Journal of Applied Informatics and Computing (JAIC)
Vol.7, No.2, December 2023, pp. 128~135
e-ISSN: 2548-6861

Real-Time Visitor Counting with Dynamic Facial Recognition using Python and Machine Learning

I Made Bhaskara Gautama [1]*, I Gusti Ngurah Wikranta Arsa [2]*, I Made Arya Budhi Saputra [3]*, IGKG Puritan Wijaya ADH [4]*, Dewa Gede Yudisena Nanda Sutha [5]*
* Fakultas Informatika dan Komputer, Institut Teknologi dan Bisnis STIKOM Bali
bhaskara@stikom-bali.ac.id [1], arsa@stikom-bali.ac.id [2], aryabudhi@stikom-bali.ac.id [3], puri@stikom-bali.ac.id [4], 200040084@stikom-bali.ac.id[5]

Article Info

Article history:

Received 2023-08-31
Revised 2023-09-15
Accepted 2023-09-26

Keyword:

Computer Vision,
Face Recognition,
Machine Learning,
Visitor Counter.

ABSTRACT

Visitor data or the number of visitors at a particular location is crucial information to be obtained. This data can serve various purposes, particularly in enhancing customer satisfaction. For instance, predicting the number of visitors at tourist destinations enables tourism management to be better prepared for welcoming and providing optimal services to arriving visitors. Visitor count data can also be employed to automatically restrict visitors during the COVID-19 pandemic, ensuring a safe and comfortable environment with limited attendees. To acquire visitor data, a system capable of accurate visitor detection is required. This research utilizes computer vision to detect visitor faces. The developed system, programmed in Python, functions by detecting visitor faces and conducting a count based on the detected faces. To prevent the same visitor from being detected multiple times, a facial recognition method with dynamic facial data collection is implemented in this study. The constructed system successfully counted 27 out of 28 visitors over two days. However, the system has limitations, particularly in terms of the restricted detection area. Therefore, a physical mechanism mandating visitors to undergo facial scanning and registration needs to be established, ensuring recorded data corresponds to the actual visitor count.

I. INTRODUCTION

Visitor data or footfall is crucial for various types of establishments, whether they are profit-oriented or non-profit. This data is collected for specific purposes. There are various methods that utilize technology to gather visitor data. One commonly employed approach is by leveraging embedded systems based on the Internet of Things (IoT) [1]–[4], and another involves computer vision, specifically facial detection or object detection [5], [6]. Both methods have their own strengths and weaknesses. Embedded systems require relatively low costs, making them suitable for large-scale production. However, constructing an embedded system capable of accurately detecting and counting visitors often necessitates the use of multiple sensors (sensor fusion), intricate environmental adjustments to ensure sensor performance, and a trial-and-error process contingent on the visitor detection location. More accurate detection and counting can be achieved through facial detection. Nevertheless, this method demands a relatively higher investment due to the requirement for cameras and powerful computational devices.

Visitor data collected using more accurate methods can be leveraged in the field of data science. One of the places that can benefit from this data is establishments that serve visitors. Locations prioritizing service quality rely on this data to enhance visitor satisfaction. For instance, predicting the future number of visitors to tourist destinations allows tourism management to be better prepared to welcome visitors and deliver optimal services [7]. During the COVID-19 pandemic, real-time visitor data is also utilized to limit the number of visitors in a particular location [8]. This approach offers comfort to visitors during the pandemic, ensuring that an excessive number of visitors is not present at the location.

Accurate visitor data, along with additional supplementary data, can also be employed to regulate room temperature automatically [9]. This contributes to maintaining comfort for visitors or individuals within the premises.

The issue stemming from this situation is the necessity for accurate visitor detection. Facial detection methods utilizing machine learning or deep learning indeed exhibit high accuracy in detecting faces. Nevertheless, several aspects need to be adapted according to real-world conditions. For instance, what if the same person's face is detected multiple times? This could potentially impact the accuracy of visitor counts, as there is a possibility that the same individual might be counted as more than one visitor.

