains bounded by the diameter of the tree and is independent of the total number of peers present in the system.

4.2 Modified Broadcast Protocol

In the context of broadcasting, it may be noted that, in general, inter-cluster broadcast involves always intra-cluster broadcast as well, with the exception when a cluster-head wants to update some control information (ex. broadcasting of updated *TOI* by the root of tree) maintained only by different cluster-heads in the system. Therefore, we focus specifically on broadcasting by a cluster-head C_i^h of a cluster C_i on the tree to all other cluster-heads.

An interesting observation is that if the root is node 0 (logical address), even an incomplete tree always remains a connected one; on the other hand, for any other cluster-head as root, an incomplete tree may not remain connected. To explain the idea briefly and clearly, consider the complete tree of Figure 1. Its root is node 0. If root changes to some other node, say node 13, the tree still remains a complete one as is shown in Figure 2. This property of the architecture arising from 'virtual neighbors' has been discussed in detail earlier. Now assume that the tree in Figure 1 does not have node 14, i.e., cluster 14 is yet to be formed. So the tree is incomplete, yet it is connected. In this situation, if node 0 broadcasts some packets, all other nodes will receive copies. Now assume that node 13 is the broadcaster and node 13 is assumed to be the root. It is seen that its immediate neighbor on its left branch should be node 14 (Remark 1); however node 14 does not exist in the tree as assumed above. So propagation of any broadcast packet along the left branch cannot take place based on the Broadcast-Complete protocol [15]. Therefore, broadcast fails. However, if node 13 unicasts its packets first to node 0 (root) which will then act as the pseudo broadcaster on behalf of node 13, all nodes get copies because the tree remains connected with node 0 as its root. This is actually the idea used in the Broadcast-Incomplete protocol [15]. This has led us to consider modifying only the existing Broadcast-Incomplete protocol to appropriately handle the second scenario.

An informal sketch of the modified incomplete broadcast protocol

Step 1: A broadcast source node X will unicast its packets to the root node 0.

Step 2: Root 0 sends packets to its neighbors on left and right branches.

Step 3: Each receiving node on the left branch sends packets to its neighbor on this branch till a receiving node is a leaf node.

Step 4a: The i^{th} receiving node on the right branch sends packets to its neighbor on the i^{th} left branch originating at the i^{th} node until the i^{th} receiving node is a leaf node.

Step 4b: The i^{th} receiving node sends packets to its neighbor, the $(i+1)^{th}$ node on the right branch until the i^{th} receiving node is a leaf node.

Step 4c*:* Propagation along the i^{th} left branch continues as in Step 3.

In the above informal sketch, a broadcast source node X will unicast its packets to the root node 0, which in turn, will execute a modified version of the broadcast-incomplete protocol. That is, node 0 will act as the pseudo broadcast source (like in CBT multicast [2] the core is the pseudo multicast source). Hence, the tree will remain connected with node 0 as its root even if the original tree is an incomplete one. This justifies our consideration to modify only the existing broadcast-incomplete protocol.

Since node X is a part of the tree, eventually it will participate in the broadcast by node 0 and will receive a copy of the packet which it already unicasted to node 0. Note that node X may need to forward the received packet further depending on its location on the tree. Therefore, this approach will generate only one duplicate packet per broadcast packet irrespective of the size of the tree. The formal presentation of the protocol appears in Figure 4.

It may be noted that instead of using the left branches originating at nodes on the right branch (as in step 4 above), the protocol can use the right branch of the root and all right branches emanating from the nodes on the left branch of the root. In this way, it will also generate only one duplicate packet per packet broadcast as well. We use the following data structures and notations.

The structure of broadcast packet, BP appears as: < # hops (N_h), increment (Inc), flag (L/R), Information (Info) >

Interpretation of the different entries in the broadcast packet P is stated below.

# hops (N_h): is initialized by the broadcast source X with (d-1); each receiving node on the tree along a propagation path will decrement N_h by 1, before forwarding the received packet to the next node along the path.

Increment (Inc): is used to determine the logical address of the next node for packet forwarding.

Flag (L/R): it is either L or R. Flag L denotes that a received packet needs to be propagated along a left branch until the leaf level is reached. Similarly, flag R denotes packet propagation along a right branch. For ease of understanding the protocols we name the broadcast packet BP as BP_L if flag is L; otherwise we name it BP_R.

Info: denotes the actual information to broadcast.

Address (X): logical address of node X

IP address of X: IP(X)

In this context, it may be mentioned that if cluster-head C_0^h (i.e. node 0) along with its all member peers in C_0 have left the network, the cluster-head with current logical address as 1 assumes the role of the root of the tree and its logical address becomes 0 and at the same time any other cluster-head with logical address H will have its newly assigned logical address as (H-1); the table of information (TOI) will be updated accordingly, which will reflect a new, possibly incomplete, yet connected, tree with its root as node 0 (formerly node 1). However, it is all about ‘churn handling’ which has already been reported in detail in [23]. Therefore, in the following algorithm by ‘node 0’ it means the current root.

We have modified the existing broadcast-incomplete protocol [15] to incorporate the solution for scenario 2 as discussed in the previous section. The modified portion appears on lines 12 to 14 (Figure 5) and it resolves the issue raised in scenario 2. This small piece of code is crucial in the modified protocol. Below, we have explained its importance considering again Example 1.

Initially peers in cluster C_1 have instances of resource type with code 1. Assume that later cluster-head C_1^h declares that it has just possessed another new resource type with 6 as its code. Therefore, now cluster-head C_1^h represents virtually two clusters, one consisting of peers possessing resource type with code 1 and the other with code 2 (refer to Figure 3b). Now without any loss of generality we shall consider the following three possible cases.

Case 1. Assume that node 2 (i.e., cluster-head C_2^h) is the source of broadcast (appeared as X in the protocol). It starts unicasting its broadcast packets to the root node 0 which then broadcasts the packets to all nodes (cluster-heads) following the

Protocol Generalized-Broadcast-Incomplete

```
Executed by broadcast source X
    Node X unicasts packets to node 0 for broadcasting
Executed by root node 0    // node 0 acts as the pseudo broadcast source
    $N_h = N_h-1$                     / node 0 builds a broadcast packet $BP_L$
    Inc = 1
    flag = L                          / n = number of residue classes = number of
    $BP_L$ packet = < $N_h$, Inc, L, Info >     distinct resource types /
    Node 0 forwards the $BP_L$ packet to the node with address,     propagation along the left branch of node 0
      [(address (X) + Inc) mod n]                                    takes place
    $N_h = N_h-1$                     / node 0 builds a broadcast packet $BP_R$
    Inc = 2
    flag = R
    $BP_R$ packet = < $N_h$, Inc, R, Info >     / propagation along the right branch of node
    Node 0 forwards the $BP_R$ packet to the node with address,     0 takes place
      [(address (X) + Inc) mod n]
Executed by a receiving node $C_i^h$
    if $C_i^h \neq X$
        if IP($C_i^h$) ≠ IP(X)
            $C_i^h$ keeps a copy
        else $C_i^h$ does not keep a copy   / $C_i^h$ has multiple distinct resource types;
                                            already has copy of every packet since it is the
                                            source X
    else does not keep a copy
    if $N_h = 1$                      / it is a leaf level node
        stops forwarding
    else
        if flag = L in the received packet
            if [(address ($C_i^h$) + (Inc + 1)) mod n] ≤ $N_{max}$   / $N_{max}$ is the largest current logical address in
                                                                     the tree
                $N_h = N_h-1$         / build a new $BP_L$ packet
                Inc = Inc+1           / n = number of residue classes = number of
                new $BP_L$ packet = < $N_h$, Inc, L, Info >     distinct resource types
                $C_i^h$ forwards the $BP_L$ packet to the node   / propagation along the left branch of $C_i^h$
                  with address, [(address ($C_i^h$) + Inc) mod n]   continues
            else
                stops forwarding      / no such address exists; tree is incomplete
```

```
26.   else                                              / flag is R and $C_i^h$ are on the right branch of the broadcast source
27.       if [(address ($C_i^h$) + (Inc)) mod n] ≤ $N_{max}$
28.           Inc = Inc in the received $BP_R$ packet       / build a new $BP_L$ packet
29.           $N_h = N_h-1$
30.           flag = L
31.           new $BP_L$ packet = < $N_h$, Inc, L, Info >
32.           $C_i^h$ forwards the new $BP_L$ packet to the
              node with address, [(address ($C_i^h$) + Inc) mod n]   / propagation along the left branch of $C_i^h$ continues
33.       else
34.           stops forwarding                          / no such address exists; tree is incomplete
35.       if [(address ($C_i^h$) + (Inc + 1)) mod n] ≤ $N_{max}$
36.           $N_h = N_h-1$                                 / build a new $BP_R$ packet
37.           Inc = Inc+1
38.           flag = R
39.           new $BP_R$ packet = < $N_h$, Inc, R, Info
              >                                         / propagation along the right branch of $C_i^h$ continues
40.           $C_i^h$ forwards to the node with
              address,
              [(address ($C_i^h$) + Inc) mod n]
41.       else                                          / no such address exists; tree is incomplete

42.           stops forwarding
```

Figure 5: Protocol Generalized- Broadcast-Incomplete

protocol. However, it is observed that every node receives exactly one copy of each packet except node 2, because node 2 will also receive copies of its already unicasted packets from the root node 0 because of broadcasting. So, it discards the received copies. This has appeared in the protocol as stated in *italics* below.

```
12.   if $C_i^h \neq X$
          if IP($C_i^h$) ≠ IP(X)
              $C_i^h$ keeps a copy
13.       else $C_i^h$ does not keep a copy
14.   else   does not keep a copy
```

Case 2. Assume that node 1 is the source, i.e., $C_1^h = X$. Cluster-head C_1^h has appeared twice on the tree once with logical address 1 and another with logical address 6. So, C_1^h itself being the source of broadcast (as node 1), will discard the packets when it receives as node 6 from root node 0 because of broadcasting. This has appeared in the protocol as stated in *italics* below.

```
12.   if $C_i^h \neq X$
          if IP($C_i^h$) ≠ IP(X)
              $C_i^h$ keeps a copy
13.       else $C_i^h$ does not keep a copy
14.   else   does not keep a copy
```

Case 3. If logical addresses of the broadcast node X and a receiving node are different, still there is a chance that both nodes are actually the same one (Case 2 above). However, if not, their IP addresses will differ and the receiving node will keep copies of the received packets. This has appeared in the protocol as stated in *italics* below.

```
12.    if  $C_i^h \neq X$
           if  $IP(C_i^h) \neq IP(X)$
               $C_i^h$ keeps a copy
           else  $C_i^h$ does not keep a copy
       else   does not keep a copy
```

Theorem 1. The protocol generates exactly one extra copy per packet broadcast.

Proof. Propagation of the broadcast information (Info) takes place along the left and right branches of the root node; also, it takes place along all left branches originating at all nodes on the right branch. Propagation stops when a receiving node is a leaf node. Therefore, all nodes receive the broadcast information.

Besides, each broadcast packet is received only once by each node on the tree except the source node X. Since node X is a part of the tree, eventually it will participate in the broadcast by node 0 and will receive a copy of the packet which it already unicasted to node 0. Therefore, there is only one extra packet generated per packet broadcast.

Complexity: The hop complexity is O(d) and as in Broadcast-incomplete protocol [15] complexity is dependent only on the number of the distinct resource types n present in the system, which in turn determines the value of the number of levels d of the tree.

Bandwidth Utilization: It offers very high bandwidth utilization because it generates only one duplicate packet per broadcast packet and the number of such duplicate packets is independent on the total number of peers present in the network.

Remark 6. The proposed method to generalize the architecture is remarkably simple and efficient, and it does not affect the existing intra-cluster data look-up protocol; it affects the existing inter-cluster data look-up and the broadcast protocol is arguably a minimal way (as is observed in the modified portion of the original pseudo code in case of the broadcast protocol).

5 Conclusion

Authors, in recent studies, have exploited the architectural properties of the Pyramid tree P2P network to design different communication protocols with reasonably low search latency. However, these recent contributions still lack in one very important aspect (like other existing interest-based architectures): in the architecture, it is assumed that no peer can have more than one resource type and this could be a very hard restriction practically. In the present work, we have addressed this issue of generalizing the architecture and have come up with effective solutions. Effect on the architecture due to generalization is just reflected in the modified 'Table of Information' which is actually consulted by the various protocols for data/query propagation. We have shown that generalization has no effect on the existing intra-cluster data look-up protocol. Only the existing inter-cluster data look up and the broadcast protocols need to be modified. The two modified protocols have the same low bandwidth requirements and look-up complexities as those of the already existing ones.

As a continuation of our research, we are now working on designing protocols for secured communication in the generalized architecture.

References

[1] L. Badis, M. Amad, D. Aîssani, K. Bedjguelal and A. Benkerrou, "ROUTIL: P2P Routing Protocol Based on Interest Links," 2016 International Conference on Advanced Aspects of Software Engineering (ICAASE), Constantine, pp. 1-5, 2016, doi: 10.1109/ICAASE. 2016.7843852.

[2] Tony A. Ballardie, "Core Based Tree Multicast Routing Architecture, Internet Engineering Task Force (IETF), RFC 2201, September 1997.

[3] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable," Proc. ACM SIGCOMM, Karlsruhe, Germany, pp. 407-418, August 25-29, 2003.

[4] Shiping Chen, Baile Shi, Shigang Chen, and Ye Xia, "ACOM: Any-Source Capacity-Constrained Overlay Multicast in Non-DHT P2P Networks," *IEEE Tran. Parallel and Distributed System*s, 18(9)1188-1201, Sep. 2007.

[5] Wen-Tsuen Chen, Chi-Hong Chao and Jeng-Long Chiang, "An Interested-based Architecture for Peer-to-Peer Network Systems," 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06), Vienna, pp. 707-712, 2006, doi: 10.1109/AINA.2006.93.

[6] Jie Cheng and Ryder Donahue, "The Pirate Bay Torrent Analysis and Visualization," *IJCSET*, 3(2):38-42, Feb. 2013.

[7] P. Ganesan, Q.Sun, and H. Garcia-Molina, "Yappers: A Peer-to-Peer Lookup Service over Arbitrary Topology," Proc. IEEE Infocom 2003, San Francisco, USA, pp. 1250-1260, March 30-April 1 2003.

[8] Bidyut Gupta and Mohammad Mohsin, "Fault-Tolerance in Pyramid Tree Network Architecture," *J. Computer Systems Science and Engineering*, 10(3):164-172, July,1995.

