International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 2 Volume No.7 March - April – 2023
DOI:10.46647/ijetms.2023.v07i02.085 ISSN: 2581-4621

EFFECTIVE APPROACH FOR FACE RECOGNITION AND ACTIVE SHAPE 3D MODELS USING KERNEL PRINCIPAL COMPONENT ANALYSIS

Surya Kant Prajapati[1], Mrs. C. Navamani M.E[2]
[1]*PG Scholar, Department of Computer Science and Engineering, Nandha Engineering College (Autonomous), Erode, Tamilnadu, India[1]*
[2]*Assistant Professor, Department of Computer Science and Engineering, Nandha Engineering College (Autonomous), Erode, Tamilnadu, India[2]*

ABSTRACT
Face Recognition is a computer application that is capable of detecting, tracking, identifying or verifying human faces from an image or video captured using a digital camera. Although lot of progress has been made in domain of face detection and recognition for security, identification and surveillance purpose, but still there are issues hindering the progress to reach or surpass human level accuracy. These issues are variations in human facial appearance such as; varying lighting condition, noise in face images, scale, pose etc. Kernel principal component analysis (KPCA) as a powerful nonlinear feature extraction method has proven as a preprocessing step for classification algorithm. A face recognition approach based on KPCA and genetic algorithms (GAs) is proposed. By the use of the polynomial functions as a kernel function in KPCA, the high order relationships can be utilized and the nonlinear principal components can be obtained. After that nonlinear principal components, we use GAs to select the optimal feature set for classification.
KEYWORDS: Face recognition, human faces, KPCA, classification algorithm, GAs, surveillance, feature, classification.

1. INTRODUCTION

1.1 KERNEL PRINCIPAL COMPONENT ANALYSIS

A brief background study of biometric and face recognition algorithms were presented. To study assesses the performance and KPCA Computational time and face recognition accuracy. The experimental results shown an Average Testing Time of 1.54 seconds for PCA and 67.0929 seconds for KPCA, it implies that it takes a longer Computational time for KPCA than PCA. However, the experiment revealed that 72.5% performance recognition accuracy while KPCA has 80.0% performance recognition accuracy, indicating that KPCA outperforms the PCA in terms of recognition accuracy should be noted that the results were basically limited by configuration of the computer system used, resolution of the digital camera, different environmental conditions like illumination and different distances between the camera and every face. In summary PCA tradeoff recognition accuracy for testing time while KPCA tradeoff testing time for recognition accuracy.

1.2 KERNEL PRINCIPAL COMPONENT ANALYSIS FEATURES

- Kernel PCA is the nonlinear form of PCA, which better exploits the complicated spatial structure of high-dimensional features.
- Reconstruction of pre-images for kernel PCA.
- Linear dimensionality reduction and feature extraction
- Reconstruction of pre-images for kernel PCA.
- Pattern Classification for Synthetic Data
- Classification for Aligned Human Face Images

1.3 FACE RECOGNITION

Step 1: Face detection
The camera detects and locates the image of a face, either alone or in a crowd. The image may show the person looking straight ahead or in profile.

Step 2: Face analysis
Next, an image of the face is captured and analyzed. Most facial recognition technology relies on 2D rather than 3D images because it can more conveniently match a 2D image with public photos or those in a database. The software reads the geometry of your face. Key factors include the distance between your eyes, the depth of your eye sockets, the distance from forehead to chin, the shape of your cheekbones, and the contour of the lips, ears, and chin. The aim is to identify the facial landmarks that are key to distinguishing the face.
Step 3: Converting the image to data
The face capture process transforms analog information (a face) into a set of digital information (data) based on the person's facial features. Your face's analysis is essentially turned into a mathematical formula. The numerical code is called a faceprint. In the same way that thumbprints are unique, each person has their own face print.
Step 4: Finding a match
The faceprint is then compared against a database of other known faces. For example, the FBI has access to up to 650 million photos, drawn from various state databases. On Facebook, any photo tagged with a person’s name becomes a part of Facebook's database, which may also be used for facial recognition. If the faceprint matches an image in a facial recognition database, then a determination is made.