Therefore, this research is carried out by implementing facial recognition. Facial recognition is commonly implemented in attendance systems, monitoring systems, or access systems. In attendance systems, initial registration is required to determine which individuals' presence needs to be recorded. When the system is operational, it matches the faces captured by the camera with the registered facial data [10]–[12]. This principle applies to monitoring systems and access systems as well. Prior registration is essential to identify the faces that are being tracked or granted access [13], [14]. The implementation of facial recognition for visitor detection has been done before. This system was used to detect the faces of library visitors. However, registration is also required in this study because the system's operation is similar to an attendance system [15]. In this study, no preliminary registration is required as the identity of the visitors is not necessary. Facial data used for matching is dynamically collected as the system operates. This approach is adopted to ensure the accuracy of visitor counts and to avoid the same face being counted as multiple visitors within a single day.

The facial recognition conducted in this study employs machine learning. The method utilized consists of several stages, including face detection, posing and projecting the face, face encoding, and face verification. In the face detection stage, the algorithm employed is the Histogram of Oriented Gradients (HOG) algorithm. This algorithm works fast and has a higher accuracy compared to the Viola-Jones algorithm [16], [17]. In the process of posing and projecting the face, multiple stages are involved. These stages encompass face area detection, determination of face landmarks, and transforming the face to directly face forward. The face encoding stage employs a pre-trained Deep Convolutional Neural Network model. The outcome of this process is a set of 128 measurements associated with a facial image. Subsequently, this encoding result is matched in the face verification stage. The developed system is also capable of capturing unrecognized faces, which are then used as facial data for subsequent matching. Additionally, in cases where the same face is detected on the same day, the counted number of visitors remains as one visitor.

II. METHOD

An overview of the system is depicted in Figure 1.

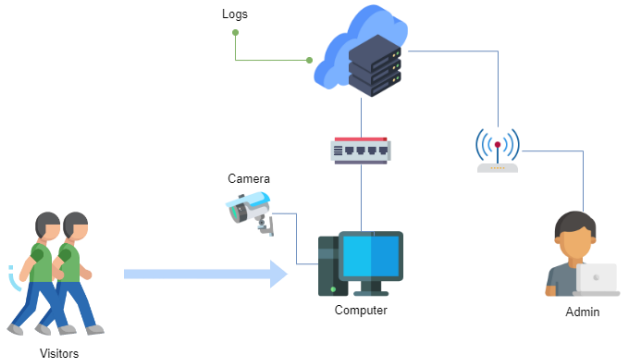

Figure 1. System Overview

The faces of entering visitors are captured using a camera affixed to a computer. The facial recognition system operates locally on this computer, a design chosen to enable direct hardware access to the camera. For each successfully recorded visitor's face, data in the form of facial image files and logs are documented and uploaded to a web server. The logs on the web server are stored in a MySQL database. Storing files and logs on the web server serves the purpose of facilitating monitoring processes, which can be administered by system administrators from anywhere with an internet connection. Furthermore, administrators can generate visitor recording reports based on the logs stored on the web server.

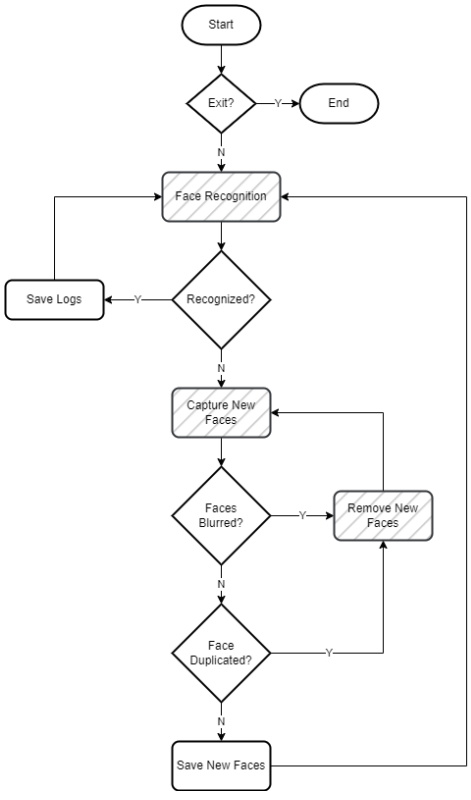

Figure 2. Flowchart of the Developed System

The general algorithm of the system developed in this research is illustrated in Figure 2. The developed system

begins with the facial recognition process. If a face is successfully recognized, a new visit log is stored in the database. If the detected face is unknown to the system, the process of capturing a new face is initiated. Due to the real-time nature of facial recognition and the limited computational resources available, instances of duplicated, unclear, or even non-classifiable facial image captures can occur. Consequently, before storing newly captured facial images, the validity of each image is examined using two categories: blurred images and duplicated faces.

