TSABIT
Journal of Computer Science
https://journal.ilmubersama.com/index.php/tsabit

Click here and write your Article Category

Implementation Of The Discrete Sine Transform (Dst) Algorithm For Video File Compression

Muhammad Abdillah Nasution [1], Khairuddin Nasution [2], Mhd. Zulfansyuri Siambaton [3]

[1,2,3] *Department of Informatics Engineering, Faculty of Engineering, Universitas Islam Sumatera Utara, Medan, 20217, North Sumatra, Indonesia*

ARTICLE INFORMATION

Received: February 00, 00
Revised: March 00, 00
Available Online: April 00, 00

KEYWORDS

Video Compression; Discrete Sine Transform (DST); Video Frame; Frequency Domain; PSNR

CORRESPONDENCE

Phone: +6289524271871
E-mail: abdillah.mione@gmail.com

A B S T R A C T

Video is one of the multimedia data formats whose use continues to grow across fields such as education, entertainment, and digital communication, a trend that has increased the demand for storage capacity and bandwidth. A video is composed of a large collection of frames containing substantial amounts of pixel data, which results in comparatively large file sizes. This condition calls for a compression method capable of reducing file size without significantly degrading visual quality. This study implements the Discrete Sine Transform (DST) algorithm in the video file compression process and analyzes its effect on file size and video quality. DST is applied to transform data from the spatial domain into the frequency domain so that data redundancy can be reduced while important information is retained. The compression procedure divides each video into a series of frames and applies the DST transformation to optimize data efficiency. Testing was carried out on video files in MP4 and AVI formats, with compressed video quality evaluated using the Peak Signal to Noise Ratio (PSNR) parameter and compression efficiency measured by comparing file sizes before and after compression. The results show that the DST method is able to reduce video file size while maintaining visual quality. The AVI format achieved an average compression efficiency of 44.96%, while the MP4 format achieved 28.76%. In addition, the PSNR values for all tested videos remained above 37 dB, indicating that the compressed video quality can still be classified as good.

INTRODUCTION

Video is one of the most widely used forms of multimedia data, composed of a sequence of static frames displayed in rapid succession to produce the perception of continuous motion. Because each frame carries a large volume of pixel data, video files tend to be large, particularly for content with high resolution or long duration. Large video file sizes pose a persistent challenge for storage and transmission, since substantial memory capacity and bandwidth are required to handle them. As the volume of video data continues to grow, the need for solutions that reduce file size without significantly compromising video quality becomes increasingly important [1].

One widely applied solution to this problem is video compression. Video compression aims to reduce data redundancy so that file size can be decreased without discarding the essential information contained in the video, thereby making storage and distribution more efficient [2]. A common approach in video compression relies on mathematical transformation, which converts data from the spatial domain into the frequency domain. This approach allows important information to be preserved while less significant data is reduced, making the compression process more effective[3].
The Discrete Sine Transform (DST) is one such transformation method that can be applied to video compression. DST represents data in the frequency domain, which allows the size of the data to be reduced. Building on this principle, the present study implements the DST method for video compression using Python, with the aim of examining its capability to reduce file size without significantly degrading video quality[4].

https://doi.org/10.12345/xxxxx

Based on this background, the research problem addressed in this study concerns how the DST algorithm operates within the video compression process and what results it produces when applied to video files[5]. Accordingly, this study aims to describe the working mechanism of the DST algorithm in video compression and to examine the outcomes obtained when DST is applied for that purpose. The scope of this study is limited to video files in MP4 and AVI formats, with the compression process focused specifically on applying the DST algorithm at the frame level. The system was implemented in Python using OpenCV for video processing, NumPy for numerical operations, SciPy for the DST transformation, PyQt5 for the application interface, Matplotlib for visualizing test results, and FFmpeg for video encoding. Evaluation of the compression results is based on file size, compression ratio, processing time, and the Peak Signal-to-Noise Ratio (PSNR). Audio compression is outside the scope of this study, which concentrates solely on video data.