2. LITERATURE REVIEW

2.1 ROOTKIT DETECTION VIA INVARIANT INFERENCE:

It proposes a novel approach that detects rootkits by identifying violations of automatically-inferred kernel data structure invariants. Section 3 presents an overview of our approach, and presents examples of both control and non-control data attacks that were detected in our experiments. Kernel-level rootkits are a form of malicious software that compromise the integrity of the operating system. Such rootkits stealthily modify kernel data structures to achieve a variety of malicious goals, which may include hiding malicious user space objects, installing backdoors, logging keystrokes and disabling firewalls. Recent studies have shown a phenomenal increase in malware that use stealth techniques commonly employed by rootkits. The most recent list of threat predictions, also by MacAfee, contains several recent examples of Trojan horses that were used to commit bank fraud. These Trojan horses used stealth techniques to hide on a victim’s system, disable anti-virus software, prevent signature updates, or include the victim’s system into a botnet. The increase in the number and complexity of rootkits can be attributed to the large and complex attack surface that the kernel presents.

2.2 REAL-TIME OPEN-SOURCING DRIVER IMPLEMENTATION:

Assessment of the Real-time Preemption Patches (RT-Preempt) and their impact on the general purpose performance of the system. With the maturing of the Real-time Preemption Patches (RT-Preempt) and their stepwise integration into the Mainline Linux kernel since version 2.6.18, we set out to answer the questions: * How good is RT-Preempt with respect to the worst-case latency? * How expensive is RT-Preempt with respect to a possible performance degradation of the system? Taking that a lot of the preemption techniques deployed have their origin in scalability demands and not so much in real-time requirements, the most interesting case to look into is related to uni-processors -on these we would expect the worst-case impact of RT-Preempt. To answer the question, we ran an extensive benchmark series on 2.8-GHz P4 and 1-GHz/600MHz VIA CIII boards, measuring general OS performance parameters as well as the real-time capabilities. For the latter, a trivial par port toggle program was used. The results show that high-end CPUs are well supported by RT-preempt in general. Low-end systems typically of interest for automation and control, however, still need some work. In this paper we will outline the method used for evaluation and present the details of the results.

2.3 DUAL-DEDUP LOWER LAYER BLOCK DEDUPLICATION TO THE KNOWLEDGE OF PAGE CACHE MANAGEMENT:

The amount of data being produced and consumed is increasing every day. As a result, there can be a large amount of redundant data in the storage system. Storing and accessing these duplicate data unnecessarily consumes disk space and I/O bandwidth. Deduplication techniques are widely deployed to remove the redundancy. In particular, the deduplication solutions that work at the block level are proven to be effective. These solutions aim to effectively use disk space and write bandwidth by avoiding duplicate data writes to the storage. However, such a design might not help in improving the read performance, which is critical for many modern-day applications. The Linux kernel implements an in-memory cache of pages, called the page cache, to improve I/O performance by minimizing disk accesses. The page cache has pages originating from regular file systems, and it is indexed by a file and the offset within the file. However, due to such a design, deduplication information is currently not available to the page cache. Due to this, the kernel cannot avoid read requests from going to the disk on offsets that are not present in the page cache, even though the requested data duplicates another offset that is already cached.

2.4 NVIDIA JETSON AGX XAVIER:

NVIDIA Jetson is the world’s leading platform for AI at the edge. Its high-performance, low-power computing for deep learning and computer vision makes it the ideal platform for compute-intensive projects. The Jetson platform includes a variety of Jetson modules together with NVIDIA JetPack™ SDK. Each Jetson module is a computing system packaged as a plug-in unit (a System on Module (SOM)). NVIDIA offers a variety of Jetson modules with different capabilities. JetPack bundles all of the Jetson platform software, starting with NVIDIA Jetson Linux. Jetson Linux provides the Linux kernel, bootloader, NVIDIA drivers, flashing utilities, sample file system, and more for the Jetson platform.