Broadly speaking, three primary processes are present in this sequence: Face Recognition, Capture New Faces, and Remove Faces, denoted by distinct process symbols in the flowchart. Furthermore, a web server application has been developed to receive log storage requests from the facial recognition application. This application is constructed using the HTML, CSS, JavaScript, and PHP programming languages. Log data is stored on the web server using the relational database management system MySQL. These four processes are elaborated in greater detail in the subsequent section.

A. Face Recognition

Facial recognition is executed through several stages, as indicated in Figure 3.

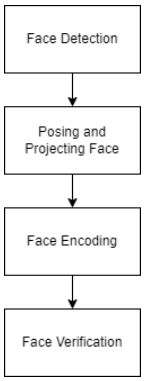

Figure 3. Face Recognition Process

In the stage of face detection and feature extraction based on the frames captured by the camera, the method employed is Histogram of Oriented Gradients (HOG) [18]. This method captures the distribution of gradient directions in an image. It's used for object detection by analyzing how intensity changes from pixel to pixel and grouping these changes into histograms. These histograms help describe object shapes and structures, making HOG effective for detecting objects with distinct outlines. This method was tested using 13,233 facial data obtained from Labeled Faces in the Wild [19]. A total of 13,176 faces were successfully detected, achieving an accuracy of 99.57%. Figure 4, Figure 5, and Figure 6 are created using the Python programming language by utilizing facial data available on Kaggle [20]. Figure 4 shows the application of HOG.

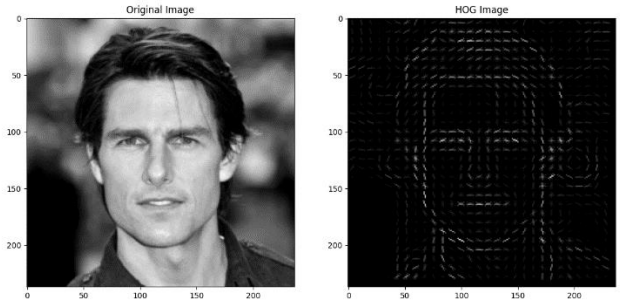

Figure 4. Histogram of Oriented Gradients (HOG)

To obtain the facial region, the HOG image in Figure 4 is matched against HOG images generated from thousands of facial images provided by the dlib library in Python. Successfully detected faces may not always be perfectly oriented to face forward, which can pose challenges and reduce accuracy during face matching. Consequently, a scheme is required to adjust the images, aligning features such as eyes, nose, mouth, and others to the same positions during matching.

Before alignment, face landmark estimation is utilized to determine the positions of the eyes, nose, mouth, and facial contour [21]. This method involves determining 68 points on the face that mark these areas. Lines connecting these landmark points can be drawn to create an image like the one shown in Figure 5.

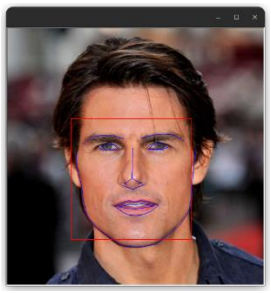

Figure 5. Face Landmark

The blue lines indicate the facial area created using face landmark estimation. Next, the affine transformation method is employed to maintain this area's position during the matching process [22]. Figure 6 shows the resulting facial appearance, prepared and ready for the encoding process.

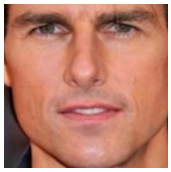

Figure 6. Face for Encoding

The face encoding stage is conducted using a pre-trained Deep Convolutional Neural Network model provided by OpenFace [23]. For each facial image, a set of 128 measurement values is generated, indicating the