[9] Bidyut Gupta, Nick Rahimi, Shahram Rahimi, and Ashraf Alyanbaawi, "Efficient Data Lookup in Non- DHT Based Low Diameter Structured P2P Network," Proc. IEEE 15th Int. Conf. Industrial Informatics (IEEE INDIN), Emden, Germany, pp.143-148, July 2017.

[10] M. Hai and Y. Tu, "A P2P E-Commerce Model Based on Interest Community," 2010 International Conference on Management of e-Commerce and e-Government, Chengdu, pp. 362-365, 2010, doi: 10.1109/ICMeCG. 2010.80.

[11] Mujtaba Khambatti, Kyung Ryu, and Partha Dasgupta,. "Structuring Peer-to-Peer Networks Using Interest-Based Communities," Lecture Notes in Computer Science, 1st International Workshop, DBISP2P 2003, Berlin, September 2003.

[12] S. K. A. Khan and L. N. Tokarchuk, "Interest-Based Self Organization in Group-Structured P2P Networks," 2009 6th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, pp. 1-5, 2009, doi: 10.1109/CCNC.2009.4784959.

[13] M. Kleis, E. K. Lua, and X. Zhou, " Hierarchical Peer-to-Peer Networks using Lightweight SuperPeer Topologies," Proc. IEEE Symp. Computers and Communications, pp. 944-950, 2005.

[14] D. Korzun and A. Gurtov, "Hierarchical Architectures in Structured Peer-to-Peer Overlay Networks," Peer-to-Peer Networking and Applications, Springer, pp. 1-37, March 2013

[15] Koushik Maddali, Indranil Roy, Swathi Kaluvakuri, Bidyut Gupta, Narayan Debnath, "Design of Broadcast Protocols for Non DHT-Based Pyramid Tree P2P Architecture," *IJCA*, 28(4)193-203, December 2021.

[16] Z. Peng, Z. Duan, J.Jun Qi, Y. Cao, and E. Lv, "HP2P: A Hybrid Hierarchical P2P Network," Proc. Intl. Conf. Digital Society, pp. 18-24, 2007.

[17] N. Rahimi, K. Sinha, B. Gupta, and S. Rahimi, "LDEPTH: A Low Diameter Hierarchical P2P Network Architecture," Proc. IEEE 14th Int. Conf. on Industrial Informatics (IEEE INDIN), Poitiers, France, pp. 832-837, July 2016.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network, CAN," *ACM*, 31(4):161-172, 2001.

[19] Arnold L. Rosenberg, "The Diogenes Approach to Testable Fault-Tolerant Arrays of Processors," *IEEE Tran. Computers*, c-32(10):902-910, Oct. 1983.

[20] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large Scale Peer-to-Peer Systems," Proc. FIP/ACM Intl. Conf. Distributed Systems Platforms (Middleware), pp. 329-350, 2001.

[21] Indranil Roy, Bidyut Gupta, Banafsheh Rekabdar, and Henry Hexmoor, "A Novel Approach Toward Designing A Non-DHT Based Structured P2P Network Architecture," (Proceedings of 32nd Int. Conf. Computer Applications in Industry and Engineering), EPiC Series in Computing, 63(2019):121-129, 2019.

[22] Indranil Roy, Swathi Kaluvakuri, Koushik Maddali, Abdullah Aydeger, Bidyut Gupta, and Narayan Debnath, "Capacity Constrained Broadcast and Multicast Protocols for Clusters in Pyramid Tree-Based Structured P2P Network," *IJCA*, 28(3):122-131, Sep. 2021.

[23] Indranil Roy, Swathi Kaluvakuri, Koushik Maddali, Ziping Liu, and Bidyut Gupta, "Efficient Communication Protocols for Non DHT-Based Pyramid Tree P2P Architecture," *WSEAS Transactions on Computers*, 20:108-125, July 2021 (Invited paper).

[24] Indranil Roy, Koushik Maddali, Swathi Kaluvakuri, Banafsheh Rekabdar, Ziping Liu, Bidyut Gupta, and Narayan Debnath, "Efficient Any Source Overlay Multicast In CRT-Based P2P Networks ─ A Capacity - Constrained Approach," Proc. IEEE 17th Int. Conf. Industrial Informatics (IEEE INDIN), Helsinki, Finland, pp. 1351-1357, July 2019.

[25] H. Shen, G. Liu, and L. Ward, "A Proximity-Aware Interest-Clustered P2P File Sharing System," *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1509-1523, 1 June 2015, doi: 10.1109/TPDS.2014. 2327033.

[26] K. Shuang, P Zhang, and S. Su, "Comb: A Resilient and Efficient Two-Hop Lookup Service for Distributed Communication System," Security and Communication Networks, 8(10):1890-1903, 2015.

[27] Stocia, R. Morris, D. Liben-Nowell, D. R. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Tran. Networking*, 11(1):17-32, Feb. 2003.

[28] Z. Tu, W. Jiang and J. Jia, "Hierarchical Hybrid DVE-P2P Networking Based on Interests Clustering," 2017 International Conference on Virtual Reality and Visualization (ICVRV), Zhengzhou, China, pp. 378-381, 2017, doi: 10.1109/ICVRV.2017.00087.

[29] M. Xu, S. Zhou, and J. Guan, "A New and Effective Hierarchical Overlay Structure for Peer-to-Peer Networks," *Computer Communications*, 34:862-874, 2011.

[30] M. Yang and Y. Yang, "An Efficient Hybrid Peer-to-Peer System for Distributed Data Sharing," *IEEE Trans. Computers*, 59(9):1158-1171, Sep. 2010.

Indranil Roy is an Assistant Professor in the Department of Computer Science at the Southeast Missouri State University. He received his MS and Ph.D. degrees in Computer Science from Southern Illinois University, Carbondale in 2018 and 2022, respectively. His current research interest includes the design of architecture and communication protocols for structured peer-to-peer overlay networks, security in overlay networks, and Blockchain.

Nick Rahimi (photo not available) is the Director of Cyber Innovation Lab and an Assistant Professor at the School of Computing Sciences & Computer Engineering of the University of Southern Mississippi (USM). Dr. Rahimi obtained two

Bachelor of Science degrees in Computer Software Engineering and Information Systems Technologies with a concentration in Cybersecurity and received his Master and Ph.D. degrees in Computer Science from Southern Illinois University (SIU). His research interest lies in the area of cybersecurity, blockchain, cryptography, internet anti-censorship, machine learning in cybersecurity, distributed systems, and decentralized networks. Prior to joining USM, Dr. Rahimi was a tenure track Assistant Professor at SIU for 2 years and Southeast Missouri State University (SEMO) for one year respectively. Moreover, he has over eight years of experience in industry as a software engineer and team leader.

Ziping Liu received her PhD in Engineering Science from Southern Illinois University at Carbondale in 1999 and began her computing career at Motorola, where she developed software for mobile phones. Currently, she is a Professor of Computer Science at Southeast Missouri State University, where she has been teaching since 2001. Her research interests encompass a wide range of topics, including machine learning, cloud computing, secured software design, wireless ad hoc networks and sensor networks, distributed computing and game development.

Bidyut Gupta (photo not available) is currently a Professor of Computer Science at the School of Computing, Southern Illinois University at Carbondale. His research interests include fault tolerant distributed computing, design of P2P network architectures with low latency communication protocols, fog computing and its applications, and high latency networks. He is a Senior member of IEEE and ISCA.

Narayan C. Debnath (photo not available) is currently the Founding Dean of the School of Computing and Information Technology at Eastern International University, Vietnam. He is also serving as the Head of the Department of Software Engineering at Eastern International University, Vietnam. Formerly, Dr. Debnath served as a Full Professor of Computer Science at Winona State University, Minnesota, USA for 28 years, and the elected Chairperson of the Computer Science Department at Winona State University for 7 years. Dr. Debnath has been the Director of the International Society for Computers and their Applications (ISCA), USA since 2014.

Professor Debnath made significant contributions in teaching, research, and services across the academic and professional communities. He has made original research contributions in software engineering, artificial intelligence and applications, and information science, technology and engineering. He is an author or co-author of over 500 research paper publications in numerous refereed journals and conference proceedings in Computer Science, Information Science, Information Technology, System Sciences, Mathematics, and Electrical Engineering. He is also an author of over 15 books published by well-known international publishers including Elsevier, CRC, Wiley, Bentham Science, River Publishing, and Springer.

Dr. Debnath has made numerous teaching, research and invited keynote presentations at various international conferences, industries, and teaching and research institutions in Africa, Asia, Australia, Europe, North America, and South America. He has been a visiting professor at universities in Argentina, China, India, Sudan, and Taiwan. He has been maintaining an active research and professional collaborations with many universities, faculty, scholars, professionals and practitioners across the globe. Dr. Debnath is an active member of the IEEE, IEEE Computer Society, and a Senior Member of the International Society for Computers and their Applications (ISCA), USA.

Application of Artificial Intelligence to Predict Permeability in Low Permeability Limestone Formation

Ahmed Jubair Mahmood*
Al-Farabi University College, Baghdad, IRAQ

Mohammed Ahmed Jubair[†] and Thulfiqar H. Mandeel[†]
Imam Ja'afar Al-Sadiq University, 66002, Al-Muthanna, Iraq

Abstract

Porosity (ϕ) and permeability (k) are two important rock properties used in oil, gas, and water resources calculations. The mathematical relationship between porosity and permeability is not easily demonstrated. Due to the heterogeneity of rock properties, there is no exact mathematical formula from which permeability could be calculated depending on porosity. Some researchers depend on the core analysis to introduce a general mathematical relation between ϕ and k, while others depend on the flow zone indicator (FZI) method to find such a relation. Recently, supervised machine learning has gained much popularity in establishing a relationship between complex non-linear datasets. This type of machine learning algorithm has shown its superiority over petroleum engineering regression techniques in terms of prediction errors for high dimensional data, computational power, and memory. In this work, the FZI method was applied to a data set for Khasib formation in the East Baghdad oil field to present a mathematical formula relating ϕ and k. In addition an AI algorithm was used to predict k depending on ϕ for the formation under study. Results proved that the predicted values of k had better agreement with the actual k values compared to the k values calculated using the FZI method. The accuracy of results is measured by calculating the coefficient of determination (R^2).

Key Words: Porosity, permeability, FZI, ANN, R^2.

1 Introduction

A petroleum reservoir is a heterogeneous geological system with large intrinsic complexity. Porosity and permeability are two important rock properties used in oil and gas reservoir and water resource calculations. The capability of a rock to hold fluids depends on its porosity while permeability controls the ability of fluids to flow into the porous rock. There is a relationship between porosity and permeability, but this relation is not easily demonstrated especially in carbonate rocks compared with clastic rocks [12, 15]. Sometimes there is a positive relation between them while on the other hand, there is low permeability with high porosity and vice versa. If the quality of the carbonate reservoir is merely evaluated by porosity, the results could be quite inconsistent with the actual production preference [11, 18-19].

For decades a considerable number of researches have been done trying to establish a mathematical relation between porosity and permeability. The traditional core analysis, well logs, and rock petrophysical properties such as pore geometry, capillary pressure, surface area, water saturation, and production data were used to find this relation. The Kozeny, Kozeny–Carmen (K–C) correlation and their modifications are the most widely accepted methodology in the oil industry. [6-7, 17, 22].

Amaefule et al. [3] presented a modification for K-C correlation by introducing the concept of the Reservoir Quality Index (RQI) and Flow Zone Indicator (FZI) to enhance their capability to capture the various reservoir flow behavior based on their respective characteristics. Yet, there are challenges in using the original correlation due to its inherent limitations and oversimplified assumptions that prevent accurate Hydraulic Flow Unit (HFU) definitions. [2-3, 8, 16]. Recently several researchers utilized different artificial intelligence methods to get a more accurate estimation of permeability in carbonate reservoirs such as fuzzy logic, genetic algorithm, PSO, and Artificial Neural Networks (ANNs) [9, 14, 23]. In this work, the FZI method and ANN were used for permeability estimation depending on porosity. The data was collected from the results of core analysis for one of the carbonate reservoirs in the east Baghdad oil field [20]. The accuracy of results is measured by calculating the coefficient of determination (R^2). Results demonstrate that the artificial intelligence algorithm predicts permeability more accurately than the FZI method.

2 Brief Background of Supervised Machine Learning Algorithms

The implementation of supervised machine learning methods to solve complicated problems has gained momentum in many

* Department of Petroleum Engineering. Email: dr.ahmed.jabir@alfarabiuc.edu.iq.

[†] Department of Computer Technical Engineering, College of Information Technology. Email: mohammed.a@sadiq.edu.iq, thulfiqar.hussein@sadiq.edu.iq

industries including the petroleum industry. Most of these complex problems were impeding critical decision-making and enhanced advancement in the industry hence, researchers progressively moved from using empirical correlations and linear regression models to the application of AI techniques which have been welcomed due to their added value in the industry (Ali, [1]). The first pattern recognition algorithm was proposed by Fisher in the mid-1930s where two normal distribution populations were modeled indicating that the Bayesian solution is a quadratic decision function [10];

$$\mathrm{F}_{sq}(X) = sign\left[\frac{1}{2}(x-m_1)^T\sum_{1}^{-1}(x-m_1) - \frac{1}{2}(x-m_1)^T\sum_{2}^{-1}(x-m_2) + \ln\frac{|\Sigma_2\quad|}{|\Sigma_1\quad|}\right] \qquad (1)$$

This formed the basis of many supervised machine learning algorithms in use today. Supervised machine learning algorithms can generally be put as machines, being trained to map input data (x) to output data (y) by learning from a set of target function (f) mathematically put as;

$$y = f(x) \qquad (2)$$

The dataset is usually divided into a 70:30 or 80:20 ratio for training and testing. Predictive modeling or analytics is hence described as a supervised machine learning algorithm that learns certain hidden and complex features from the target function (f) that are otherwise invisible or complex to statistical methods to make predictions of the output data (y) during training and testing. The model is then applied to new and unseen data (X) to validate its prediction accuracy, efficiency, and errors. The main reason for further research in supervised machine learning is to improve prediction accuracy, minimize errors and enhance computational efficiency.

3 Artificial Neural Network (ANN)

3.1 Concept and Theoretical Framework

Artificial Neural Networks (ANNs), a biologically inspired intelligence model, gained major attention in the '80s when the neuroscience industry clocked some important advancements in its use leading to high interest in understanding the importance of NN models [21]. The brain's neuron functions are replicated by large sets of algorithms representing the ANNs which are capable of establishing relationships amongst highly anomalous nonlinear variables and producing sophisticated, accurate, and reliable results to complex problems through learning and training [1].