2.5 ADDRESSING ANDROID KERNEL ACTIVITY EXTENSIVELY, A KERNEL BASED CROWDNET ARCHITECTURE FOR CLOUD COMPUTING PLATFORMS:

Neural network can maintain the performance of identifying Android malware by learning Android characteristics. But owing to the rapid growth of information on Android devices, it cannot deal with the massive data efficiently for millions of Android applications. In this paper, we propose a RaNetMalDozer framework for malware detection on Android phones to improve the accuracy rate of classification and the training speed. The RaNetMalDozer (RNMD) can dynamically select hidden centers by a heuristic approach and efficiently gather the large-scale datasets of 2550 Android applications by our automatic RNMD data collector. To systematically analyze Android kernel behaviors, we investigate 112 Android kernel features of task_struct and implement EBPN and RNMD methods. Furthermore, compared to the traditional neural network, the EBPN method which achieves 84% accuracy rate, the RNMD method can achieve 94% accuracy rate with the half of training and evaluation time. Our experiments also demonstrate the RNMD method can be used as a better technique of the Android malware detection.

2.6 MUTUAL DATA MONITORING UNDER THE IMPERFECT OF WIRELESS NETWORKING HAS BEEN EXAMINED IN A CLASS OF MULTI-AGENT SYSTEMS:

Android Inc. was founded in Palo Alto, CA in Oct. 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White to develop, in Rubin’s words, “smarter mobile devices that are more aware of its owner’s location and preferences”. The Android OS was originally developed to work with digital cameras but they soon realized that market was too small, the company then outfits effort into producing a smartphone operating system that would rival Symbian and Microsoft Windows Mobile. In July 2005 Google acquired Android Inc. for about $50 million, the key employees stayed with Google after being bought out. Once Google bought the company, they developed a mobile device platform powered by the Linux kernel. Google marketed the platform to handset makers and carriers on the basis of providing a flexible, upgradable system. On Nov. 5, 2007 the Open Handset Alliance, which is a group of technology companies including Google, device manufacturers such as HTC, Sony and Samsung, wireless carriers such as Sprint Nextel and T-Mobile, and chipset makers such as Qualcomm and Texas Instruments, stated their goal collectively was to develop open standards for mobile devices. The same day Android was unveiled as its first product, a mobile device platform built on the Linux kernel.

2.7 APPLICATION DEVELOPMENT RESEARCH BASED ON ANDROID PLATFORM:

The n-kernel ROP (Return Oriented Programming) is a useful technique that is often used to bypass restrictions associated with non-executable memory regions. For example, on default kernels1, it presents a practical approach for bypassing kernel and user address separation mitigations such as SMEP (Supervisor Mode Execution Protection) on recent Intel CPUs. The above privilege escalation payload allocates a new credential struct (with uid = 0, gid = 0, etc.) and applies it to the calling process. We can construct a ROP chain that will perform the above operations without executing any instructions residing in user space, i.e., without setting the program counter to any user-space memory addresses. The end goal is to execute the entire privilege escalation payload in kernel space using a ROP chain. This is may not be required in practice, however. For example, in order to bypass SMEP, it is sufficient to flip the SMEP bit using a ROP chain and then a standard privilege escalation payload can be executed in user space.

2.8 STUDY OF SECURITY FLAWS IN THE LINUX KERNEL BY FUZZING:

A microkernel is a software or a program in which user services and kernel services are present in different address space. Due to which the size of the microkernel becomes smaller than that of a monolithic kernel. But as the user services and kernel services are in different address space in order for a user service to use a kernel service, message passing was used. This makes the execution of microkernel to be slower. The microkernel is easily extendible. Due to which if a new service has to be added then it would not require any changes to the kernel itself. Also, if any user service crashes it doesn't affect the working of the microkernel. The examples of microkernel would be QNX, minx, Symbian, Mac OS X, L4Linux, Integrity, K42, etc.