METHOD

This research was carried out entirely on a personal computing environment rather than at a fixed physical research site; all stages, from data collection to system design, implementation, and testing, were conducted digitally. The study can be classified as a laboratory-based simulation, as the entire research process was conducted within a controlled Python development environment.

Tools and Materials

The hardware used in this study consisted of a laptop or computer with an Intel Core i5 or AMD Ryzen 5 processor, 8 GB of RAM, at least 256 GB of storage, and the Windows 10 operating system. The software environment was built around the Python programming language, supported by several libraries: OpenCV-Python for reading and processing video files, NumPy for matrix operations, SciPy for executing the DST transformation, PyQt5 for building the desktop application interface, Matplotlib for visualizing test results, the subprocess module for invoking FFmpeg from within Python, and the os and time modules for file management and processing-time measurement. FFmpeg itself was used for encoding and saving the compressed video output. The materials used in this research consisted of video files in MP4 and AVI formats, the resulting compressed videos, and supporting scientific literature on video compression and the DST algorithm.

Data Collection Technique

Data collection in this study combined three complementary techniques. A literature study was conducted by reviewing scientific journals, books, articles, and previous research related to video compression, digital image processing, the DST algorithm, and PSNR-based quality testing. An experimental procedure was then carried out by directly testing the developed compression system on several video files with varying formats and resolutions, in order to assess the system's compression capability. Finally, documentation was performed by recording the test results, including file sizes before and after compression, processing time, and PSNR values obtained throughout the study.

System Development Method

System development followed the Waterfall method, a sequential model in which each stage, from requirements analysis through to evaluation, is completed before the next stage begins. This method was chosen because it allows the development of the video compression application to proceed in a systematic and well-structured manner. The stages applied in this study consisted of requirements analysis, system design, system implementation, system testing, and system evaluation.

System Requirements

Functional requirements define the core operations that the system must be able to perform, while non-functional requirements relate to the overall quality and performance of the system. Table 1 and Table 2 summarize the functional and non-functional requirements identified for this study.

Table 1. Functional Requirements

ID	Functional Requirement
F-01	The system is able to read a video file.
F-02	The system is able to extract video frames.
F-03	The system is able to perform the DST transformation.
F-04	The system is able to perform thresholding.
F-05	The system is able to reconstruct video frames.

 https://doi.org/10.12345/xxxxx

F-06	The system is able to save the compressed video.
F-07	The system is able to compute the PSNR value.

Table 2. Non-Functional Requirements

ID	Non-Functional Requirement
NF-01	The system runs on Windows 10.
NF-02	A minimum of 8 GB of RAM is required.
NF-03	The system provides an easy-to-use interface.
NF-04	The system is able to process MP4 and AVI video files.
NF-05	The system runs stably throughout the compression process.

System Design

System design describes the overall workflow of the video compression process, from the initial video input to the production of the compressed video output. The DST-based compression workflow proceeds through the following stages: the process begins when a video file is provided as input; the system reads the video using OpenCV; the video is decomposed into individual image frames; each frame undergoes DST transformation to convert its data from the spatial domain into the frequency domain; thresholding is applied to the resulting frequency coefficients to remove those considered less significant; an inverse DST is then applied to restore the transformed coefficients to frame form; the processed frames are reassembled into a video and encoded using FFmpeg; the PSNR value is computed to assess the quality of the compressed video relative to the original; and the compressed video is then displayed and saved to the application's output folder.

Data handled within the system can be grouped into three categories. Input data consists of video files in MP4 and AVI format, captured at resolutions of 720p and 1080p. Process data includes compression parameters such as block size, DST threshold, frame data, video metadata, and the FFmpeg encoding step. Output data includes the compressed video file, the percentage reduction in file size, the compression processing time, the metadata of the compressed video (format, resolution, duration, and file size), and the PSNR value.