There are generally two types of neural networks with the most rudimentary and straightforward ANN paradigm being the Feed-Forward Neural Network (FFNN) which is a multilayer interconnection of perceptron where the output layer does not form a loop for feedback connections or recurrent networks but in a forward unidirectional flow [24]. A simplified neuron network model can be represented mathematically as;

$$h_\theta(x) = \frac{1}{1+e^{-\theta x^T}} \qquad (3)$$

Where $h_\theta(x)$ is referred to as the output, x is the input but x and θ are the parameter vectors. A typical FFNN architecture is shown in Figure 1 below. The other type of ANN is the Feedback Neural Network (FBNN) popularly called the Back-Propagation ANN (BPANN) which is widely used in supervised learning. This type of neural network is of a similar architectural structure to the FFNN but allows the creation of a loop where erroneous information is sent back for the iterative altering of weight values until error can no longer improve to achieve a more accurate output variable. A typical BPNN archetypal structure is shown in Figure 2.

The ANN works like the neuron connections in the brain with multiple interconnections where each node (point) is linked to the other in the form of a pathway for interaction with each other. The ANN can work with a single hidden layer to assign weights to each node in a neural structure [13]. The training phase feeds the input data as vectors through a NN framework. The output error is computed and looped back, for a BPANN, into the network for the iterative altering of the weights using gradient descent to be done to reduce the error based on experience until it can no longer be improved. This process is repeated until a bias value that gives a more accurate prediction is obtained. The mathematical equation of the error function derivative used to update the weights by gradient descent is represented as [4].

$$\Delta\mathrm{w(t)} = \ \omega\nabla\mathrm{E(T)} + \alpha\Delta\mathrm{e(t-1)} \qquad (4)$$

Where Δw is the weight update, E is the error observed between the predicted and actual output, ω is the learning parameter, and α is the momentum parameter (<1). With each increase in hidden layers depending on the complexity of the problem being worked on, we will be entering into the realms of deep learning [23].

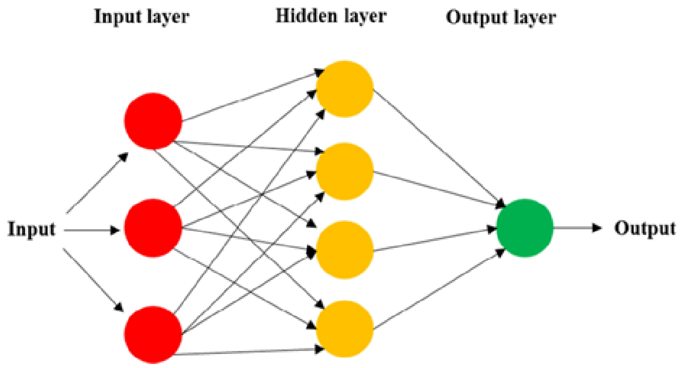


Figure 1: A typical feed-forward neural network architecture (Saggaf et al., 2003)

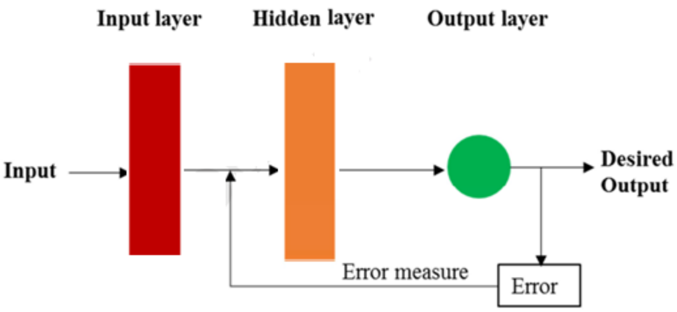


Figure 2: A back-propagation neural network architecture (Saggaf et al., 2003)

Within the scope of reservoir characterization, NNs are commonly used as a highly effective supervised machine-learning technique for classification problems. This is mainly attributed to the unique ability of a neural network to mimic the human way of thinking [5] to solve classification problems by creating complex dynamic estimation functions that offer improved performances over other algorithms. Given adequate computational power, ANN can theoretically learn the shape of any function necessary for classification. Regression analysis helps to model the relationship that exists between a dependent and one or several independent variables showing significant relations between them and the change of the dependent value as a result of a change in the independent variables.

4 The FZI Method

Kozeny [17]: concluded one of the important formulas to estimate permeability

$$K = a\,(\emptyset/S) \qquad (5)$$

Where, s is the surface area per unit bulk volume (L^2/L^3), Carmen [6]: changed the Kozeny formula and introduced permeability in packs of a uniform size.

$$K = \left(\frac{1}{f_g \tau \ S^2}\right)\left(\frac{\emptyset^3}{(1-\emptyset)^2}\right) \qquad (6)$$

Where f_g is the shape factor, dimensionless and τ is the Tortuosity (dimensionless).

Amaefule et al., [3]: suggested two known methods to estimate permeability and indicate hydraulic units for uncored wells, first method is reservoir quality index (RQI) and the second is flow zone indicator (FZI), where the hydraulic unit will be introduced as a unit of reservoir rock which given a special relationship between porosity and permeability. A lot of mathematical resolutions are applied to equation (2) to become as follows:

$$0.0314\sqrt{\frac{k}{\emptyset}} = \left[\frac{1}{s\sqrt{f g\,\tau}}\right]\left[\frac{\emptyset}{1-\emptyset}\right] \qquad (7)$$

Surface area, tortuosity, and shape factor could be measured differently in the reservoir so that term of the Kozeny-Carmen formula $\frac{1}{s\sqrt{f g\,\tau}}$ is assumed by the square root of FZI^2.

RQI can be introduced as the following term:

$$RQI = 0.0314\sqrt{\frac{k}{\emptyset}} \qquad (8)$$

And $\emptyset_z$ can be normalized as follows:

$$\emptyset_z = \emptyset/(1-\emptyset) \qquad (9)$$

So FZI will be:

$$\text{FZI} = \text{RQI} / \emptyset_z \qquad (10)$$

Then RQI vs. $\emptyset_z$ can be plotted on (log–log) paper, where similar FZI values of the core sample will appear as a straight line, while various FZI values of the core sample show on other parallel straight lines [3].

4.1 Coefficient of determination (R^2)

The coefficient of determination, or R^2, is a measure that provides information about the goodness of fit of a model. In the context of regression, it is a statistical measure of how well the regression line approximates the actual data. It is therefore important when a statistical model is used either to predict future outcomes or in the testing of hypotheses.

$$R^2 = 1 - \frac{sum\ squared\ regression\ (SSR)}{total\ sum\ of\ squares\ (SST)} \qquad (11)$$

5 Data Set

The data set was for the Khasib formation, which is one of the reservoir rocks in the East Baghdad (EB) oil field in Iraq. It is a low permeability porous limestone from the upper cretaceous age with shelly lime and chalky lime in some sections. The data are from core analysis for the cored intervals in wells EB- 4, 11, 12 and 16 [20]. The porosity range is (6 to 29.24%) and the permeability range is (0.1 to 28.9) md. Figure 3 shows porosity – permeability plot for the data.

6 Methodology

Two methods were utilized to predict core permeability depending on its porosity. The first is the FZI method in which FZIs were calculated from equation 6 for each core data and rounded to the nearest integer (0, 1, 2, etc.). The similar FZI value of the core sample will have appeared as a straight line on a log-log plot of RQI vs $\emptyset_z$, while various FZI values of the core sample show on other parallel straight lines. From each straight-line equation, a relation between porosity and permeability was deduced, then used to calculate permeability for the set

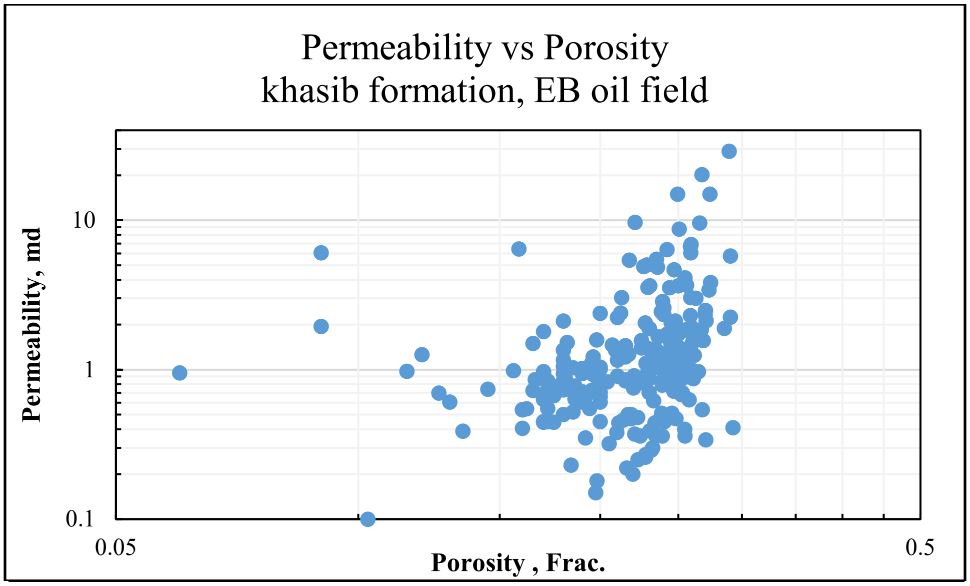


Figure 3: Porosity permeability plot

having the same rounded FZI value. The second method is using one artificial intelligence algorithm. Usually in these algorithms data set is split up into two groups.

One of these groups is used as input training data for the algorithm, while the second one is for comparison between the predicted and the original values. More than one algorithm had been adopted to perform permeability prediction depending on the location, depth, and porosity of the data used. Part of the data is fed for training and the remaining data is compared with that predicted by the algorithm. Finally, the algorithm that has the best regression values were selected to perform the work.

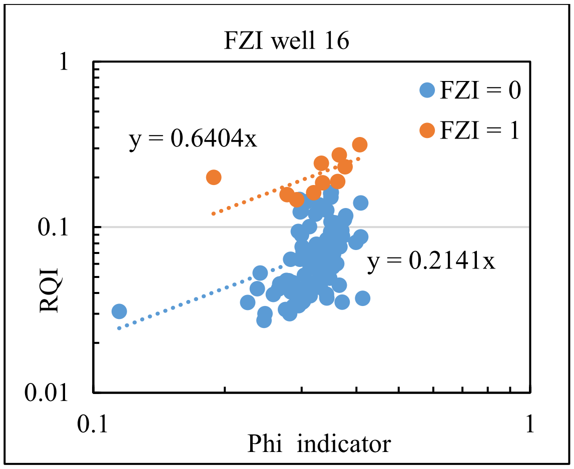
7 Results and Discussion

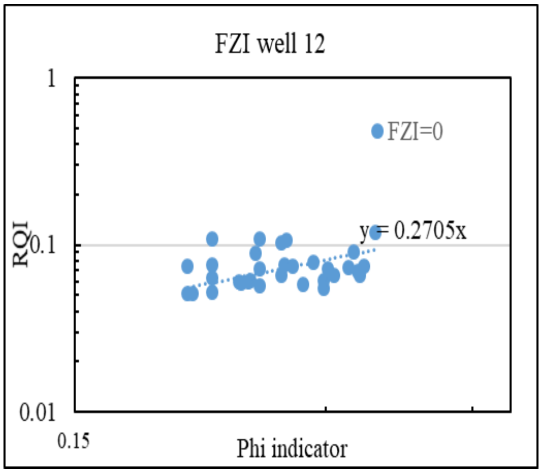
Applying the FZI method to the data of the four wells under study shows that most zones identified with FZI = 0, with some points having an FZI value of 1 which is attributed to low permeability values. The FZI plots for the four wells are shown in Figure 4.

The equations relating porosity with permeability resulted from the FZI relations used to calculate the core permeability. Application of the ANN algorithm resulted in predicted permeability values for each corresponding porosity. To get a better comparison, plots of measured core permeability against calculated permeability using the FZI method and plotted against predicted permeability by ANN for each well are given in Figures 5 and 6.

The coefficients of determination (R^2) in Figures 5 and 6 were better for permeability predicted by the ANN methods in comparison with that calculated using the FZI method as shown in Table 1.

The plot of measured core permeability, calculated permeability using the FZI method, and that predicted by the ANN algorithm versus depth for the four well in Figures 7 and