2.9 MONOLITHIC AND MICROKERNEL APPROACHES:

Data-intensive science is a scientific discovery process that is driven by knowledge extracted from large volumes of data rather than the traditional hypothesis driven discovery process. One of the key challenges in data-intensive science is development of enabling technologies to allow researchers to effectively utilize these large volumes of data in an effective manner. This paper introduces the concept of data prospects to address the challenges of data intensive science. With data prospecting, we extend the familiar metaphor of data mining to describe an initial phase of data exploration used to determine promising areas for deeper analysis. Data prospecting enhances data selection through the use of interactive discovery engines. Interactive exploration enables a researcher to filter the data based on the first look analytics, discover interesting and previously unknown patterns to start new science investigations, verify the quality of the data, and corroborate whether patterns in the data match existing science theories or mental models.

2.10 LINUX KERNEL FUNCTION CALL:

The architecture of most modern processors, with the exception of some embedded systems, involves a security model. For example, the rings model specifies multiple privilege levels under which software may be executed: a program is usually limited to its own address space so that it cannot access or modify other running programs or the operating system itself, and is usually prevented from directly manipulating hardware devices (e.g. the frame buffer or network devices).

2.11 KERNEL-BASED FEATURE EXTRACTION WITH A SPEECH TECHNOLOGY APPLICATION:

Kernel principal component analysis (KPCA) is a powerful and widely applied nonlinear feature extraction technique. However, as originally proposed, KPCA may be cumbersome or infeasible in large scale data sets, which motivated the development of low complexity iterative extraction algorithms, mainly aiming image processing applications. Recently, some online KPCA extraction algorithms were proposed, but most of them suffer from low-convergence speed. This paper proposes a new algorithm based on fixed point iterative equations for KPCA extraction, expanding kernel components using a compact dictionary, dynamically built from data, according to a user defined accuracy parameter. The algorithm relies on simple equations, can track nonstationary environments, and requires reduced storage, enabling its use in real-time applications operating in low-cost embedded hardware. Results involving open-access image data sets show improved accuracy and convergence speed, as well as permitted effective improvements in practical image applications, as compared to state-of-art online KPCA techniques.

2.12 CONTROLLING OF PROCESSES EXECUTION AND SCHEDULING:

The kernel schedules its own operations is fundamentally different from the way it schedules threads. A request for kernel-mode execution can occur in two ways. A running program may request an operating-system service (system call or page fault), or a device controller may deliver a hardware interrupt that causes the CPU to start executing a kernel-defined handler for that interrupt. The problem for the kernel is that all these tasks may try to access the same internal data structures. This creates the potential of data corruption, and a common way to address it is critical section – portions of code that access shared data and must not be allowed to execute concurrently. Linux Kernel uses spinlocks and semaphores (as well as reader-writer versions of these two lock) for locking in the kernel.

2.13 IOT INFLUENCES IN OS KERNEL OPERATIONS:

Internet of Things (IoT) is rapidly growing and contributing drastically to improve the quality of life. Immense technological innovations and growth is a key factor in IoT advancements. Readily available low cost IoT hardware is essential for continuous adaptation of IoT. Advancements in IoT Operating System (OS) to support these newly developed IoT hardware along with the recent standards and techniques for all the communication layers are the way forward. The variety of IoT OS availability demands to support interoperability that requires to follow standard set of rules for development and protocol functionalities to support heterogeneous deployment scenarios. IoT requires to be intelligent to self-adapt according to the network conditions. In this paper, we present brief overview of different IoT OSs, supported hardware, and future research directions. Therein, we provide overview of the accepted papers in our Special Issue on IoT OS management: opportunities, challenges, and solution. Finally, we conclude the manuscript.