characteristics of that face. Figure 7 shows the measurement result obtained from a facial image.

```
faceRecognition.py > ...
1  import face_recognition
2  import matplotlib.pyplot as plt
3
4  # Load images
5  image_of_person_1 = face_recognition.load_image_file("tom.jpg")
6

PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL

bhaskara@BG-PC:~/PythonProject/faceRegDet$ python3 faceRecognition.py
[array([-0.08181055,  0.17689188,  0.04059437, -0.05406533, -0.17733683,
        0.07377131, -0.04403474, -0.05197026,  0.25006151, -0.12149715,
        0.26234666,  0.07976352, -0.15773514, -0.04598662, -0.09485552,
        0.06457629, -0.16015647, -0.10873017, -0.10853373, -0.14991012,
        0.01675667,  0.05635317, -0.05219109,  0.04493024, -0.2584365 ,
       -0.17724115,  0.00544458, -0.02622311,  0.0841393 , -0.19136333,
       -0.00275828,  0.09052493, -0.2260789 , -0.03498367, -0.01726354,
        0.11164017, -0.0875763 , -0.02520141,  0.18256691,  0.00281294,
       -0.11715262,  0.09640108,  0.00521718,  0.27460417,  0.20258701,
        0.02053064, -0.02323381, -0.1010858 ,  0.17647511, -0.26279941,
        0.03388709,  0.23314944,  0.06950737,  0.07073839,  0.08136982,
       -0.21115407, -0.01253188,  0.11964408, -0.06462648,  0.02505679,
        0.01303283, -0.07253224, -0.02413128, -0.07626289,  0.1136591 ,
        0.04967417, -0.05034201, -0.13554083,  0.10447024, -0.14348173,
       -0.0312951 ,  0.21934326, -0.03509613, -0.19468392, -0.17959033,
        0.08769462,  0.4135423 ,  0.23687749, -0.16547242,  0.01598401,
       -0.05855237, -0.04160954,  0.06985941, -0.01871304, -0.11977957,
       -0.05683525, -0.12258612,  0.09589548,  0.2533789 ,  0.04271796,
       -0.01625546,  0.22797023,  0.09500023, -0.01319232,  0.00297006,
       -0.02439235, -0.13699174,  0.02568819, -0.07368087, -0.02644492,
        0.09304838, -0.07800364, -0.03128002,  0.0677465 , -0.19000074,
        0.21113327, -0.01393091, -0.01302545, -0.03924732, -0.03570021,
       -0.10098077,  0.08387085,  0.21474308, -0.29768026,  0.23916888,
        0.11091112,  0.00949849,  0.18295491,  0.13745938,  0.04955611,
       -0.0250951 , -0.05883642, -0.05906203, -0.13531777,  0.03214902,
       -0.14485627,  0.13948405, -0.0276754 ])]
```

Figure 7. Face Encoding Results

In the Face Verification stage, the process of matching the captured facial image within a frame with the stored facial images takes place. This matching process involves comparing the measurement outcomes from the face encoding stage with the measurements of all stored facial images. These measurement results are processed using Linear Support Vector Machine Classification [24]. If the measurement results of the facial image in the frame exhibit similarity or correspondence with the stored facial data, the system provides feedback in the form of the filename of the matching stored face. The face matching process successfully recognized 11,494 faces correctly out of the 13,233 available facial images, achieving an accuracy of 87%.

B. Capture New Faces

This process is executed when there is no stored facial image that matches the facial image within the camera frame (refer to the flowchart in Figure 2). This mechanism renders the developed system dynamic, as manual registration is not required for enrolling new faces. This approach is feasible because the facial matching process is not intended for verifying individuals' identities, but solely for distinguishing one visitor from another. The necessity to identify distinct visitors aims to establish an accurate visitor count system.

To obtain new visitor faces, the Viola-Jones algorithm is employed for face detection. This choice is made due to the Viola-Jones algorithm having a quicker computational time compared to HOG, even though it offers lower accuracy [25]. When capturing visitor faces, extremely high accuracy in face detection is not required; instead, speed takes precedence. A lower face detection accuracy suffices to ensure that the captured image truly represents a frontal, clear facial view.

The method used for capturing new faces is described in Figure 8.