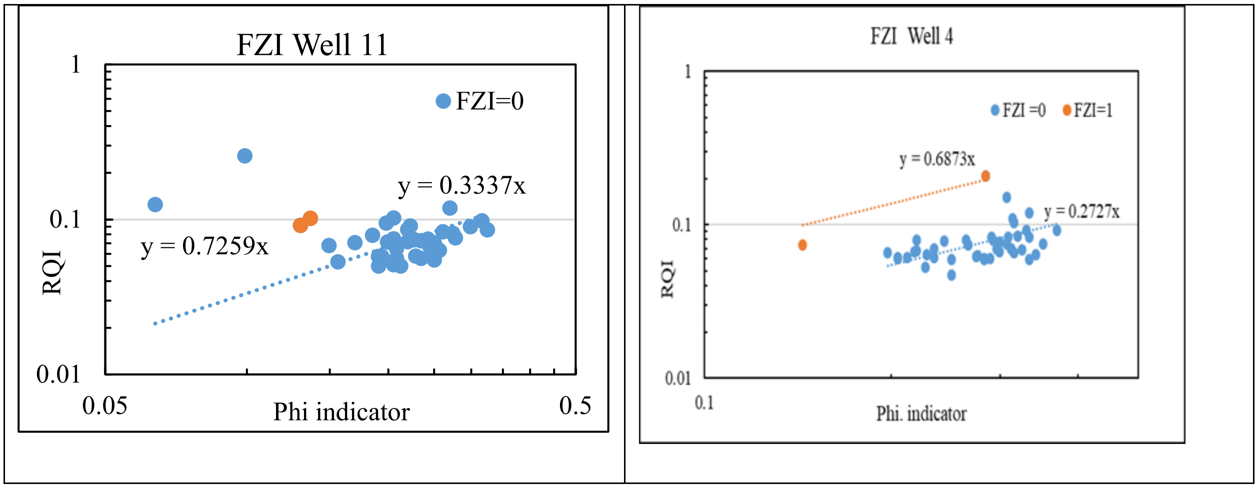


Figure 4: The FZI plot for wells EB- 16, 12, 11, and 4

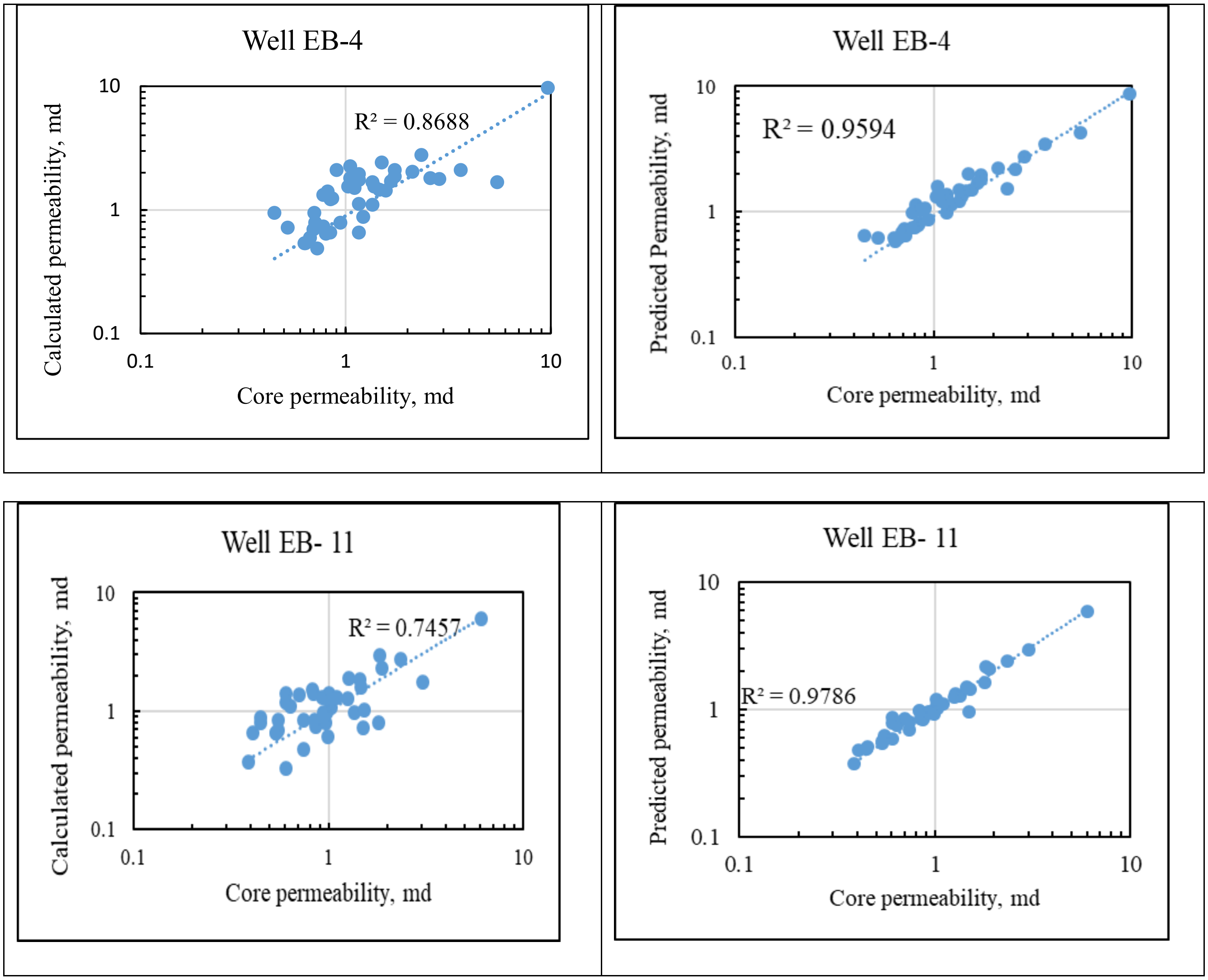


Figure 5: Calculated and predicted permeability vs core permeability wells EB-4 and 11

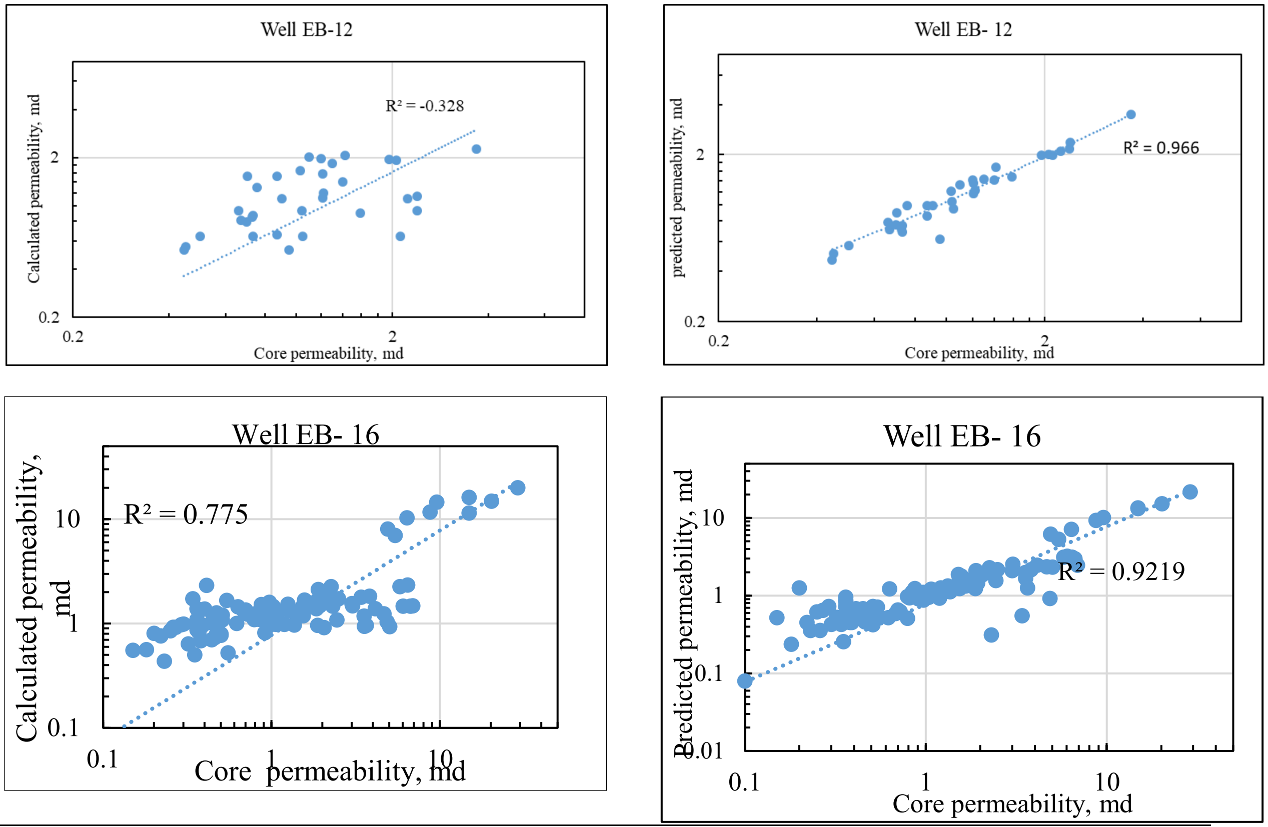


Figure 6: Calculated and predicted permeability vs core permeability wells EB-12 and 16

Table 1: Comparison between the R^2 values.

Well No.	Value of R^2	
	The FZI method	Predicted by ANN
12	0.328	0.966
16	0.775	0.9216
4	0.6953	0.9594
11	0.7457	0.9786

8 indicate that there is better agreement between the predicted and the measured values compared with the calculated ones.

Finally, Figure 9 presents a plot of the measured permeability for the four wells vs the calculated permeability shows that R^2 = 0.7632 while the plot vs the predicted permeability shows that R^2 = 0.9188.

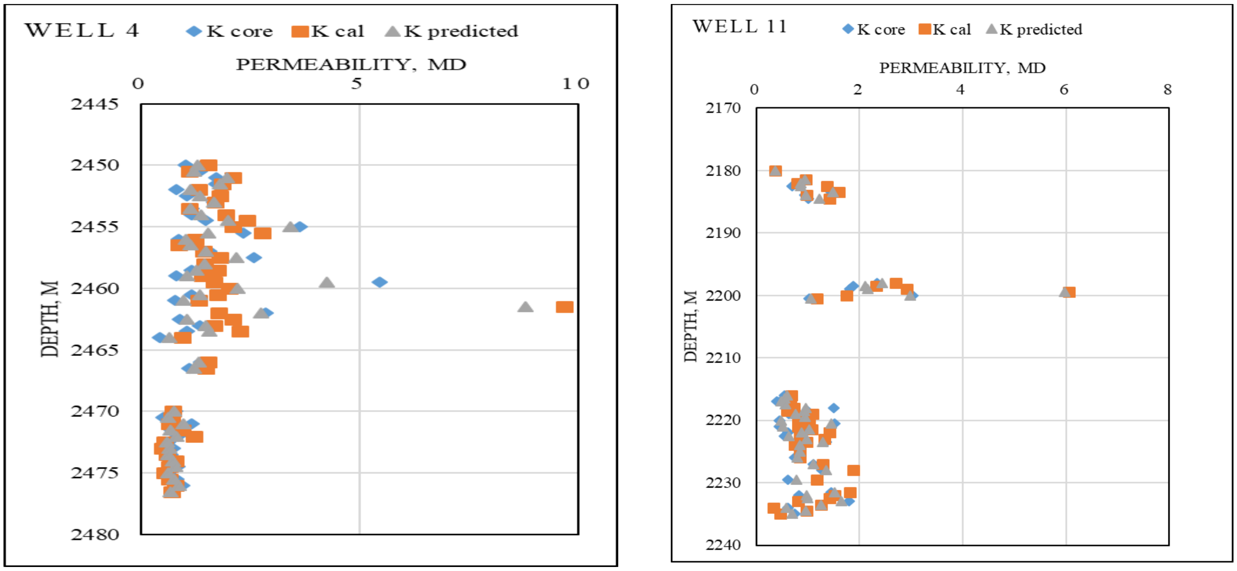


Figure 7: Permeability vs depth for wells EB-4 and EB-11

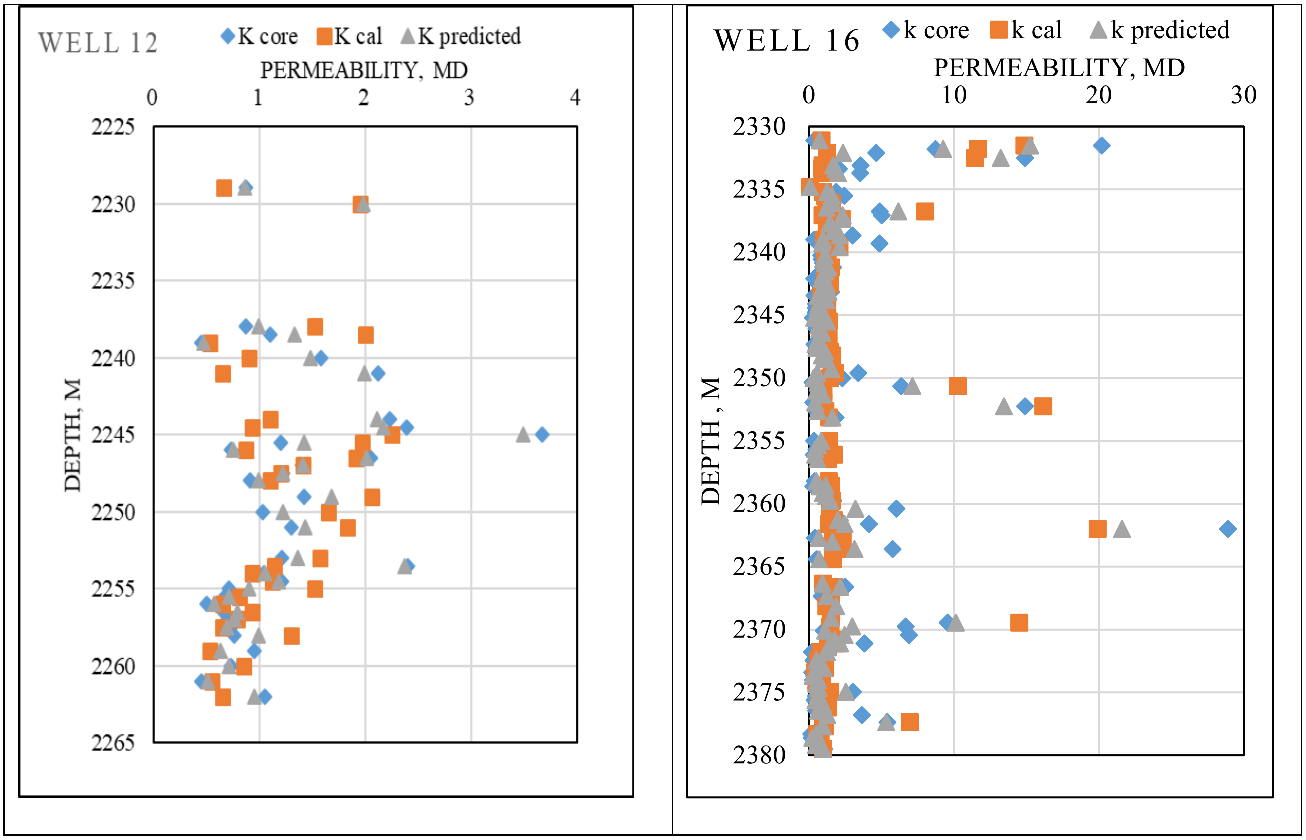


Figure 8: Permeability vs depth for wells EB-12 and EB-16

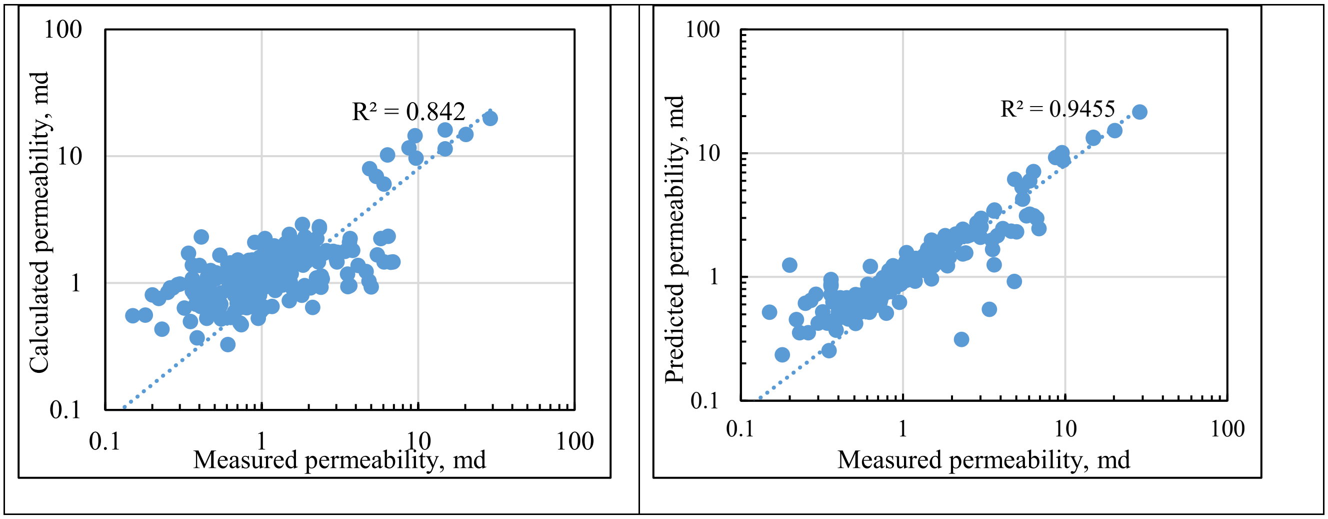


Figure 9: Calculated and predicted permeability vs measured permeability of the four wells