Application Interface and Algorithm Implementation

The application interface was designed around a single main view containing all of the compression system's features, including video input, compression parameter settings, the compression process itself, a preview of both the original and compressed video, the compression results, video quality metrics, a comparison of file sizes, and a system log. The main interface allows the user to select a video file, view its information, configure compression parameters such as block size, DST threshold, and the MP4 CRF value, and run the compression process; it also displays a preview of the original frame alongside the compressed result. A second view presents the compression outcome and system log, showing the file size before and after compression, the compression ratio, the PSNR value, the processing time, and a running log of the compression steps as they occur; from this view, the user can also open the output folder and retrieve the compressed video.

The DST algorithm implementation in this study follows nine sequential stages: reading the input video using OpenCV; extracting the video into individual image frames; converting each frame to grayscale to simplify computation; dividing each frame into smaller blocks to optimize the DST transformation; applying the DST transformation to convert frame data from the spatial domain to the frequency domain; thresholding the resulting DST coefficients to remove those considered less significant; applying the inverse DST to restore the transformed data to frame form; reconstructing the processed frames into a complete video; and encoding the resulting video using FFmpeg. The DST transformation itself was carried out using the SciPy library, while frame reading was handled through OpenCV.

RESULTS AND DISCUSSION

This section presents the results of implementing the DST algorithm for video file compression, along with an analysis of the resulting system performance. The system was implemented in Python with the support of OpenCV for reading video and extracting frames, SciPy for the DST transformation, PyQt5 for the application interface, Matplotlib for visualization, and FFmpeg for encoding the compressed video output. The compression procedure followed a frame-based approach: each input video was decomposed into a sequence of frames, and each frame was processed using the DST transformation to convert its data from the spatial domain into the frequency domain. Following transformation, selected coefficients were reduced through thresholding to decrease data size, after which an inverse DST restored the frames before they were reassembled into the compressed output video. Testing was conducted on eight video files, comprising four videos in MP4 format and four in AVI format, in order to compare compression performance across the two formats. The parameters observed during testing included file size before and after compression, the percentage reduction in file size, processing time, and the PSNR value.

System Implementation

The compression application was implemented on a computer running the Windows operating system, using Python as the primary programming language. The software environment included OpenCV for reading and processing video, NumPy for matrix operations on video frames, SciPy for executing the DST transformation, PyQt5 for building the desktop application interface, Matplotlib for visualizing file-size comparisons, and FFmpeg for encoding the compressed output. On the hardware side, the system was run on a laptop or computer whose specifications were sufficient for video processing, although hardware capability can influence processing time, particularly for high-resolution or long-duration videos.

The compression workflow itself proceeds from video input, through frame reading, DST transformation, thresholding, inverse DST, encoding, and finally to saving the compressed video. The application accepts a video file through a browse function, after which it reads and displays the video's file name, resolution, duration, frame rate, and file size so that the user can review the video's characteristics prior to compression. Once the compression parameters, including block size, DST threshold, and the MP4 CRF value, have been set, the system runs the compression process; for AVI videos, a bitrate setting is used instead to prevent the output file size from increasing. During processing, the system displays a progress bar and a running log, and upon completion presents the original file size, the resulting file size, the compression ratio, the percentage size reduction, the PSNR value, the processing time, and a comparison chart of file sizes.

The system accepts video files in MP4, AVI, MKV, and MOV formats, although testing in this study was limited to MP4 and AVI in line with the research scope. After a file is selected, the system reads its metadata using OpenCV, including file name, resolution, duration, frame rate, and file size, which confirms that the video can be read by the system before compression begins. Once a video has been loaded, the user can configure the block size and DST threshold parameters, which respectively determine how each frame is divided into smaller segments and the coefficient cutoff value applied during thresholding. Each frame is converted into an appropriate color space before the DST transformation is applied to its components, followed by thresholding to remove small frequency coefficients and an inverse DST to restore the frame. Once all frames have been processed, they are reassembled into a video, and FFmpeg performs the final encoding step.