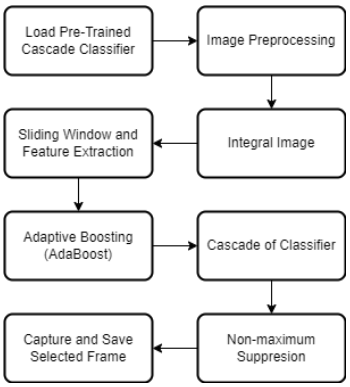

Figure 8. Method to Capture Face

The pre-trained model utilized is provided by OpenCV. During image preprocessing, the camera frame is transformed from RGB to grayscale, as color features are unnecessary for subsequent processes. This grayscale image is then converted into an integral image or what's known as a summed-area table. The result is the sum of pixel values within a rectangular region of an image. This is generated from the original image through cumulative pixel value summation. Each value within the integral image signifies the cumulative sum of pixel values above and to the left of its corresponding pixel in the original image.

The Haar Cascade classifier operates via a sliding window technique. A rectangular window of varying sizes traverses the integral image. Within each window, a series of Haar-like features are computed. These Haar-like features consist of simple rectangular patterns that capture intensity variations. These features represent contrasts between adjacent areas of the image and encompass characteristics like edge features, line features, and center-surround features. An illustration of a Haar-like feature is depicted in Figure 9 [26].

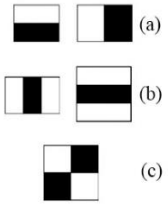

Figure 9. Haar-like Feature

The Haar Cascade classifier uses a boosting algorithm called Adaptive Boosting or AdaBoost [27]. AdaBoost combines a set of weak classifiers (individual Haar-like features) into a strong classifier. It assigns higher weights to misclassified samples, allowing subsequent weak classifiers to focus on the hard-to-classified examples. The Haar Cascade classifier employs a cascade of classifiers to efficiently reject non-face regions. This cascade consists of

multiple stages, each with a set of weak classifiers. Non-face regions can be quickly rejected in the early stages, while more computation is spent on potential face regions. To eliminate duplicate detections, a non-maximum suppression step is applied. Overlapping detections are pruned by retaining only the one with the highest confidence score. This ensures that each detected face is only captured once.

C. *Remove Faces*

This process is an additional step required due to limitations in the specifications of the utilized devices. To ensure that the stored facial images are valid, meaning they are indeed facial images and clear, each segment of the facial image to be stored undergoes a preliminary check. There are two conditions to determine whether a facial image segment should be retained or discarded.

The first condition involves checking whether the image segment is a facial image and whether the face to be stored already exists. To examine facial images, the same method used in the face recognition section is employed. The second condition involves assessing the clarity of the facial image segment. This assessment is conducted using the Laplacian operator [28]. The Laplacian operator is employed to detect image blur. It identifies edges and details within an image by calculating the second derivative of the image intensity concerning its position. When used to detect blur, the Laplacian operator indirectly emphasizes areas with low high-frequency components, which typically indicate sharp edges and intricate details. Applied to a blurred image, the Laplacian response is reduced in regions affected by blur. Based on trial and error, the threshold used to determine whether an image is blurry or not is set at 50.

D. *Web Server Application*

The application built on the remote web server is utilized to manage log data and facial data sent by the locally running facial recognition application. Data is transmitted using the HTTP protocol, which is handled by PHP on the web server. Log data is stored using the relational database management system MySQL, while facial images are stored as JPG files on the web server.

Log data is stored within a table in the database with the structure shown in Figure 10.

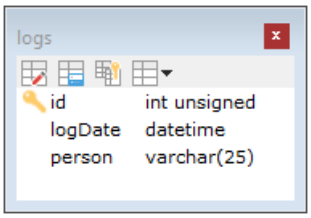

Figure 10. Logs Data Structure

The constructed website encompasses several features, including a login mechanism, a dashboard page, a history section, and a page for generating reports.

III. RESULTS AND DISCUSSION

The system developed was tested at the library of Institut Teknologi dan Bisnis STIKOM Bali. The library has a single entrance for visitors. The visit procedure in the library requires new visitors to input their membership ID on a computer located near the entrance to record their visit. The developed system was placed on the same computer, resulting in two records for entering visitors: (1) Based on facial recognition; (2) Based on membership ID input. Both sets of data were used and matched to assess the performance of the facial recognition system. The testing was conducted over two days from August 14th to August 15th, 2023. Figure 11 shows a diagram illustrating the testing scheme conducted.