8 Conclusion

AI is being used more in exploration, development, production, reservoir engineering, and management planning to speed up decision-making, reduce cost, and save time. Supervised machine learning is popular for connecting complex non-linear datasets. This approach outperforms petroleum engineering approaches in prediction errors, computational power, and memory. Two significant rock qualities, porosity, and permeability are employed in estimations of oil, gas, and water resources. It is difficult to show the mathematical connection between permeability and porosity. There is no precise mathematical formula from which permeability may be determined relying on porosity due to the variety of rock qualities. This research utilizes one of the most popular artificial algorithms which is named Genetic Algorithm. The algorithm is used to predict and calculate the value of the porosity and permeability of the rock. The results revealed that the ANN approach is better than the mathematical approach. In finding the final solution based on all possible solutions that is produced in the search space, it consumes a high amount of time to find the optimal solution.

References

[1] J. K. Ali, “Neural Networks: A New Tool for the Petroleum Industry?” In European Petroleum Computer Conference, OnePetro, March 1994.

[2] Mohammed S. Al-Jawad and Israa J. Ahmed, “Permeability Estimation by Using the Modified and Conventional FZI Methods,” *Journal of Engineering*, 24(3):59-67, March 2018.

[3] J. O. Amaefule, M. Altunbay, D. Tiab, D. G. Kersey, and D. K. Keelan, “Enhanced Reservoir Description: Using Core and Log Data to Identify Hydraulic (Flow) Units and Predict Permeability in Uncored Intervals/Wells,” In SPE Annual Technical Conference and Exhibition. OnePetro, October 1993.

[4] P. Avseth and T. Mukerji, “Seismic Lithofacies Classification from Well Logs using Statistical Rock Physics,” Petrophysics-*The SPWLA Journal of Formation Evaluation and Reservoir Description,* 43(02):70-81, 2002.

[5] S. Bhattacharya, P. K. Ghahfarokhi, T. R. Carr, and S. Pantaleone, “Application of Predictive Data Analytics to Model Daily Hydrocarbon Production using Petrophysical, Geomechanical, Fiber-Optic, Completions, and Surface Data: A Case Study from the Marcellus Shale, North America,” *Journal of Petroleum Science and Engineering,* 176:702-715, 2019.

[6] P. C. Carman, “Fluid Flow through Granular Beds,” Trans. Inst. Chem. Eng. 15:150-166, 1937. https://doi.org/10.1016/S0263-8762(97)80003-2.

[7] P. C. Carman, “Fundamental Principles of Industrial Filtration,” Trans. Inst. Chem. Eng. 16:168-188, 1938.

[8] D. K. Davies and R. K. Vessell, “Identification and Distribution of Hydraulic Flow Units in a Heterogeneous Carbonate Reservoir: North Robertson Unit, West Texas,” In Permian Basin Oil and Gas Recovery Conference. OnePetro, March 1996.

[9] Mohamed Elmorsy, Wael El-Dakhakhni, and Benzhong Zhao, “Generalizable Permeability Prediction of Digital Porous Media via a Novel Multi-Scale 3D Convolutional Neural Network,” Water Resources Research, 10.1029/2021WR031454.

[10] R. A. Fisher, “The Use of Multiple Measurements in Taxonomic Problems,” Annals of Eugenics, 7(2):179-188, 1936.

[11] W. Guanghui, L. Honghui, L. Zhang, W. Chenglin, and Z. Bo, “Reservoir-Forming Conditions of Ordovician Weathering Crust in the Maigaiti Slope, Tarim Basin, NW China,” Petroleum Exploration and Development, 39(2):155-164, 2012.

[12] W. Guoqi, S. Ping, J. ZHANG, J. I. A. O. Guihao, X. I. E. Wuren, and X. I. E. Zengye, “Formation Conditions and Exploration Prospects of Sinian Large Gas Fields, Sichuan Basin,” Petroleum Exploration and Development, 40(2):139-149, 2013.

[13] K. Hornik, M. Stinchcombe, and H. White, “Multilayer Feedforward Networks are Universal Approximators,” Neural Networks, 2(5):359-366, 1989.

[14] Zehui Huang, John Shimeld, Mark Williamson, and John Katsubet, “Permeability Prediction with Artificial Neural Network Modeling in the Venture Gas Field, Offshore Eastern Canada,” GEOPHYSICS, 61(2):422-436, March-April 1996.

[15] W. Jing, L. Huiqing, X. Jie, and H. Zhang, “Formation Mechanism and Distribution Law of Remaining Oil in Fracture-Cavity Reservoir,” Petroleum Exploration and Development, 39(5):624-629, 2012.

[16] Hayder A. Jumaah, Sameer N. Al Jawad, and Wafa Al Kattan, “Predicting Permeability by Integrating FZI Approach with Rock Typing in Heterogeneous Carbonate Reservoir,” AIP Conference Proceedings, 2443:030003, 2022.

[17] J. Kozeny, “Uber kapillare Leitung des Wassers im Boden-Aufstieg, Versickerung und Anwendung auf die Bewasserung, Sitzungsberichte der Akademie der Wissenschaften Wien,” Mathematisch Naturwissenschaftliche Abteilung, 136:271-306, 1927.

[18] H. Ling, Z. Lun, L. Jianxin, M. A. Ji, L. U. I. Ruilin, W. A. N. G. Shuqin, and Z. H. A. O. Wenqi, “Complex Relationship between Porosity and Permeability of Carbonate Reservoirs and Its Controlling Factors: A Case Study of Platform Facies in Pre-Caspian Basin,” Petroleum Exploration and Development, 41(2):225-234, 2014.

[19] Z. Lun, C. Yefei, N. Zhengfu, W. U. Xuelin, L. I. U. Lifang, and C. H. E. N. Xi, “Stress Sensitive Experiments for Abnormal Overpressure Carbonate Reservoirs: A Case from the Kenkiyak Fractured-Porous Oil Field in the Littoral Caspian Basin,” Petroleum Exploration and Development, 40(2):208-215, 2013.

[20] Ministry of Oil, Routine Core Analysis (RCA) for Wells (EB- 4,11,12,16 and 35), 1981-1983.

[21] S. Mohaghegh, R. Arefi, I. Bilgesu, S. Ameri, and D. Rose, “Design and Development of an Artificial Neural Network for Estimation of Formation Permeability,” SPE Computer Applications, 7(06):151-154, 1995.

[22] Shun Nomura, Yuzuru Yamamoto, and Hide Sakaguchi. “Modified Expression of Kozeny–Carman Equation Based on Semilog–Sigmoid Function,” Soils and Foundations 58:1350-1357, 2018.

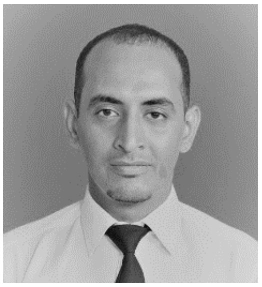
[23] A. Rostami, A. Kordavani, S. Parchekhari, A. Hemmati-Sarapardeh, and A. Helalizadeh, “New Insights into Permeability Determination by Coupling Stoneley Wave

Propagation and Conventional Petrophysical Logs in Carbonate Oil Reservoirs," Scientific Reports, 12(1):11618, 2022.

[24] P. Saikia, R. D. Baruah, S. K. Singh, and P. K. Chaudhuri, "Artificial Neural Networks in the Domain of Reservoir Characterization: A Review from Shallow to Deep Models," Computers and Geosciences, 135:104357, 2020.

AHMED JUBAIR MAHMOOD Received the B.Sc. degree in Petroleum and Mining Engineering, College of Engineering–Baghdad University, 1980, the MSc. Degree in Petroleum Engineering College of Eng.– Baghdad University, 1985 and Ph.D. Petroleum Eng.– College of Eng.– Baghdad University 2007. Currently, he is working as a senior lecturer at the Department of Petroleum Engineering, Al-Farabi university college, Baghdad, Iraq. He can be contacted at email dr.ahmed.jabir@alfarabiuc.edu.iq.

MOHAMMED AHMED JUBAIR was born in Iraq. He received a B.Sc. degree in Computer Engineering from Al-Marrif University, Iraq, in 2012. After completing the B.Sc. degree, he worked as a teaching assistant at Al-Marrif University from 2012-2014. In 2017, He received an M.Sc. degree from the Computer Science Networks Department, Universiti Kebangsaan Malaysia (UKM), Malaysia. He received a Ph.D. degree from the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia (UTHM), Malaysia in 2021. Currently, he is working as a senior lecturer at the Department of Computer Technical Engineering, College of Information Technology, Imam Ja'afar Al-Sadiq University in Iraq. His research interests are in the area of Software Engineering, Artificial Intelligence, Communication Systems, Networking, and Machine Learning. He can be contacted at email mohammed.a@sadiq.edu.iq.



THULFIQAR H. MANDEEL was born in Iraq in 1991, He received an M.Sc. degree in Embedded System Design Engineering and a Ph.D. degree in Computer Engineering from Universiti Malaysia Perlis (UniMAP) in Perlis, Malaysia in 2015 and 2019, respectively. He joined UniMAP in 2016 as a graduate assistant and is currently a lecturer at Imam Ja'afar Al-Sadiq University in Iraq. His current research interests include image processing, biometric recognition, medical image classification, and deep learning. He can be contacted at email: Thulfiqar.hussein@sadiq.edu.iq.

A Review of Image Steganography Tools

Anal Kumar[*] and Hermann Jamnadas[*]
Fiji National University, Nadi, Fiji

Vishal Sharma[†]
Fiji National University, Nasinu, Fiji

S M Muyeen[‡],
Qatar University, Doha 2713, Qatar

A. B. M Shawkat Ali[§]
University of Fiji, Lautoka, Fiji

Abstract

Steganography is viewed like cryptography in one sense and both of them aim to guarantee the privacy of data. With the increasing application of steganography in the digital world, many issues must be understood in computer forensic inspection. There is a wide variety of tools and techniques, with their own focus and weaknesses. Stable changes must be made and more recent adaptations have been made. Initially, Steganography overview provided computer forensics experts in this field with knowledge and understanding of steganography. Consequently, the strength of most steganography tools depends on how well the software shrouds data or hides identification. As such it is basic to audit and comprehend which image-based steganography tool might be reasonably best and what are a portion of the basic pointers to recognizing concealed information in records. This research has been carried out through downloading and testing different image based steganography tools and assessing many articles and conference procedures. Various image steganography tools that were randomly selected and the list is not intended to be an exhaustive one consisting of all possible image steganography tools. Additionally, there is a filter that was applied to separate all steganography tools that were designed in Java and were windows based with GUI interface. These filtered Java and Windows based steganography tools were tested to see the encode and decode time difference, the visual differences and file size variance. It tends to be plainly seen that many picture steganography tools have blemishes that take into consideration the simple recognition or sign of a concealed document in a stego-picture. As such it is basic that individuals engaged with data security use those picture steganography tools that will diminish the opportunity of the message being identified since the strength of steganography is not to make the message ambiguous but instead to cause it to appear as though it was never there in any case. The authors posit that this research will be valuable for those keen on utilizing or testing image-based steganography tools.

Key Words: Image steganography, stego-picture, steganalysis, steganography tools

1 Introduction

Security of information has been a core concern for most of humanity. Many relied on elements of cryptography to encrypt secret messages and communiques to prevent malicious parties from taking advantage of successful interceptions of such messages [19]. However, cryptography does not hide the communication from other parties making it possible for others to attempt to alter the contents of the encrypted message [7]. Hence, steganography is useful as it allows for such communication to be hidden from others by embedding a secret message inside a multimedia file such as an image file, audio file, or a video file [20]. Images are a suitable medium to act as "carriers" since they are one of the most prevalent types of media and can store substantial amount of data without affecting image quality [1]. Image steganography tools takes a secret message or file and embeds it in a cover image to create a stego-imag [1, 18]. The strength of the image steganography lies in the fact that nobody is aware or is suspicious of the existence of hidden data in a nondescript image file. Hence a steganography system or tool will become useless if the "carriers" are suspected of harboring a secret message [4, 18]. It is essential therefore to use only those image steganography tools that will yield the least amount of suspicion.

This paper will thus begin with a brief overview of the steganography tools to be reviewed, the tests to perform on

[*] Department of Computer Science and Information Systems. Email: anal.kumar@fnu.ac.fj

[†] Department of Computer Science and Information Systems.

[‡] Department of Electrical Engineering.

[§] Department of Computer Science and Mathematics.

such tools and the analysis of the results that will help determine which of the image steganography tools will yield stego-images that are nondescript in nature.

2 Image Steganography Tools Overview

2.1 Hide 'N' Send

Hide'N'Send is a clever little application that can be utilized to cover text records containing private messages, data and passwords by concealing them inside JPEG pictures, which you can share through email or some other medium. It's additionally important that the application bolsters a lot of camouflage, hash and encryption calculations. Alongside concealing the documents, the device can be utilized to remove the inserted records also and to finish everything off, it requires a secret phrase (that you characterize during the document concealing cycle).