After compression, the system computes the resulting file size and compares it against the original file size to obtain the compression ratio, alongside the PSNR value, which indicates the quality of the compressed video relative to the original. The output includes the original file size, the compressed file size, the compression ratio, the percentage size reduction, the number of frames processed, the PSNR value, and the processing time, all presented in both numerical and graphical form. The compressed video file is automatically saved to an output folder, retaining the format of the original input. A video supplied in MP4 format is compressed to MP4, while a video supplied in AVI format is compressed to AVI, ensuring that the comparison between the two formats remains consistent with the research design. Each output file name includes the suffix “hasil_dst” to indicate that it was produced through the DST-based compression process.

System Testing

System testing was conducted to evaluate the application's ability to compress video using the DST method, focusing on file size before and after compression, the percentage size reduction, the PSNR value, and processing time. The test data consisted of eight videos, comprising four in MP4 format and four in AVI format, captured at resolutions of 720p and 1080p to provide variation in the test results. Table 4 lists the videos used in testing.

Table 4. Test Video Data

No.	Video Name	Format
1	Video 1	MP4
2	Video 2	MP4
3	Video 3	MP4
4	Video 4	MP4
5	Video 1	AVI
6	Video 2	AVI
7	Video 3	AVI
8	Video 4	AVI

Testing was performed by loading each video file into the DST-based compression application. The system read each video using OpenCV and decomposed it into image frames, after which each frame was processed using the DST transformation to convert the video data into the frequency domain. Thresholding was then applied to reduce frequency coefficients considered less significant, followed by an inverse DST to reconstruct the frames before they were reassembled into a video using FFmpeg. Test results were

 https://doi.org/10.12345/xxxxx

obtained by comparing file size before and after compression and by computing the PSNR value to assess compressed video quality; a smaller resulting file size combined with a higher PSNR value was taken to indicate a more favorable compression outcome.
The testing parameters used in this study consisted of file size before compression, file size after compression, compression ratio, processing time, and PSNR. The compression ratio, expressed as a percentage, was calculated using the following formula:

$$Compression\ Ratio = ((Original\ Size - Resulting\ Size) / Original\ Size) \times 100\% \quad (1)$$

where Original Size denotes the file size before compression and Resulting Size denotes the file size after compression. A positive compression ratio indicates a successful reduction in file size, with larger values corresponding to greater size reduction.
All eight test videos were processed using identical compression parameters to ensure an objective comparison: a block size of 16, a DST threshold of 10, a CRF value of 28 for MP4 video, and a bitrate of 300k for AVI video, with PSNR used as the quality parameter throughout. The results were recorded in tabular form and visualized graphically to facilitate comparison.

Compression Test Results

Table 5 presents the file size reduction results obtained from testing the eight videos.

Table 5. Compression Test Results for Eight Videos

No.	Video	Format	Size Before (MB)	Size After (MB)	Reduction (%)
1	AVI_1	AVI	4.76	2.41	49.36
2	AVI_2	AVI	3.81	2.18	42.78
3	AVI_3	AVI	3.23	1.86	42.41
4	AVI_4	AVI	2.40	1.32	45.00
5	MP4_1	MP4	3.47	2.59	25.36
6	MP4_2	MP4	3.01	2.22	26.24
7	MP4_3	MP4	2.97	2.01	32.32
8	MP4_4	MP4	2.81	1.94	30.96

The results in Table 5 show that all eight videos experienced a reduction in file size after compression, regardless of format. Videos in AVI format exhibited a higher degree of size reduction than those in MP4 format: the AVI videos showed reductions ranging from 42.41% to 49.36%, while the MP4 videos ranged from 25.36% to 32.32%. These results indicate that the DST-based compression method is effective for both formats tested. This difference in compression performance can be attributed to the inherent characteristics of each video format. AVI files typically have a larger initial size and a lower degree of built-in compression, which leaves more room for size reduction. MP4 files, by contrast, already incorporate relatively efficient compression techniques, which limits the additional reduction that can be achieved. Consequently, while the DST method was able to compress both formats, it produced more favorable results for AVI than for MP4.

PSNR and Processing Time Results

In addition to file size and compression ratio, the testing also examined PSNR and processing time, the results of which are summarized in Table 6.