2.14 STRAIGHT LINE DISTANCE BETWEEN TWO POINTS IN THE PLANE:

System extensions are modern replacements of kernel extensions in macOS Catalina. With system extensions, Apple provides new frameworks for developers to perform tasks previously reserved for kernel extensions. The primary new benefit of system extensions is that they run in the user space rather than in the kernel space; by running in the user space, system extensions cannot compromise the built-in security or stability of macOS. Although kernel extensions do still work in macOS Catalina, Apple has deprecated the use of certain types of kexts and developers should work to move their kexts to system extensions as equivalent system extension frameworks become available. Currently, there are three new system extension frameworks available to replace kexts. Kexts that operate outside of these new frameworks (such as virtualization software like VMware Fusion) must continue to use kexts until Apple offers equivalent system extension frameworks.

2.15 TESTING TIME (TT) BETWEEN K UNIT AND K SELF TEST:

K Unit (K Unit - Linux Kernel Unit Testing) is an entirely in-kernel system for “white box” testing: because test cod e is part of the kernel, it can access internal structures and functions which aren’t exposed to userspace. K Unit tests therefore are best written against small, self-contained parts of the kernel, which can be tested in isolation. This aligns well with the concept of ‘unit’ testing. For example, a K Unit test might test an individual kernel function (or even a single code path through a function, such as an error handling case), rather than a feature as a whole. This also makes K Unit tests very fast to build and run, allowing them to be run frequently as part of the development process. There is a K Unit test style guide which may give further pointers in Test Style and Nomenclature k self test (Linux Kernel Selftests), on the other hand, is largely implemented in userspace, and tests are normal userspace scripts or programs. This makes it easier to write more complicated tests, or tests which need to manipulate the overall system state more (e.g., spawning processes, etc.). However, it’s not possible to call kernel functions directly from k self test. This means that only kernel functionality which is exposed to userspace somehow (e.g. by a syscall, device, filesystem, etc.) can be tested with k self test. To work around this, some tests include a companion kernel module which exposes more information or functionality. If a test runs mostly or entirely within the kernel, however, K Unit may be the more appropriate tool. K self test is therefore suited well to tests of whole features, as these will expose an interface to userspace, which can be tested, but not implementation details. This aligns well with ‘system’ or ‘end-to-end’ testing.

2.16 IMPLEMENTATION LINUX 4.0.4 KERNELS:

Applications of an embedded system have been expanded throughout the years. It is usually observed in the digital consumer market where cell phones are replaced by smart phones and internet tablets. More functionality is brought into one single purpose system for giving the enhanced services to the users. The Linux Kernel based operating system is used in most of these electronic devices. The biggest protests about Linux are the size and speed at which it boots. The users expect these devices to have fast boot up time and a small amount of memory. In multiple real-time systems, many Linux versions are used that need a large level of accessibility and negligible downtime when updating systems. This results in advancing Linux boot time and size optimization. The objective of this paper is to keep the researchers up to date about Linux Optimization techniques. First of all, this paper survey the various boot time optimization techniques and size reduction techniques with respect to conventional Linux system.

2.17 KERNEL MALWARE ANALYSIS WITH UN-TAMPERED AND TEMPORAL VIEWS OF DYNAMIC KERNEL MEMORY:

Dynamic kernel memory has been a popular target of recent kernel malware due to the difficulty of determining the status of volatile dynamic kernel objects. Some existing approaches use kernel memory mapping to identify dynamic kernel objects and check kernel integrity. The snapshot-based memory maps generated by these approaches are based on the kernel memory which may have been manipulated by kernel malware. In addition, because the snapshot only reflects the memory status at a single time instance, its usage is limited in temporal kernel execution analysis. We introduce a new runtime kernel memory mapping scheme called allocation-driven mapping, which systematically identifies dynamic kernel objects, including their types and lifetimes. The scheme works by capturing kernel object allocation and deallocation events. Our system provides a number of unique benefits to kernel malware analysis: One an un-tampered view wherein the mapping of kernel data is unaffected by the manipulation of kernel memory and a temporal view of kernel objects to be used in temporal analysis of kernel execution. We demonstrate the effectiveness of allocation-driven mapping in two usage scenarios. First, we build a hidden kernel object detector that uses an un-tampered view to detect the data hiding attacks of 10 kernel rootkits that directly manipulate kernel objects (DKOM). Second, we develop a temporal malware behavior monitor that track and visualizes malware behavior triggered by the manipulation of dynamic kernel objects. Allocation-driven mapping enables a reliable analysis of such behavior by guiding the inspection only to the
events relevant to the attack.