2.2 Invisible Secrets 4

Invisible Secrets is a application that permits you to scramble your private documents by methods for various devices. With this program, users have the option to shield your information and messages from inquisitive eyes. The application incorporates a component that permits you to scramble and shroud documents into different record types like photographs or sound records. There's additionally another device that empowers you to secure your documents by changing them into an incoherent configuration that must be perused with the right secret key. This secret phrase is profoundly encoded utilizing solid encryption calculations.

2.3 JHide

A command line tool written in Java to shroud document records in pictures, JHide is a little programming application whose design is to help you conceal delicate documents inside custom pictures. The apparatus can be conveyed on all Windows forms out there, given that you have the Java working stage introduced on the host PC. The photos that contain private documents look equivalent to other photographs put away in your PC so they won't raise any doubts to different clients. They can be opened with similar devoted watchers, sent through email, or printed.

2.4 Open Puff

OpenPuff is a "proficient steganography tool" that permits clients to hide records in picture, sound, video, or Flash documents. It gives a wide exhibit of highlights to shield concealed information from revelation. OpenPuff is an expert steganography device with extraordinary highlights, appropriate for exceptionally delicate information undercover transmission. In OpenPuff information is part of numerous transporters. Just the right transporter succession empowers viewing. In addition, up to 256Mb can be covered up, on the off chance that you have enough transporters at removal. The last transporter will be dispatched with arbitrary pieces to make it undistinguishable from others.

2.5 OpenStego

OpenStego gives two primary functionalities: Data Hiding: It can shroud any information inside a cover document (for example pictures) and Watermarking with an undetectable mark. It tends to be utilized to recognize unapproved document duplicating. Utilizing OpenStego is quite direct. There are two methods of activity - information covering up and watermarking.

2.6 StegHide

Steghide is a steganography program that can conceal information in different sorts of picture and sound documents. The tone respectively test frequencies are not changed subsequently making the installing safe against first-request factual tests. Steghide highlights include compression of installed information, encryption of installed information, inserting of a checksum to confirm the respectability of the extracted information and backing for JPEG, BMP, WAV and AU records.

2.7 StegoShare

Stegoshare can be effortlessly utilized for unknown record sharing. An uploader downloads legitimate pictures from a public photograph facilitating webpage, and inserts the edited document into those pictures. The uploader then transfers pictures to the public photograph deluge tracker and puts the connections referring to the stego pictures with blue-penciled document's portrayal on a discussion or blog. Downloaders, seeders, and public photograph trackers, whenever found disseminating unlawful records, are shielded from lawful indictment, since they can generally utilize conceivable deniability, saying that they don't knew the slightest bit about the illegal document in the pictures.

2.8 S-Tool

S-Tools (Steganography Tools) is a program composed by Andy Brown. It is maybe the most generally perceived steganography tool accessible today. BMP, GIF, and WAV documents can be utilized as the cover records that disguise the mystery messages. It is not difficult to use, with basic relocating of the records. S-Tools will conceal the mystery message inside the cover document through arbitrary accessible pieces. These accessible pieces are resolved using a pseudorandom number generator. This nonlinear inclusion makes the presence and extraction of mystery messages more troublesome.

2.9 VSL Virtual Steganographic Laboratory

Virtual Steganographic Laboratory (VSL) is a graphical square charting tool that permits complex utilizing, testing and changing of techniques both for picture steganography and steganalysis. VSL furnishes straightforward GUI alongside secluded, module design.

2.10 Xiao Steganography

Xiao Steganography is a half breed steganography tool that permits clients to conceal documents inside pictures (BMP) or sound (WAV) records. The device likewise permits clients to scramble the shrouded record with an assortment of upheld encryption calculations (counting RC4 and 3DES) and hashing calculations (counting SHA and MD5). The client gives a transporter document (the covering for the shrouded record), the record to stow away, a decision of encryption calculation, and a secret key.

2.11 Steganography Studio

Steganography Studio software is a tool to learn, use and analyze key steganographic algorithms. It implements several algorithms highly configurable with a variety of filters. This software is developed in Java, allowing use in any operating system.

2.12 El Carpincho Project

El Carpincho Project is a Java based portable and simple program for text and file encryption (Symmetric, Asymmetric and Steganography)

2.13 Hide & Reveal

Hide & Reveal is both an open-source steganography software and a java library distributed under the GNU GPL. It is primarily designed for scientists wishing to experiment new hiding techniques or steganalysis on various carriers.

Table 1: Table of image steganography tools, source: [9, 13, 15, 17]

Steganography Tools	Brief Description	Image Formats Supported
Hide 'N' Send	Supports encryption and hashing Relies on F5 and LSB steganography algorithms	JPEG
Invisible Secrets 4	Enterprise/commercial software East-tec Corporation software product Contains email-encryption, password manager, file shredder, application locker, IP-IP password transfer and cryptoboard tool	JPEG, PNG, BMP
JHide	Simple steganography tool Java application	BMP
Open Puff	Steganography and watermarking tool Allows to embed data in more than one carrier file	BMP, JPEG, PNG, TGA
OpenStego	Open source Steganography and watermarking tool Java Application Supports encryption and password protection	BMP, PNG
StegHide	Cross platform Open source Supports compression and encryption Relies on graph theory matching algorithm	JPEG, BMP
StegoShare	Supports embedding large files in multiple image carriers Relies on fixed location LSB algorithm	BMP, JPEG, PNG, GIF, TIFF
S-Tool	Steganography tool Supports Encryption and Password protection	BMP, GIF
VSL Virtual Steganographic Laboratory	Image steganography and steganalysis software	BMP, PNG, JPEG, TIFF
Xiao Steganography	Windows platform Developed by Nakasoft Supports encryption, hashing and password protect options	BMP
Steganography Studio	Different hiding methods (LSB, LSB Matching, SLSB), Open source	BMP, PNG, GIF
El Carpincho Project	A new version, with a lot of new and great features like steganography, swing interface, more algorithms, printing, etc	JPG, JPEG, BMP, PNG, TIF, GIF, (Input); PNG (Output)
Hide & Reveal	Allows to hide any type of file within BMP, PNG and TIF images	BMP and PNG

Table 1 lists the various image steganography tools that were selected, and the list is not intended to be an exhaustive one consisting of all possible image steganography tools. Moreover, there is a filter that was applied to separate all steganography tools that were designed in Java and were Windows based with GUI interface. These filtered Java and Windows based steganography tools were tested to see the encode and decode time difference.

3 Selection of Steganography Tools to Test

The steganography tools to be tested were searched with the aid of Google Search Engine and the SourceForge site (https://sourceforge.net). The search terms used for Google Search was "java steganography tool" and the search terms used for the SourceForge site was "steganography java". The tools were then selected from the search results based on the following criteria:

- Has a GUI interface
- Can run on Windows (preferably Windows 10)
- Uses images to hide the secret message or file
- Developed using JAVA
- Not suspected to be malicious software or harboring malicious code.

A total of seven steganography tools was selected for testing when applying the above criteria:

- Jhide
- Open Stego
- StegoShare
- VSL Virtual Steganographic Laboratory
- Steganography Studio
- El Carpincho Project
- Hide & Reveal

4 Data Set Used

Image files that employ lossless compression such as BMP are considered better for steganography compared to image files that employ lossy compression such as JPEG [8, 11]. As such it is possible that one of the suspicious traits of stego images is the use of image file types that are commonly employed for steganography. Hence for the purpose of testing the seven steganography tools, it has been decided to use both BMP (lossless compression) and JPG (lossy compression) cover images.

Table 2: Table of dataset details

Steganography Dataset	Image Size	Image Format	Number of images
Steganalysis Dataset from Mendeley	512x512 px	JPEG, RGB – BMP	1500

4.1 Selection of Cover Image

A steganalysis dataset from Mendeley was used for the purpose of selecting a cover image for testing the steganography tools. From the dataset the C0002.bmp file was selected to be used. A jpg file was created from the selected cover image file using MS Paint software.

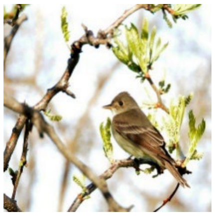


Figure 1: C0002.bmp cover image file

4.2 Selection of Secret Message Used

The secret message, "The quick brown fox jumps over the lazy dog", is stored in a textfile called Test.txt to be encoded in a cover image.

There exists a dataset, Steganalysis Dataset from Mendeley, that was specifically created for the purpose of steganography experiments. This dataset contains 3000 RGB-BMP images, dimensions 512x512, for steganography, steganalysis and similar image processing applications. It contains 1500 RGB-BMP images, dimensions 512x512, transformed from Caltech birds' dataset in JPEGC format. A detailed explanation of the dataset is described in Table 2 below.

Figure 2 represents some of the images collated from the dataset described.

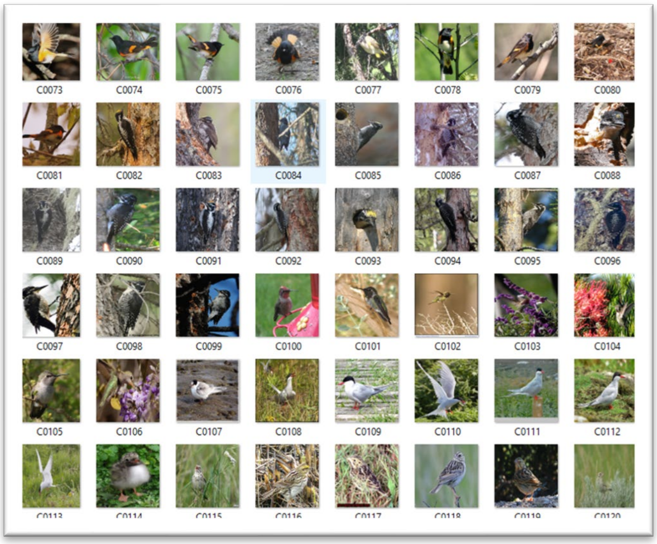


Figure 2: Table of dataset images

5 Evaluation Framework

5.1 Evaluation tool

The weighted score decision matrix was selected to evaluate the seven steganography tools as it is a commonly used in many projects for evaluating software choices [14].

The weighted score decision matrix is a project management technique that is useful in determining or selecting the most appropriate software [14]. It allows a set of choices such as in this case steganography tools to be evaluated in terms of a set of criteria that needs to be considered [5].

5.. The criteria and weightings

One of the aims of steganography is to elicit as little suspicion as possible and hence a few of the criterion will be based on structural detection as well as visual detection. Structural detection involves the comparison of the original cover image and the stego image and visual detection involves the use of the human eye to spot discrepancies in the stego image [6]. With regards to structural detection and visual detection, the following are some of the criteria to be used in the weighted score decision matrix:'

- **Visual Difference** in terms of Visual Distortion/Detectable artifacts/ Unique or unusual colors in image/ Cropping or Padding of Images [2, 10].
- **File Size Difference** [10].
- **File Type Difference** as a change in file type from cover image to stego image will yield more suspicion.
- **Numerous File Type Support** as more file type support will allow the steganography tool to potentially use image file types that are less in use in steganography such as lossy image types [8] to avoid suspicion.

The remaining criteria is related to the performance of the steganography software:

- **Encode Time** to measure time taken to encode secret message in cover image to create stego image.
- **Successful Decode Output** to see if the decoding was successful.
- **Decode Time** to measure time taken to decode secret message in stego image.

The weights for each criterion is applied in the following order:

- **Visual Difference** – weight of 30%.
- **File Size Difference** – weight of 20%.
- **File Type Difference** – weight of 10%.
- **Numerous File Type Support** – weight of 10%.
- **Encode Time** – weight of 5%.
- **Successful Decode Output** – weight of 20%.
- **Decode Time** – weight of 5%.

6 Scoring Process

The following describes how scores are assigned for each criterion. The maximum score for each criterion is 10 and the minimum is 0.

6.1 For Criteria Visual Difference, File Type Difference, Successful Decode Output

- **Visual Difference**, a score of 10 is assigned if there is no discernable visual difference between cover and stego images and a score of 0 is assigned if there is.
- **File Type Difference**, a score of 10 is assigned if there is no change in file type between cover and stego images and a score of 0 is assigned if there is.
- **Successful Decode Output**, a score of 10 is assigned if there is a successful decode output and 0 if none.

6.2 For Criteria File Size Difference, Numerous File Type Support, Encode Time, Decode Time

Scoring process for the following criteria relies on the ranking of raw readings of each test. The following will describe in detail the process of rank scoring:

- **File Size Difference,** the absolute file size difference between the cover and stego image is calculated for each test and sorted in ascending order. Tests with smaller absolute file size difference are ranked higher in comparison to tests with higher absolute file size differences. Scores are assigned based on ranks with the highest rank (rank 1) having a maximum score of 10 and the lowest rank having a score of 0.
- **Numerous File Type Support**, the number of image file types supported is determined for each test and sorted in descending order. Tests with a higher number of image file types supported are ranked higher in comparison to tests with lower number of image file types supported. Scores are assigned based on ranks with the highest rank (rank 1) having a maximum score of 10 and the lowest rank having a score of 0.
- **Encode Time**, the stopwatch app from timeanddate.com/stopwatch on google chrome on mobile phone Samsung M31 was used to record the execution time of the encode time for each test. The execution time is rounded to the nearest higher whole second and then sorted in ascending order. Tests with smaller encode times are ranked higher in comparison to tests with higher encode times. Scores are assigned based on ranks with the highest rank (rank 1) having a maximum score of 10 and the lowest rank having a score of 0.

- **Decode Time**, the stopwatch app from timeanddate.com/stopwatch on google chrome on mobile phone Samsung M31 was used to record the execution time of the decode time for each test. The execution time is rounded to the nearest higher whole second and then sorted in ascending order. Tests with smaller decode times are ranked higher in comparison to tests with higher decode times. Scores are assigned based on ranks with the highest rank (rank 1) having a maximum score of 10 and the lowest rank having a score of 0.

7 Evaluation and Experimental Setup

A test was conducted for each cover image type (jpg and bmp image files), steganography algorithm (if the steganography tool supports more than one) and the steganography tool. At least 2 tests are conducted for each tool. Each test involves an encode attempt using the specified steganography tool, specified cover image (jpg or bmp cover image file), specified steganography algorithm (if the steganography tool supports more than one), and secret message as well as a decode attempt. The output of the encoded attempt (stego image) is compared with the original cover image to see if there are any discernable differences in terms of Visual Difference, File Size Difference, and File Type Difference, with the measurements recorded in the weighted score decision matrix. The encode attempt is also timed (Encode Time) with the measurements recorded in the weighted score decision matrix. If the encode attempt is not successful, no measurements are entered in the weighted score decision matrix and the score assigned is 0 for those criteria where measurements cannot be obtained. The encode output (stego image) is then decoded with the decode attempt timed (Decode Time) and the decode attempt is checked to see if it results in a decode output or secret message (Successful Decode Output). The measurements of the decode attempt, Decode Time, and Successful Decode Output, are then entered in the weighted score decision matrix. The measurements for Numerous File Type Support can be conducted in advance of the tests and entered in the weighted score decision matrix.