Table 6. PSNR and Processing Time Results

No.	Video	Format	PSNR (dB)	Processing Time (s)
1	AVI_1	AVI	38.37	96.30
2	AVI_2	AVI	38.46	74.17
3	AVI_3	AVI	37.95	74.13
4	AVI_4	AVI	39.45	78.15
5	MP4_1	MP4	38.74	101.59
6	MP4_2	MP4	38.01	87.33
7	MP4_3	MP4	39.74	86.86
8	MP4_4	MP4	39.44	84.15

As shown in Table 5, the PSNR values obtained across all compressed videos ranged from 37.95 dB to 39.74 dB, indicating that compressed video quality was well preserved relative to the original, with no substantial perceptual difference. For AVI videos, PSNR values ranged from 37.95 dB to 39.45 dB, with processing times between 74.13 and 96.30 seconds; for MP4 videos, PSNR values ranged from 38.01 dB to 39.74 dB, with processing times between 84.15 and 101.59 seconds. These results indicate that both formats were able to maintain a good level of compressed video quality. Processing time varied across the eight videos, a variation attributable to differences in file size and format. Overall, the DST method was found not only to reduce video file size but also to preserve compressed video quality, as reflected in the relatively high PSNR values obtained throughout testing.

Overall Analysis

Across all eight test videos, the DST-based system was able to compress every file successfully, with all videos showing a reduction in size after processing. The largest size reduction was observed for video AVI_1, at 49.34%, while the smallest was observed for video MP4_1, at 25.41%. This variation indicates that compression effectiveness differs across individual videos, influenced by factors such as initial file size, format, resolution, duration, and visual characteristics. When examined by format, AVI videos consistently showed a higher percentage size reduction than MP4 videos. The average compression efficiency obtained was 44.96% for AVI and 28.76% for MP4, a pattern consistent with the differences in native compression efficiency between the two formats and the underlying encoding characteristics of each. With respect to quality, PSNR values for all tested videos remained above 37 dB, indicating that compressed video quality stayed within a good range and that the visual distortion introduced by compression was comparatively minor. Taken together, these results indicate that the Discrete Sine Transform method is capable of reducing video file size without significantly degrading visual quality, supporting its applicability as a compression technique for digital video files.

CONCLUSION

Based on the implementation and testing carried out in this study, several conclusions can be drawn. The Discrete Sine Transform (DST) algorithm can be successfully implemented for video file compression using the Python programming language, supported by the OpenCV, SciPy, NumPy, PyQt5, Matplotlib, and FFmpeg libraries. The compression process proceeds through video frame extraction, DST transformation, thresholding, inverse DST, frame reconstruction, and encoding of the compressed video. Testing results show that the DST method is capable of reducing video file size while maintaining visual quality. The AVI format achieved an average compression efficiency of 44.96%, compared to 28.76% for the MP4 format, and the PSNR values for all tested videos remained above 37 dB, indicating that compressed video quality can still be classified as good. These findings confirm that the implementation of the DST algorithm for video file compression was successful and consistent with the objectives of this study. For future research, subsequent studies could examine additional video formats, such as MKV and MOV, and incorporate higher video resolutions to obtain more varied test results. Future work could also compare the DST method against other transformation techniques, such as the Discrete Cosine Transform (DCT) and the Discrete Wavelet Transform (DWT), to identify approaches that may yield more optimal compression outcomes.

REFERENCES

Journal Article

[1] Shao, L., Zhang, Y., & Chen, H. (2024). Multimedia Video Storage Optimization Using Compression Techniques. Journal of Multimedia Technology, 18(1), 33–41.

[2] Khalid, M., Rahman, A., & Yusuf, M. (2024). Video Compression Techniques for Multimedia Data Transmission. International Journal of Computer Applications, 185(4), 15–22.

[3] Salomon, D., & Motta, G. (2023). Handbook of Data Compression (6th ed.). Springer.

[4] Richardson, I. (2023). The H.264 Advanced Video Compression Standard. Wiley.

[5] Sayood, K. (2022). Introduction to Data Compression (6th ed.). Morgan Kaufmann.