2.18 L4 MICROKERNELS:

The L4 microkernel has undergone 20 years of use and evolution. It has an active user and developer community, and there are commercial versions that are deployed on a large scale and in safety-critical systems. The lessons learnt in those 20 years about microkernel design and implementation. We revisit the L4 design papers, and examine the evolution of design and implementation from the original L4 to the latest generation of L4 kernels. We specifically look at seL4, which has pushed the L4 model furthest and was the first OS kernel to undergo a complete formal verification of its implementation as well as a sound analysis of worst-case execution times. We demonstrate that while much has changed, the fundamental principles of minimality, generality and high inter-process communication (IPC) performance remain the main drivers of design and implementation decisions.

2.19 VARIABILITY BUGS IN THE LINUX KERNEL: A QUALITATIVE ANALYSIS:

Feature-sensitive verification pursues effective analysis of the exponentially many variants of a program family. However, researchers lack examples of concrete bugs induced by variability, occurring in real large-scale systems. Such a collection of bugs is a requirement for goal-oriented research, serving to evaluate tool implementations of feature-sensitive analyses by testing them on real bugs. We present a qualitative study of variability bugs collected from bug-fixing commits to the Linux kernel repository. Analyse each of the bugs, and record the results in a database. In addition, we provide self-contained simplified C99 versions of the bugs, facilitating understanding and tool evaluation. Our study provides insights into the nature and occurrence of variability bugs in a large C software system, and shows in what ways variability affects and increases the complexity of software bugs.

2.20 DYNAMICS KERNEL FOR THE MC2 MODEL II: FLEXIBLE GMRES ELLIPTIC SOLVER:

Traditional semi-implicit formulations of nonhydrostatic compressible models may not be stable in the presence of steep terrain when pressure gradient terms are split and lagged in time. If all pressure gradient terms and the divergence are treated implicitly, the resulting wave equation for the pressure contains off-diagonal cross-derivative terms leading to a highly nonsymmetrical linear system of equations. In this paper we present a more implicit formulation of the Mesoscale Compressible Community (MC2) model employing a Generalized Minimal Residual (GMRES) Krylova iterative solver and a more efficient semi-LaGrange advection scheme. Open boundaries now permit exact upwind interpolation and the ability to reproduce simulations to machine precision is illustrated for one-way nesting at equivalent resolution. Numerical simulations of hydrostatic and nonhydrostatic mountain waves demonstrate the stability and accuracy of the new adiabatic kernel. The computational efficiency of the model is reported for 1D Jacobi and 3D Alternating Direction Implicit (ADI) line relaxation preconditioned implemented with a parallel data transposition strategy.

3. COMPARATIVE ANALYSIS TABLE

Title	**Techniques & Mechanisms**	**Parameter Analysis**	**Future Work**
Rootkit detection via invariant inference	Rootkit-detection in user-mode COMEX experiences execution	Disintegration of Resources 32 Dell servers	Danger of rootkits will be to conceal complex targeted attacks
Real-time open-sourcing driver Implementation	PREEMPT RT output has been enhanced. Xenomai best result.	Lowest power output	Efficiency of kernel will be improved
Dual-Dedup lower layer block deduplication to the knowledge of page cache Management	Linux EXT4	Enhanced read performance 34 percent by using FIO benchmarks for a data collection of 25 percent duplicate data.	Reduce traffic congestion and parking lot concerns in metropolitan regions