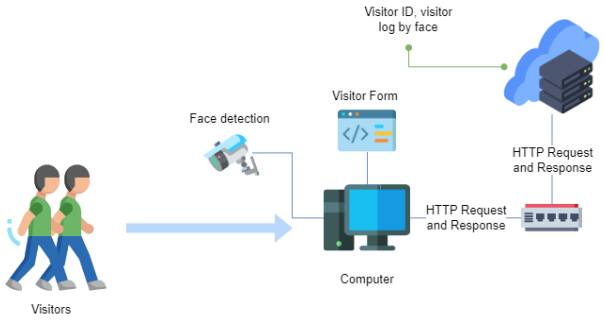

Figure 11. Testing Scheme

A. *Facial Recognition Data*

Figure 12 shows facial capture results from the developed system. A pixelate (mosaic) filter has been applied to the screenshots to preserve visitors' privacy. These locally captured files are periodically uploaded to the web server using triggers whenever a new face is added.

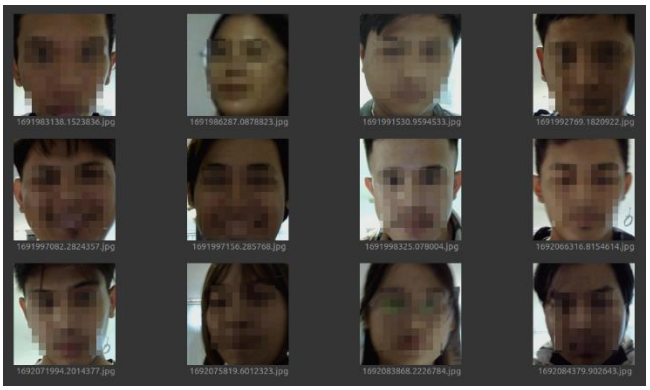

Figure 12. Visitor Facial Data

The file naming system for newly stored faces is automatically generated using sequential numbering and a timestamp to prevent duplicate filenames.

Figure 13 shows the logs stored on the web server. These logs are created based on the data sent by the facial recognition application. The display in the image represents the history log view within the developed web application.

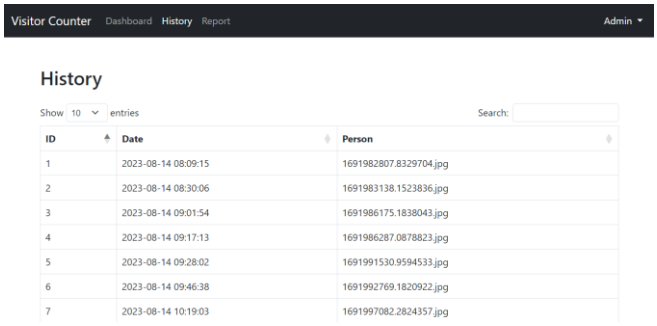

Figure 13. Log History

B. Visitor Visit Form Data

Table I contains data collected from the visitor visit form filled out by visitors on the computer. Certain data details have been concealed to safeguard data privacy.

TABLE I
VISITOR FORM DATA

visitor_id	member_id	institution	checkin_date
57672	2000***48	STIKOM Bali	8/14/2023 8:09
57673	2000***80	STIKOM Bali	8/14/2023 8:30
57674	1900***99	STIKOM Bali	8/14/2023 9:01
57675	1900***28	STIKOM Bali	8/14/2023 9:17
57676	2200***37	STIKOM Bali	8/14/2023 9:28
57677	2000***64	STIKOM Bali	8/14/2023 9:46
57678	2000***52	STIKOM Bali	8/14/2023 10:19
57679	2000***26	STIKOM Bali	8/14/2023 10:50
57680	2000***28	STIKOM Bali	8/14/2023 12:16
57681	2100***77	STIKOM Bali	8/14/2023 12:17
57682	1700***28	STIKOM Bali	8/14/2023 12:34
57683	2000***66	STIKOM Bali	8/14/2023 12:59
57688	2200***39	STIKOM Bali	8/15/2023 7:00
57689	2200***89	STIKOM Bali	8/15/2023 7:01
57690	2000***88	STIKOM Bali	8/15/2023 7:28
57691	2000***47	STIKOM Bali	8/15/2023 9:25
57692	2000***29	STIKOM Bali	8/15/2023 9:25
57693	2000***95	STIKOM Bali	8/15/2023 10:59
57694	2000***48	STIKOM Bali	8/15/2023 14:25
57695	2000***95	STIKOM Bali	8/15/2023 15:36
57696	2000***93	STIKOM Bali	8/15/2023 16:30
57697	1900***99	STIKOM Bali	8/15/2023 16:31

C. Data Comparison

On the first day, August 14th, 2023, 12 visitors entered the library. The stored logs based on facial recognition and the data from the visit form both indicate the same number, 12 visitors. The recorded time difference between the two sets of data is less than one minute. Table II shows a comparison between the two datasets.