[6] Gonzalez, R. C., & Woods, R. E. (2022). Digital Image Processing (4th ed.). Pearson Education.

[7] Britanak, V., Rao, K. R., & Yip, P. (2007). Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations. Academic Press.

[8] Rao, K. R. (2017). Discrete Cosine Transform: Algorithms, Advantages, Applications. Academic Press.

[9] Rahman, M., & Islam, S. (2022). Image Compression Using Discrete Wavelet Transform. International Journal of Image Processing, 11(3), 55–63.

[10] Ahmed, N., & Khan, R. (2023). Digital Multimedia Processing Using Transform Methods. Journal of Multimedia Systems, 15(2), 120–128.

[11] Zhang, Y., Liu, H., & Wang, X. (2023). Wavelet-Based Video Compression for Digital Multimedia Systems. Journal of Visual Communication and Image Representation, 91, 103745.

[12] Saputra, R., & Wijaya, A. (2023). Implementasi Discrete Sine Transform pada Pengolahan Video Digital. Jurnal Teknologi Informasi dan Multimedia, 8(2), 77–85.

[13] Sari, I.P., Zulherry, A., Basri, M., & Hayani W. (2025). Pembelajaran Pemrograman berbasis Machine Learning sebagai Upaya Peningkatan Computational Thinking. Jurnal Penelitian, Pendidikan dan Pengajaran: JPPP 6 (3), 245-250