NVIDIA Jetson AGX Xavier	Memory-aware equal share scheduling algorithm	Isolated the real stall from the CPU cycles	Lower TDP GPUs
Addressing Android kernel activity extensively, a kernel based CrowdNet architecture for cloud computing platforms	Science, technology, and innovation policy (STIP)	Protected large- scale data validation and doubled kernel learning.	The role of data science in policy analysis is being examined
Mutual data monitoring under the imperfect of wireless networking has been examined in a class of multi-agent linear systems	STM 32-running Revolutionary Internet of Things	Restricted Bandwidth and attachment connections	To improve the CPU environment and resources
Application Development Research Based on Android Platform	Association for Computing Machinery (ACM)	Analyze network reliability when the networks are interrupted or attacked based on the process of the methodology developed.	Help to understand the functionality of the system and Design effective methods of Software development
Study of Security Flaws in the Linux Kernel by Fuzzing	Security implement as a solution and how to prevent this violation of User information to optimize Security.	Bootstrap Resampling	Demonstrates to stop fraud of device permissions by holding a lock on personal user ID set software.

Monolithic and Microkernel Approaches	Context-aware framework for IoT controller design	Context-aware framework for IoT controller design.	Useful for new science education
Linux Kernel function call	Feature Extraction Algorithms	Principal Component Analysis (PCA)	This method can detect subtle device upsets that are not detectable by normal I/O or attached devices to the user.
Kernel-Based Feature Extraction with a Speech Technology Application	AdaBoost, AUC-RF	PLINK’s indep-pairwise	We can reduce the time spent to find susceptibility SNPs/genes to establish the diagnostic tools
Controlling of Processes Execution and Scheduling	CPU allocation cost	Linear Discriminant Analysis (LDA)	web hypermedia systems
IoT influences in OS kernel operations	web-enabled mobile applications	Discrete Cosine Transform (DCT)	healthcare, agriculture, security surveillance,
Straight line distance between two points in the plane	Euclidean metric classifier	Active Shape Models (ASM)	Euclidean metric classifier
Testing Time (TT) Between K Unit and k self test	PCA	Independent Component Analysis (ICA)	K PCA recognized 32 images

Implementation Linux 4.0.4 kernels	Hyper-vision performance	Da Ca Po benchmark 9.19	KPCA Computational time and face recognition accuracy
Kernel Malware Analysis with Un-tampered and Temporal Views of Dynamic Kernel Memory	Static Analysis	malware information	Static Analysis
L4 Microkernels: The Lessons from 20 Years of Research and Deployment	Portable Operating System Interface (POSIX)	x86 through AMD64 to several ARM platforms.	Portable Operating System Interface (POSIX)
Variability bugs in the Linux kernel: A qualitative analysis	Ticket lock mechanism	DPDK4.18.147.8 .1.rt24.101.el8_1. x86_64	Ticket lock mechanism
Dynamics Kernel For The MC2 Model II: Flexible GMRES Elliptic Solver	IOMMU	x86 hardware machine model	unmanned ground vehicle (UGV)robot architecture definition language (RADL)

4. CONCLUSION

Face recognition technology has come a long way in the last twenty years. Today, machines are able to automatically verify identity information for secure transactions, for surveillance and security tasks, and for access control to buildings etc. These applications usually work in controlled environments and recognition algorithms can take advantage of the environmental constraints to obtain high recognition accuracy. However, next generation face recognition systems are going to have widespread application in smart environments -- where computers and machines are more like helpful assistants.

To achieve this goal computers must be able to reliably identify nearby people in a manner that fits naturally within the pattern of normal human interactions. They must not require special interactions and must conform to human intuitions about when recognition is likely. This implies that future smart environments should use the same modalities as humans, and have approximately the same limitations. These goals now appear in reach -- however, substantial research remains to be done in making person recognition technology work reliably like beyond obstacles such as masks/scarf/or anything used to hide face by creating 3D modelling of a subject to predict the closest person with that face features such as jaw like, skull structure and other bone structures, in widely varying conditions using information from single or multiple modalities.

5. REFERENCES

[1] Ross A. An introduction to multi-biometrics. 15th European Signal Processing Conference (EUSIPCO), Poznan, Poland;2020. [2] P. Singh, S. Medida, and K. Duraisamy,

[2] . Sushma J, Sarita SB, Rakesh SJ. recognition and modelling system. Journal of Global Research in Computer Science. 2019;2(7):30-37.