TABLE II
DATA COMPARISON ON THE FIRST DAY OF TESTING

Visitor Form Data		Log History		
Institution	**Check-in Date**	**ID**	**Log Date**	**File Name**
STIKOM Bali	8/14/2023 8:09	1	8/14/2023 8:09	1691982807.8329704.jpg
STIKOM Bali	8/14/2023 8:30	2	8/14/2023 8:30	1691983138.1523836.jpg
STIKOM Bali	8/14/2023 9:01	3	8/14/2023 9:01	1691986175.1838043.jpg
STIKOM Bali	8/14/2023 9:17	4	8/14/2023 9:17	1691986287.0878823.jpg
STIKOM Bali	8/14/2023 9:28	5	8/14/2023 9:28	1691991530.9594533.jpg
STIKOM Bali	8/14/2023 9:46	6	8/14/2023 9:46	1691992769.1820922.jpg
STIKOM Bali	8/14/2023 10:19	7	8/14/2023 10:19	1691997082.2824357.jpg
STIKOM Bali	8/14/2023 10:50	8	8/14/2023 10:50	1691997156.285768.jpg
STIKOM Bali	8/14/2023 12:16	9	8/14/2023 12:16	1691998325.078004.jpg
STIKOM Bali	8/14/2023 12:17	10	8/14/2023 12:17	1691999314.7957132.jpg
STIKOM Bali	8/14/2023 12:34	11	8/14/2023 12:34	1691986288.4124672.jpg
STIKOM Bali	8/14/2023 12:59	12	8/14/2023 12:59	1692000853.7111306.jpg

On the second testing day, 16 visitors entered the library. The stored logs based on facial recognition registered 15 visitors, while the data from the visit form recorded only 10 visitors. Based on on-site observations, which involved direct observation and reviewing CCTV footage, it was noted that some visitors entered without registering their presence on the form. Furthermore, one visitor's face wasn't successfully captured by the system due to quick movement within the facial detection area. Table III shows a comparison between the two sets of data.

TABLE III
DATA COMPARISON ON THE SECOND DAY OF TESTING

Visitor Form Data		Log History		
Institution	**Check-in Date**	**ID**	**Log Date**	**File Name**
STIKOM Bali	8/15/2023 7:00	13	8/15/2023 7:00	1691982807.8329704.jpg
STIKOM Bali	8/15/2023 7:01	14	8/15/2023 7:01	1692000853.7111306.jpg
STIKOM Bali	8/15/2023 7:28	15	8/15/2023 7:28	1692066316.8154614.jpg
STIKOM Bali	8/15/2023 9:25	16	8/15/2023 9:25	1691997082.2824357.jpg
STIKOM Bali	8/15/2023 9:25	17	8/15/2023 9:25	1692071994.2014377.jpg

STIKOM Bali	8/15/2023 10:59	18	8/15/2023 9:40	1692075819.6012323.jpg
STIKOM Bali	8/15/2023 14:25	19	8/15/2023 9:41	1691986288.4124672.jpg
STIKOM Bali	8/15/2023 15:36	20	8/15/2023 10:59	1691986175.1838043.jpg
STIKOM Bali	8/15/2023 16:30	21	8/15/2023 14:25	1692084379.902643.jpg
STIKOM Bali	8/15/2023 16:31	22	8/15/2023 15:02	1692085712.8521242.jpg
		23	8/15/2023 15:19	1691991530.9594533.jpg
		24	8/15/2023 15:36	1692088322.853608.jpg
		25	8/15/2023 16:24	1692083868.2226784.jpg
		26	8/15/2023 16:30	1692100409.5817196.jpg
		27	8/15/2023 16:31	1692103658.7218068.jpg

D. Discussion

Based on the data obtained from the conducted testing, the system has performed well. Out of 28 visitors, 27 were successfully captured by the facial recognition system. Failures in counting visitors were attributed to visitors not being within the detection area for a sufficient duration, thus preventing the system from conducting accurate counting. The constructed system must initially detect faces to facilitate matching. However, due to the face detection method's accuracy not being 100%, there is a possibility of undetected faces. Consequently, in certain cases, visitors may need to remain within the detection area and adjust their positions until their faces are detected.