[14] Sari, I.P., & Batubara, I.H. (2021). User Interface Information System for Using Account Services (Joint Account) WEB-Based. International Journal of Economic, Technology and Social Sciences (Injects) 2 (2), 462-469
[15] Ramadhani, F., & Sari, I.P. (2021). Pemanfaatan Aplikasi Online dalam Digitalisasi Pasar Tradisional di Medan. Prosiding Seminar Nasional Kewirausahaan 2 (1), 806-811
[16] Sari, I.P., & Alfarisi, F. (2024). Perancangan Sistem Aplikasi Pendataan Membership Gym Menggunakan Metode Unified Software Development Process (USDP) Berbasis Web. Hello World Jurnal Ilmu Komputer 3 (1), 37-48
[17] Sari, I.P., Jannah, A., Meuraxa, A.M., Syahfitri, A., & Omar, R. (2022). Perancangan Sistem Informasi Penginputan Database Mahasiswa Berbasis Web. Hello World Jurnal Ilmu Komputer 1 (2), 106-110
[18] Satria, A., Ramadhani, F., & Sari, I.P. (2023). Rancang Bangun Sistem Informasi Penerimaan Peserta Didik Baru (PPDB) Sekolah Menengah Kejuruan Telkom 2 Medan Menggunakan Codeigniter. Wahana Jurnal Pengabdian kepada Masyarakat 2 (1), 23-31
[19] Sari, I.P., Azzahrah, A., Qathrunada, I.F., Lubis, N., & Anggraini, T. (2022). Perancangan sistem absensi pegawai kantoran secara online pada website berbasis HTML dan CSS. Blend sains jurnal teknik 1 (1), 8-15
[20] Hariani, P.P., Sari, I.P., & Batubara, I.H. (2021). Android-Based Financial Statement Presentation Model. JURNAL TARBIYAH 28 (2), 1-16
[21] Sari, I.P., Syahputra, A., Zaky, N., Sibuea, R.U., & Zakhir, Z. (2022). Perancangan sistem aplikasi penjualan dan layanan jasa laundry sepatu berbasis website. Blend sains jurnal teknik 1 (1), 31-37
[22] Sari, I.P., Batubara, I.H., Ramadhani, F., & Wardani, S. (2022). Perancangan Sistem Antrian pada Wahana Hiburan dengan Metode First In First Out (FIFO). Sudo Jurnal Teknik Informatika 1 (3), 116-123
[23] Ramadhani, F., Satria, A., & Sari, I.P. (2022). Aplikasi internet berbasis website sebagai E-Commerce penjualan komponen sport car. Blend Sains Jurnal Teknik 1 (2), 69-75
[24] Sari, I.P., Ramadhani, F., Satria, A., Apdilah, D., & Basri, M. (2023). Rancangan UI/UX Aplikasi Analytics pada Toko Online Wao Sneakers Menggunakan Figma Berbasis Mobile. Factory Jurnal Industri, Manajemen dan Rekayasa Sistem Industri 1 (3), 93-101
[25] Sari, I.P., Al-Khowarizmi, A., & Batubara, I.H. (2021). Cluster Analysis Using K-Means Algorithm and Fuzzy C-Means Clustering For Grouping Students' Abilities In Online Learning Process. Journal of Computer Science, Information Technology and Telecommunication Engineering 2 (1), 139-144
[26] Hutasuhut, B.K., Sari, I.P., & Al-Khowarizmi, A. (2023). Analysis the Effect of Digitalization and Technology on Web-Based Entrepreneurship. Journal of Computer Science, Information Technology and Telecommunication Engineering 4 (1), 350-354
[27] Sitompul, D.N., Rahmatika, A., & Sari, I.P. (2023). Application of The Sales and Purchase Program Using The Rapid Application Development Model. Al'adzkiya International of Computer Science and Information Technology (AIoCSIT) Journal 4 (1), 6-16
[28] Sari, I.P., & Batubara, I.H. (2021). Perancangan Sistem Informasi Laporan Keuangan Pada Apotek Menggunakan Algoritma K-NN. Seminar Nasional Teknologi Edukasi dan Humaniora (SiNTESa) (1).
[29] Ramadhani, F., Satria, A., & Sari, I.P. (2023). Implementasi Metode Fuzzy K-Nearest Neighbor dalam Klasifikasi Penyakit Demam Berdarah. Hello World Jurnal Ilmu Komputer 2 (2), 58-62
[30] Sari, I.P., Batubara, I.H., & Basri, M. (2022). Implementasi Internet of Things Berbasis Website dalam Pemesanan Jasa Rumah Service Teknisi Komputer dan Jaringan Komputer. Blend Sains Jurnal Teknik 1 (2), 157-163
[31] Sari, I.P., & Ramadhani, F. (2021). Pengaruh Teknologi Informasi Terhadap Kewirausahaan Pada Aplikasi Perancangan Jual Beli Jamu Berbasis WEB. Prosiding Seminar Nasional Kewirausahaan 2 (1), 874-878
[32] Ichsan, A., Zulherry, A., Lubis, T. A., & Shahnaz, B. A. Z. (2025). Utilization of Mobile Applications to Speed Up The Search for Android-Based Index Places. IJATCoS: Indonesian Journal of Applied Technology. Computer and Science, 2(1).
[33] Amada, E.P., & Zulherry, A. (2025). Klasterisasi Minat Dan Bakat Siswa Menggunakan Metode X-Means Berbasis Web: Studi Kasus SMA Negeri 1 Hamparan Perak. sudo Jurnal Teknik Informatika 4 (4), 276-283.
[34] Zulherry, A., Ramadhani, F., & Satria, A. (2024). Klasifikasi Data Tracer Study Dengan Pemanfaatan Data Mining Menggunakan Algoritma Support Vector Machine dan Neural Network. Portal Riset dan Inovasi Sistem Perangkat Lunak 2 (1), 45-54
[35] Zulherry, A. (2023). Decision making for network security with simple additive weighting method. Journal of Intelligent Decision Support System (IDSS), 6(3), 155-159.
[36] Basri, M., & Zulherry, A. (2025). Analysis of the Impact of Gambling and Online Loans in the Perspective of Informatics, Islam, and Kemuhammadiyahan. Ar-Rasyid: Jurnal Pendidikan Agama Islam, 5(1), 65-73.
[37] Asadel, A., & Zulherry, A. (2025). Detecting Zero-Width Characters Obfuscated in Phishing URLs using the XGBOOST Algorithm. Hanif Journal of Information Systems 3 (1), 43-53

[38] Zulherry, A., Sari, I.P., & Basri, M. (2025). Perancangan Aplikasi Monitoring Kehadiran Pegawai Menggunakan RFID. sudo Jurnal Teknik Informatika 4 (4), 378-384

[39] Zulherry, A., & Gunawan, M. (2025). Development of an Android-Based Smart Health Monitoring Device for Heartbeat Detection. Al'adzkiya International of Computer Science and Information Technology (AIoCSIT) Journal, 6(2), 75-79.