The following describes the tests performed for each tool:

- Jhide, 2 tests performed: one using bmp image as cover image and the other using jpg image as cover image.
- Open Stego, 2 tests performed: one using bmp image as cover image and the other using jpg image as cover image.
- StegoShare, 2 tests performed: one using bmp image as cover image and the other using jpg image as cover image.
- VSL Virtual Steganographic Laboratory, 6 tests performed: one per image file type (jpg and bmp file) and steganography algorithm (LSB, KLT, and F5)
- Steganography Studio, 14 tests performed: one per image file type (jpg and bmp file) and steganography algorithm (BattleSteg, BlindHide, DynamicBattleSteg, DynamicFilterFirst, FilterFirst, HideSeek, and SLSB)
- El Carpincho Project, 2 tests performed: one using bmp image as cover image and the other using jpg image as cover image.
- Hide & Reveal, 6 tests performed: one per image file type (jpg and bmp file) and steganography algorithm (Single LSB, Dual LSB, and Triple LSB)

The tests were all conducted on a computer with the following specifications:

- Intel® Core™ i5-6600K CPU @ 3.50GHz
- 32GB RAM
- Windows 10 Pro 64-bit OS Version 21H1 OS build 19043.1110
- Java Version 8 Update 291 (build 1.8.0_291-b10)

8 Experiment Observation and Outcome Results

The following table figures show the experiment outcomes:

It becomes apparent from observing the before mentioned test or experiment data that several steganography tools are not able to handle lossy image files such as jpg files:

- Jhide
- Hide & Reveal

Certain steganography tools also encompass faulty steganography algorithms that may yield no encode output (stego images) such as KLT steganography algorithm from VSL Virtual Steganographic Laboratory which yields no encode output regardless of cover image type.

In terms of Visual Difference, none of the steganography tools that yielded successful encode outputs had any discernable differences when compared visually with the cover image.

With regards to File Size Difference, it would seem most of the file size difference depends on the file image type used as the cover image. Jpg image file types are more likely to yield file size differences in comparison to bmp image file types.

The file size difference may partially be explained when looking at the File Type Difference. Several steganography tools, Open Stego, StegoShare, Steganography Studio, and El Carpincha Project, create the stego image as a different image file type in comparison to the cover image file type used.

With regards to Encode Times and Decode Times of Steganography software, all successful encode and decode tests have been measured as having an execution time of 1 second or less with the sole exception of the test involving Stego Share using jpg cover image which has an execution time of 2 seconds or less for Encode Time.

Only one software, Steganography Studio, has serious issues regarding Successful Decode Output whereby all tests involving this steganography tool has no decode output (cannot retrieve the secret message). VSL Virtual Steganographic Laboratory also has no decode output only when using jpg cover image and LSB as the steganography algorithm.

When looking at the Average Total Weighted Score by

Table 3: Weighted score decision matrix for steganography tools (part 1) - visual and structural detection tests

Tools	Tests on Tools	Visual Detection/Structural Detection																
		Visual Difference				File Size Difference							File Type Difference					
		Y/N	Score	Weight	Weighted Score (3)	File Size of Cover (KB)	File Size of Stego (KB)	Difference	Absolute Value of Difference	Score	Weight	Weighted Score (2)	Image Type - Input	Image Type - Output	Same? Y/N	Score	Weight	Weighted Score (1)
			10	0.3						10	0.2					10	0.1	
Jhide	Jhide - bmp	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
Jhide	Jhide - jpg	*	0	0.3	0	*	*	*	*	0	0.2	0	jpg	*	*	0	0.1	0
Open Stego	Open Stego - jpg	N	10	0.3	3	92.4	768	-675.6	675.6	3.939393939	0.2	0.787878788	jpg	bmp	N	0	0.1	0
Open Stego	Open Stego -bmp	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
StegoShare	StegoShare - jpg	N	10	0.3	3	92.4	645	-552.6	552.6	4.545454545	0.2	0.909090909	jpg	png	N	0	0.1	0
StegoShare	StegoShare - bmp	N	10	0.3	3	768	644	124	124	5.454545455	0.2	1.090909091	bmp	png	N	0	0.1	0
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- jpg, LSB	N	10	0.3	3	92.4	166	-73.6	73.6	5.757575758	0.2	1.151515152	jpg	jpg	Y	10	0.1	1
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- jpg, KLT	*	0	0.3	0	*	*	*	*	0	0.2	0	jpg	*	*	0	0.1	0
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- jpg, F5	N	10	0.3	3	92.4	160	-67.6	67.6	6.060606061	0.2	1.212121212	jpg	jpg	Y	10	0.1	1
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- bmp, LSB	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- bmp, KLT	*	0	0.3	0	*	*	*	*	0	0.2	0	bmp	*	*	0	0.1	0
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- bmp, F5	N	10	0.3	3	768	194	574	574	4.242424242	0.2	0.848484848	bmp	jpg	N	0	0.1	0
Steganography Studio	Steganography Studio - jpg, BattleSteg	N	10	0.3	3	92.4	768	-675.6	675.6	3.939393939	0.2	0.787878788	jpg	bmp	N	0	0.1	0
Steganography Studio	Steganography Studio - jpg, BlindHide	N	10	0.3	3	92.4	768	-675.6	675.6	3.939393939	0.2	0.787878788	jpg	bmp	N	0	0.1	0
Steganography Studio	Steganography Studio - jpg, DynamicBattleSteg	N	10	0.3	3	92.4	768	-675.6	675.6	3.939393939	0.2	0.787878788	jpg	bmp	N	0	0.1	0
Steganography Studio	Steganography Studio - jpg, DynamicFilterFirst	N	10	0.3	3	92.4	768	-675.6	675.6	3.939393939	0.2	0.787878788	jpg	bmp	N	0	0.1	0
Steganography Studio	Steganography Studio - jpg, FilterFirst	N	10	0.3	3	92.4	768	-675.6	675.6	3.939393939	0.2	0.787878788	jpg	bmp	N	0	0.1	0

Steganography Studio	Steganography Studio - jpg, HideSeek	N	10	0.3	3	92.4	768	-675.6	675.6	3.939393939	0.2	0.787878788	jpg	bmp	N	0	0.1	0
Steganography Studio	Steganography Studio - jpg, SLSB	N	10	0.3	3	92.4	768	-675.6	675.6	3.939393939	0.2	0.787878788	jpg	bmp	N	0	0.1	0
Steganography Studio	Steganography Studio - bmp, BattleSteg	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
Steganography Studio	Steganography Studio - bmp, BlindHide	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
Steganography Studio	Steganography Studio - bmp, DynamicBattleSteg	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
Steganography Studio	Steganography Studio - bmp, DynamicFilterFirst	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
Steganography Studio	Steganography Studio - bmp, FilterFirst	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
Steganography Studio	Steganography Studio - bmp, HideSeek	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
Steganography Studio	Steganography Studio - bmp, SLSB	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
El Carpincho Project	El Carpincho Project - jpg	N	10	0.3	3	92.4	536	-443.6	443.6	4.848484848	0.2	0.96969697	jpg	png	N	10	0.1	1
El Carpincho Project	El Carpincho Project - bmp	N	10	0.3	3	768	572	196	196	5.151515152	0.2	1.03030303	bmp	png	N	10	0.1	1
Hide & Reveal	Hide & Reveal - bmp, Single LSB	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
Hide & Reveal	Hide & Reveal - bmp, Dual LSB	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
Hide & Reveal	Hide & Reveal - bmp, Triple LSB	N	10	0.3	3	768	768	0	0	10	0.2	2	bmp	bmp	Y	10	0.1	1
Hide & Reveal	Hide & Reveal - jpg, Single LSB	*	0	0.3	0	*	*	*	*	0	0.2	0	jpg	*	*	0	0.1	0
Hide & Reveal	Hide & Reveal - jpg, Dual LSB	*	0	0.3	0	*	*	*	*	0	0.2	0	jpg	*	*	0	0.1	0
Hide & Reveal	Hide & Reveal - jpg, Triple LSB	*	0	0.3	0	*	*	*	*	0	0.2	0	jpg	*	*	0	0.1	0

* Implies data that cannot be recorded

Table 4: Weighted score decision matrix for steganography tools (part 2) - features and performance tests of steganography tools as well as total weighted score

Tools	Tests on Tools	Features and Performance of Steganography Tools																	
		Numerous File Type Support				Encode Time					Successful Decode Output				Decode Time					Total Weighted Score (10)
		Number of File Types supported	Score	Weight	Weighted Score (1)	Approximate Time	Time Range	Score	Weight	Weighted Score (0.5)		Score	Weight	Weighted Score (2)	Approximate Time	Time Range	Score	Weight	Weighted Score (0.5)	
			10	0.1				10	0.05		Y/N	10	0.2				10	0.05		
Jhide	Jhide - bmp	4	6.36363636	0	0.636364	0.551	1	10	0.05	0.5	Y	10	0.2	2	0.386	1	10	0.05	0.5	9.636363636
Jhide	Jhide - jpg	4	6.36363636	0	0.636364	*	*	0	0.05	0	N	0	0.2	0	*	*	0	0.05	0	0.636363636
Open Stego	Open Stego - jpg,	6	10	0	1	1.076	2	1.8182	0.05	0.090909091	Y	10	0.2	2	0.927	1	10	0.05	0.5	7.378787879
Open Stego	Open Stego -bmp,	6	10	0	1	0.459	1	10	0.05	0.5	Y	10	0.2	2	0.491	1	10	0.05	0.5	10
StegoShare	StegoShare - jpg	5	6.96969697	0	0.69697	0.77	1	10	0.05	0.5	Y	10	0.2	2	0.647	1	10	0.05	0.5	7.606060606
StegoShare	StegoShare - bmp	5	6.96969697	0	0.69697	0.462	1	10	0.05	0.5	Y	10	0.2	2	0.709	1	10	0.05	0.5	7.787878788
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- jpg, LSB	6	10	0	1	0.368	1	10	0.05	0.5	N	0	0.2	0	*	*	0	0.05	0	6.651515152
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- jpg, KLT	6	10	0	1	*	*	0	0.05	0	N	0	0.2	0	*	*	0	0.05	0	1
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- jpg, F5	6	10	0	1	0.395	1	10	0.05	0.5	Y	10	0.2	2	0.541	1	10	0.05	0.5	9.212121212
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- bmp, LSB	6	10	0	1	0.454	1	10	0.05	0.5	Y	10	0.2	2	0.533	1	10	0.05	0.5	10
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- bmp, KLT	6	10	0	1	*	*	0	0.05	0	N	0	0.2	0	*	*	0	0.05	0	1
VSL Virtual Steganographic Laboratory	VSL Virtual Steganographic Laboratory- bmp, F5	6	10	0	1	0.484	1	10	0.05	0.5	Y	10	0.2	2	0.408	1	10	0.05	0.5	7.848484848
Steganography Studio	Steganography Studio - jpg, BattleSteg	4	6.36363636	0	0.636364	0.52	1	10	0.05	0.5	N	0	0.2	0	0.627	1	10	0.05	0.5	5.424242424
Steganography Studio	Steganography Studio - jpg, BlindHide	4	6.36363636	0	0.636364	0.498	1	10	0.05	0.5	N	0	0.2	0	0.516	1	10	0.05	0.5	5.424242424
Steganography Studio	Steganography Studio - jpg, DynamicBattleSteg	4	6.36363636	0	0.636364	0.87	1	10	0.05	0.5	N	0	0.2	0	0.765	1	10	0.05	0.5	5.424242424

Steganography Studio	Steganography Studio - jpg, DynamicFilterFirst	4	6.36363636	0	0.636364	0.566	1	10	0.05	0.5	N	0	0.2	0	0.585	1	10	0.05	0.5	5.424242424
Steganography Studio	Steganography Studio - jpg, FilterFirst	4	6.36363636	0	0.636364	0.65	1	10	0.05	0.5	N	0	0.2	0	0.622	1	10	0.05	0.5	5.424242424
Steganography Studio	Steganography Studio - jpg, HideSeek	4	6.36363636	0	0.636364	0.825	1	10	0.05	0.5	N	0	0.2	0	0.534	1	10	0.05	0.5	5.424242424
Steganography Studio	Steganography Studio - jpg, SLSB	4	6.36363636	0	0.636364	0.576	1	10	0.05	0.5	N	0	0.2	0	0.628	1	10	0.05	0.5	5.424242424
Steganography Studio	Steganography Studio - bmp, BattleSteg	4	6.36363636	0	0.636364	0.689	1	10	0.05	0.5	N	0	0.2	0	0.721	1	10	0.05	0.5	7.636363636
Steganography Studio	Steganography Studio - bmp, BlindHide	4	6.36363636	0	0.636364	0.683	1	10	0.05	0.5	N	0	0.2	0	0.645	1	10	0.05	0.5	7.636363636
Steganography Studio	Steganography Studio - bmp, DynamicBattleSteg	4	6.36363636	0	0.636364	0.505	1	10	0.05	0.5	N	0	0.2	0	0.974	1	10	0.05	0.5	7.636363636
Steganography Studio	Steganography Studio - bmp, DynamicFilterFirst	4	6.36363636	0	0.636364	0.671	1	10	0.05	0.5	N	0	0.2	0	0.616	1	10	0.05	0.5	7.636363636
Steganography Studio	Steganography Studio - bmp, FilterFirst	4	6.36363636	0	0.636364	0.85	1	10	0.05	0.5	N	0	0.2	0	0.715	1	10	0.05	0.5	7.636363636
Steganography Studio	Steganography Studio - bmp, HideSeek	4	6.36363636	0	0.636364	0.534	1	10	0.05	0.5	N	0	0.2	0	0.572	1	10	0.05	0.5	7.636363636
Steganography Studio	Steganography Studio - bmp, SLSB	4	6.36363636	0	0.636364	0.605	1	10	0.05	0.5	N	0	0.2	0	0.559	1	10	0.05	0.5	7.636363636
El Carpincho Project	El Carpincho Project - jpg	6	10	0	1	0.479	1	10	0.05	0.5	Y	10	0.2	2	0.346	1	10	0.05	0.5	8.96969697
El Carpincho Project	El Carpincho Project - bmp	6	10	0	1	0.55	1	10	0.05	0.5	Y	10	0.2	2	0.408	1	10	0.05	0.5	9.03030303
Hide & Reveal	Hide & Reveal - bmp, Single LSB	3	1.51515152	0	0.151515	0.973	1	10	0.05	0.5	Y	10	0.2	2	0.516	1	10	0.05	0.5	9.151515152
Hide & Reveal	Hide & Reveal - bmp, Dual LSB	3	1.51515152	0	0.151515	0.386	1	10	0.05	0.5	Y	10	0.2	2	0.456	1	10	0.05	0.5	9.151515152
Hide & Reveal	Hide & Reveal - bmp, Triple LSB	3	1.51515152	0	0.151515	0.554	1	10	0.05	0.5	Y	10	0.2	2	0.435	1	10	0.05	0.5	9.151515152
Hide & Reveal	Hide & Reveal - jpg, Single LSB	3	1.51515152	0	0.151515	*	*	0	0.05	0	N	0	0.2	0	*	*	0	0.05	0	0.151515152
Hide & Reveal	Hide & Reveal - jpg, Dual LSB	3	1.51515152	0	0.151515	*	*	0	0.05	0	N	0	0.2	0	*	*	0	0.05	0	0.151515152
Hide & Reveal	Hide & Reveal - jpg, Triple LSB	3	1.51515152	0	0.151515	*	*	0	0.05	0	N	0	0.2	0	*	*	0	0.05	0	0.151515152