[3] Bolle RM, Connell JH, Pankanti S, Ratha NK, Senior K. A guide to biometrics. European Signal Processing Conference (EUSIPCO), Poznan, Poland; 2020. [4] B. Broll, et al., “A visual programming environment for learning distributed programming,” in Proc. ACM SIGCSE Tech. Symp. Comput. Sci. Educ., 2017, pp. 81–86.

[4] Fagbola TM Olabiyisi SO, Egbetola FI, Oloyede A. Review of technical approaches to face recognition in unconstrained scenes with varying pose and illumination. FUOYE Journal of Engineering and Technology. 2019;2(1).

[5] Draper BA, Baek K, Beveridge JR. Recognizing faces with PCA and ICA. Computer Vision and Image Understanding. 2020;91(1-2):115.

[6]Adedeji OT, Omidiora EO, Olabiyisi SO, Adigun AA. Performance evaluation of optimised PCA and projection combined PCA methods in facial images. Computations & Modelling. 29., 2019.

[7] Aluko JO, Omidiora EO, Adetunji AB, Odeniyi OA. Performance evaluation of selected principal component analysis- based techniques for face image recognition. International Journal of Scientific & Technology Research. 2020; 4(01).

[8] Torres L. Is there any hope for face recognition? In Proc. of the 5th International Workshop on Image Analysis for Multi-media Interactive Services. 2019;21-23.

[9] Lu T, Hou S. A Two-Layered Malware Detection Model Based on Permission for Android," in 2018 IEEE International Conference on Computer and Communication Engineering Technology (CCET). 2019;239-243.

[10] Ageed ZS, Ahmed AM, Omar N, Kak SFIEEE International Conference on Computer and Communication Engineering Technology (CCET). 2019;239-243.

[11] Adeeq MM, Abdulkareem NM, Zeebaree SR, Ahmed DM, Sami AS, Zebari RR. IoT IEEE International Conference on Computer and Communication Engineering Technology (CCET). 2019;239-243.

[12] Zebari IM, Zeebaree SR, Yasin HM. Real Computer and Communication Engineering Technology (CCET). 2019;239-243.

[13] Liu K, Xu S, Xu G, Zhang M, Sun D, Liu H 2019;239-243.

[14] Ibrahim BR, Zeebaree SR, Hussan BK. Performance Measurement for Distributed University of Zakho. 2019;7:65-69.

[15] Zeebaree S, Yasin HM. Arduino based remote controlling for home: power saving. 2020;199-204.

[16] Sadeeq M, Abdulla AI, Abdulraheem AS, (SICN). 2019;109-114

[17] Shibija K, Joseph RV. A Machine Learning Communication and Informatics (ICCCI). 2019.

[18] Lei T, Qin Z, Wang Z, Li Q, Ye D. EveDroid: 2019;6:6668-6680.

[19] Darus FM, Salleh NAA, Ariffin AFM. Android malware detection using machine learning (ICOEI). 2019;1254-1260

[20] Ren Z, Wu H, Ning Q, Hussain I, Chen B End-to-end malware detection for android IoT devices using deep learning," Ad Hoc Networks. 2020;101:102098.

[21] Maniraj S P, Roopa shettigar, Kannadasan B, S.Prabhu,” Artificial Intelligence Application in Human Resource Development”, International Journal of Biology, Pharmacy and Allied Sciences,Vol.10, No.11,2021, pp. 1089-1100, 2021.

[22] Hussain,K., Vanathi,D., Jose,B.K., kAVITHA,S., rANE,b.y., Kaur,H., Sandhya,C., “Internet of Things - Cloud Security Automation Technology Based on Artificial Intelligence”, International Conference on Applied Artificial Intelligence and Computing, ICAAIC 2022; Salem, pp 42-47, 2022.

[23] S.Jagdeeshan, P.Jaisankar, C.Navamani, E.Padma “Steganography Techniques for Data Protection in IOT” volume 29. No 1.2023.