[40] Sari, I.P., Zulherry, A., Basri, M., & Hayani W. (2025). Pembelajaran Pemrograman berbasis Machine Learning sebagai Upaya Peningkatan Computational Thinking. Jurnal Penelitian, Pendidikan dan Pengajaran: JPPP 6 (3), 245-250

[41] Vianingrum, M. H., & Zulherry, A. (2026). Implementasi Sistem Pendukung Keputusan Untuk Menganalisis Tingkat Kepuasan Pengguna E-Learning Menggunakan Metode End User Computing Satisfaction (EUCS) Di SMK Multi Karya. Hello World Jurnal Ilmu Komputer, 5(1), 77-84.

[42] Lubis, I. Q., & Zulherry, A. (2026). Implementasi Keamanan Website Dari Serangan Cross Site Request Forgery Menggunakan Algoritma HMAC-SHA256 Pada Framework Laravel. Hello World Jurnal Ilmu Komputer, 5(1), 47-58.

[43] Zulherry, A., Sari, I. P., & Basri, M. (2026). Perancangan dan Implementasi Segmentasi LAN pada Infrastruktur Jaringan Skala Menengah. Hello World Jurnal Ilmu Komputer, 4(4).

Book

[44] Indah Purnama Sari. Algoritma dan Pemrograman. Medan: UMSU Press, 2023, pp. 290.

[45] Janner Simarmata Arsan Kumala Jaya, Syarifah Fitrah Ramadhani, Niel Ananto, Abdul Karim, Betrisandi, Muhammad Ilham Alhari, Cucut Susanto, Suardinata, Indah Purnama Sari, Edson Yahuda Putra. Komputer dan Masyarakat. Medan: Yayasan Kita Menulis, 2024, pp.162.

[46] Mahdianta Pandia, Indah Purnama Sari, Alexander Wirapraja Fergie Joanda Kaunang, Syarifah Fitrah Ramadhani Stenly Richard Pungus, Sudirman, Suardinata Jimmy Herawan Moedjahedy, Elly Warni, Debby Erce Sondakh. Pengantar Bahasa Pemrograman Python. Medan : Yayasan Kita Menulis, 2024, pp.180

[47] Zelvi Gustiana Arif Dwinanto, Indah Purnama Sari, Janner Simarmata Mahdianta Pandia, Supriadi Syam, Semmy Wellem Taju Fitrah Eka Susilawati, Asmah Akhriana, Rolly Junius Lontaan Fergie Joanda Kaunang. Perkembangan Teknologi Informatika. Medan: Yayasan Kita Menulis, 2024, pp.158

[48] Indah Purnama Sari. Buku Ajar Pemrograman Internet Dasar. Medan: UMSU Press, 2022, pp. 300.

[49] Indah Purnama Sari. Buku Ajar Rekayasa Perangkat Lunak. Medan: UMSU Press, 2021, pp. 228.

[50] Purba, A. M. V., Pradipta, R., Zulherry, A., Sari, I. P., Ansya, Y. A. U., Basri, M., ... & Darnilasari, A. (2026). Pengetahuan Dasar Komputer. Yayasan Kita Menulis.

[51] Irfan, A. P., Susanto, C., Sari, I. P., Zulherry, A., Ansya, Y. A. U., Dewi, I. K., ... & Mubarak, R. (2026). Sistem Informasi Modern untuk Era Digital. Yayasan Kita Menulis.

[52] Kaban, R., Rachman, A. I., Sari, I. P., Zulherry, A., Basri, M., Fadilah, N., ... & Ahyuna, A. (2026). Pengantar Artificial Intelligence, Machine Learning, dan Big Data. Yayasan Kita Menulis.