* Implies data that cannot be recorded

Table 5: Average total weighted score by steganography tool

Tools	Average Total Weighted Score (10)
Jhide	5.1363636
Open Stego	8.6893939
StegoShare	7.6969697
VSL Virtual Steganographic Laboratory	5.9520202
Steganography Studio	6.530303
El Carpincho Project	9
Hide & Reveal	4.6515152

Steganography Tools, the top 3 steganography tool with the highest average total weighted score is the following:

1. El Carpincho Project
2. Open Stego
3. StegoShare

The bottom 3 steganography tool with the lowest average total weighted score is the following:

1. Hide & Reveal
2. Jhide
3. VSL Virtual Steganographic Laboratory

9 Challenges

Finding tools and techniques in computer forensic investigations to decipher the information hidden in steganography when necessary is challenging. It is not only to detect the existence of steganography but is significantly necessary to reveal hidden data information. [3] The development of this general tool for the detection of steganography and classification is still under development. Certain criteria are difficult to measure with sufficient accuracy:

- Measuring Encode Time and Decode Time with sufficient accuracy is difficult due to the manual nature of timing the execution.
- Discerning Visual Difference is subjective depending on the person observing the difference between Cover and Stego images.

There is a slight risk that certain steganography tools that could have met the necessary selection criteria have been overlooked and not included in the experiment.

10 Future Works and Discussion

The functioning rule of image steganography is to hide information in encrypted picture and link them to deliver a final product nearer to the original picture. [12] With the increasing application of steganography in the digital world, many issues must be understood in computer forensic inspection. There is a wide variety of tools and techniques, with their own focus and weaknesses. Stable changes must be made and more recent adaptations have been made. Initially, steganography overview provided computer forensics experts in this field with knowledge and understanding of steganography. A portion of the viewpoints that can be considered for future works are specified herein. Most of image steganography techniques use pictures as the privileged intel and there is a requirement for more examination secluded from text in picture. Analyses identified with upgrading the boundaries and diminishing the capacity limits can be additionally directed utilizing different datasets. The use of tools for observing bit changes in data can also raise suspicion, as investigated. The procedures and algorithms used in steganography are analyzed to be used as the basis for understanding how steganography works. The Image file format is the most widely used digital media medium. [16] Endeavors can be coordinated to shape a benchmark dataset containing pictures from different source cameras and picture designs. An assemblage of all potential calculations should likewise be possible to make the steganography pictures. "

11 Conclusion

It tends to be plainly seen that many picture steganography tools have blemishes that take into consideration the simple recognition or sign of a concealed document in a stego-picture. As such it is basic that individuals engaged with data security use those picture steganography tools that will diminish the opportunity of the message being identified since the strength of steganography is not to make the message ambiguous but instead to cause it to appear as though it was never there in any case. Four devices were discovered to be such devices that will limit the opportunity of outsiders recognizing a shrouded message in the stego-picture. As expressed before this paper is not proposed to be a thorough request and test into all conceivable image-based steganography tools and it is conceivable to additionally improve the weighted score framework used to evaluate the steganography instruments by fusing extra boundaries that may address other significant aspects of steganography, for example, strength.

Many image steganography tools that were randomly selected and the list is not intended to be an exhaustive one consisting of all possible image steganography tools. Additionally, there is a filter that was applied to separate all steganography tools that were designed in Java and were windows based with GUI interface. These filtered Java and Windows based steganography tools were tested to see the encode and decode time difference, the visual differences and file size variance. In any case, the authors place that this paper will be helpful for those specialists meaning to use similar basic pointers of concealed information and those expecting to test, survey and use picture steganography frameworks.

This paper began with a brief overview of the steganography tools to be reviewed, the tests to perform on such tools and the analysis of the results that helped determine which of the image steganography tools will yield stego-images that are nondescript in nature. In synopsis, this paper has expounded on the methods utilized in the new occasions for picture steganography, the latest things. Lastly, this research provides clues for computer forensic inspectors to understand the importance of type steganography tools installed, hidden, or removed on victims' computers. Finding evidence of the suspect, a certain steganography tool will trigger a suspicious impression. As shown in the results of the experiment, it is necessary to know the type of steganography tool. It very well may be inferred that deep learning has enormous potential in the picture steganography field contemplating that every one of the difficulties and challenges are filled.

References

[1] A. Aljarf, S. Amin and J. Filippas, "Creating Stego-Images Through Hiding Single and Multipile Data Using Different Steganographic Tools," *Proceedings of the IASTED International Conference*, Innsbruck, pp. 343-348, 2013.

[2] A. Almohammad and G. Ghinea, "Stego Image Quality and the Reliability of PSNR," 2010 2nd International Conference on Image Processing Theory, Tools and Applications, Paris, 2010.

[3] J. Butora and J. Fridrich, "Effect of JPEG Quality on Steganographic Security," *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, 2019.

[4] T. Denemark, P. Bas, and J. Fridrich, "Natural Steganography in JPEG Compressed Images," *Electronic Imaging,* 7:1-10, 2018.

[5] Airfocus GmbH, "Weighted Scoring," [Online]. Available: https://airfocus.com/guides/prioritization/7-most-popular-prioritization-frameworks/weighted-scoring/, [Accessed 2 September 2021].

[6] N. A. Hassan and R. Hijazi, *Data Hiding Techniques in Windows OS*, Syngress, 2017.

[7] A. Kaur, R. Kaur, and N. Kumar, "A Review on Image Steganography Techniques," *International Journal of Computer Applications,* 123(4):20-24, 2015.

[8] G. C. Kessler, "Steganography: Hiding Data Within Data," September 2001, [Online], Available: https://www.garykessler.net/library/steganography.html. [Accessed 1 September 2021.

[9] V. Kumar, "30 Useful Steganography Tools To Hide Secret Information," 2 September 2017, [Online], Available: https://www.rankred.com/steganography-tools-to-hide-secret-information/. [Accessed 17 October 2019].

[10] A. Kumar and K. Pooja, "Steganography - A Data Hiding Technique," *International Journal of Computer Applications,* 9(7):19-23, 2010.

[11] X. Liao, J. Yin, S. Guo, X. Li, and A. K. Sangaiah, "Medical JPEG Image Steganography Based on Preserving Inter-Block Dependencies," *Computers & Electrical Engineering,* 67:320-329, 2018.

[12] L. Liu, Z. Wang, Z. Qian, X. Zhang, and G. Feng, "Steganography in Beautified Images," *Mathematical Biosciences and Engineering,* 16(4):2322-2333, 2019.

[13] C. Maji, "Describing Textures in the Wild," *IEEE Conf. on Computer Vision and Pattern Recognition*, 2014.

[14] N. Morpus, "A Step-by-Step Guide for Using a Weighted Scoring Model," 2 September 2021. [Online], Available: https://www.fool.com/the-blueprint/weighted-scoring-model/, [Accessed 16 January 2021].

[15] H. Passi, "Top 10 Must-Have Tools to Perform Steganography," GreyCampus, 5 October 2018, [Online], Available: https://www.greycampus.com/blog/information-security/top-must-have-tools-to-perform-steganography, [Accessed 17 October 2019].

[16] D. R. I. M. Setiadi, E. H. Rachmawanto, and C. A. Sari, "Secure Image Steganography Algorithm Based on DCT with OTP Encryption," *Journal of Applied Intelligent System,* 2(1):1-11, 2017.

[17] P. Shankdhar, "Best Tools to Perform Steganography [Updated 2019]," 17 May 2019, [Online], Available: https://resources.infosecinstitute.com/steganography-and-tools-to-perform-steganography/#gref, [Accessed 17 October 2019].

[18] V. Sharma and S. Kumar, "A New Approach to Hide Text in Images Using Steganography," *International Journal of Advanced Research in Computer Science and Software Engineering,* III(4):701-708, 2013.

[19] A. Siper, R. Farley, and C. Lombardo, "The Rise of Steganography," 6 May 2005, [Online], Available: http://csis.pace.edu/~ctappert/srd2005/d1.pdf, [Accessed 17 October 2019].

[20] M. E. Whitman and H. J. Mattord, *Principles of Information Security*, 5th ed., Boston: Cengage Learning, 2014.

Anal Kumar was awarded a BIT degree from the University of Fiji in 2009 and Master of Science in Information Technology in 2016. He is currently a Lecturer at Department of Computing Sciences and Information Systems at Fiji National University and pursuing PhD in

Information Technology through the University of Fiji. E-mail: anal.kumar@fnu.ac.fj

Hermann Jamnadas completed his Bachelor of Commerce in Accounting and Information Systems from The University of the South Pacific, Laucala, Fiji in the year of 2011. He obtained his Postgraduate Diploma in University of Fiji, Saweni, Fiji in the year 2016 as well as his Master of Information Technology from The University of Fiji, Information Technology from The Saweni, Fiji in the year 2022. E-mail: hermann.jamnadas@fnu.ac.fj

Vishal Sharma completed his Masters of Computing Science and Information in 2013 from the University of the South Pacific, along with other IT certifications such as CCNA, CISSP and Professional Training in Big Data Analytics. He is an academic with almost 15 years of experience and has research interest in areas of Networking, Mobile Commerce, Big data Analytics, Cybersecurity. E-mail: vishal.sharma@fnu.ac.fj

S.M Muyeen is presently working as a Professor in the Electrical Engineering Department of Qatar University, Doha, Qatar. He received his Ph.D. from the Kitami Institute of Technology, Japan, in Electrical and Electronic Engineering and worked in Japan under the versatile banner of the Japan Society for the Promotion of Science (JSPS). He holds visiting/adjunct positions with many foreign universities, e.g., Shanghai Maritime University, China, Curtin University, Australia, Shanghai University of Electric Power, China. E-mail: ieee.csde@gmail.com

ABM Shawkat Ali is a Bangladeshi origin-Australian author, computer scientist and data analyst. He is the author of several books in the area of Data Mining, Computational Intelligence, and Smart Grid. He is a newspaper columnist. He is an academic and well-known researcher in the areas of Machine Learning and Data Science. He is also the founder of a research center and international conferences in Data Science and Engineering. He is now a Professor in Data Science at the University of Fiji. E-mail: abm.shawkat.ali@gmail.com

Journal Submission

The International Journal of Computers and Their Applications is published four times a year with the purpose of providing a forum for state-of-the-art developments and research in the theory and design of computers, as well as current innovative activities in the applications of computers. In contrast to other journals, this journal focuses on emerging computer technologies with emphasis on the applicability to real world problems. Current areas of particular interest include, but are not limited to: architecture, networks, intelligent systems, parallel and distributed computing, software and information engineering, and computer applications (e.g., engineering, medicine, business, education, etc.). All papers are subject to peer review before selection.

A. Procedure for Submission of a Technical Paper for Consideration

1. Email your manuscript to the Editor-in-Chief, Dr. Ajay Bandi. Email: ajay@nwmissouri.edu.
2. Illustrations should be high quality (originals unnecessary).
3. Enclose a separate page (or include in the email message) the preferred author and address for correspondence. Also, please include email, telephone, and fax information should further contact be needed.
4. **Note**: Papers shorter than 10 pages long will be returned.

B. Manuscript Style:

1. **WORD DOCUMENT**: The text should be **double-spaced** (12 point or larger), **single column** and **single-sided** on 8.5 X 11 inch pages. Or it can be single spaced double column.

   LaTex DOCUMENT: The text is to be a double column (10 point font) in pdf format.
2. An informative abstract of 100-250 words should be provided.
3. At least 5 keywords following the abstract describing the paper topics.
4. References (alphabetized by first author) should appear at the end of the paper, as follows: author(s), first initials followed by last name, title in quotation marks, periodical, volume, inclusive page numbers, month and year.
5. The figures are to be integrated in the text after referenced in the text.

C. Submission of Accepted Manuscripts

1. The final complete paper (with abstract, figures, tables, and keywords) satisfying Section B above in **MS Word format** should be submitted to the Editor-in-Chief. If one wished to use LaTex, please see the corresponding LaTex template.
2. The submission may be on a CD/DVD or as an email attachment(s). **The following electronic files should be included:**
   - Paper text (required).
   - Bios (required for each author).
   - Author Photos are to be integrated into the text.
   - Figures, Tables, and Illustrations. These should be integrated into the paper text file.
3. Reminder: The authors photos and short bios should be integrated into the text at the end of the paper. All figures, tables, and illustrations should be integrated into the text after being mentioned in the text.
4. The final paper should be submitted in (a) pdf AND (b) either Word or LaTex. For those authors using LaTex, please follow the guidelines and template.
5. Authors are asked to sign an ISCA copyright form (http://www.isca-hq.org/j-copyright.htm), indicating that they are transferring the copyright to ISCA or declaring the work to be government-sponsored work in the public domain. Also, letters of permission for inclusion of non-original materials are required.

Publication Charges

After a manuscript has been accepted for publication, the contact author will be invoiced a publication charge of **$500.00 USD** to cover part of the cost of publication. For ISCA members, publication charges are **$400.00 USD** publication charges are required.

Revised 2020

ISCA INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS Vol. 30 No. 1, March 2023