In the testing location, there isn't yet a strict physical mechanism that forces visitors to enter the detection area. Visitors might bypass or swiftly pass through the detection area. Given the limitations of the camera's detection area, a physical mechanism governing visitor entry should be established to optimize facial capture by the system.

IV. CONCLUSIONS

More accurate visitor counting can be achieved through computer vision. The constructed system, developed using the Python programming language, operates by detecting visitor faces and conducting counting based on the detected faces. To prevent the same visitor from being detected multiple times, a facial recognition method with dynamic facial data collection is implemented in this research. Face detection and recognition are accomplished using a model based on Histogram of Oriented Gradients (HOG) and Linear Support Vector Machine (SVM). The process of capturing unknown new faces is performed using a model based on the Haar Cascade method. The developed system successfully counted 27 out of 28 visitors over two days. Nonetheless, the system still possesses limitations, particularly in terms of the restricted detection area. Therefore, a physical mechanism that requires visitors to undergo facial scanning and registration needs to be implemented, ensuring that recorded data matches the actual visitor count. Subsequent research is recommended to enhance object detection with cameras positioned further away, enabling the counting of incoming and outgoing visitors using a single camera. This approach would disregard visitor details and solely provide actual visitor data present within the premises.

ACKNOWLEDGMENTS

Gratitude is extended to the Institut Teknologi dan Bisnis STIKOM Bali for funding this research.

DAFTAR PUSTAKA

[1] Almuttaqin and M. Nasir, “Rancang Bangun Alat Penghitung Jumlah Pengunjung Di Perpustakaan Politeknik Negeri Bengkalis Berbasis Mikrokontroler,” in *Seminar Nasional Industri dan Teknologi (SNIT)*, 2021, pp. 385–394.

[2] M. Fahmawaty, M. Royhan, and Mahmudin, “Perancangan Alat Penghitung Jumlah Pengunjung Di Perpustakaan Unis Tangerang Menggunakan Sensor Pir Berbasis IoT,” *JIMTEK : Jurnal Ilmiah Fakultas Teknik*, vol. 1, no. 3, pp. 253–261, 2020, [Online]. Available: www.thingspeak.com

[3] A. Atika Sari, I. Fitrianto Rahmad, and F. Tambunan, “Perancangan Dan Implementasi System Pendeteksi Pengunjung Pada Toko Berbasis Arduino,” *Jurnal FTIK*, vol. 1, no. 1, pp. 417–428, 2020, [Online]. Available: http://e-journal.potensi-utama.ac.id/ojs/index.php/FTIK/article/view/877

[4] I. M. B. Gautama, I. G. N. W. Arsa, and N. P. V. Savita, “Visitor Counter and Information Viewer at Photo Exhibitions using Embedded Systems and Web Services,” *Paradigma - Jurnal Komputer dan Informatika*, vol. 24, no. 2, pp. 152–159, Sep. 2022, doi: 10.31294/paradigma.v24i2.1430.

[5] M. N. Inrawansyah, “Implementasi Face Detection Menggunakan Metode Viola Jones Untuk Membantu Mempermudah Proses Counter Pengunjung Gedung,” *Jurnal Mahasiswa Teknik Informatika)*, vol. 1, no. 1, pp. 8–16, 2017.

[6] R. A. P, S. Achmadi, and K. Auliasari, “Penerapan Metode Convolutional Neural Network pada Aplikasi Deteksi Wajah Pengunjung Perpustakaan,” *Jurnal Mahasiswa Teknik Informatika*, vol. 6, no. 1, pp. 253–258, 2022.

[7] A. Rahim, A. Rahajoe, and M. Mahaputra, “Prediksi Jumlah Pengunjung Perperiode Terhadap Tempat Wisata Pantai Menggunakan Triple Exponential Smoothing (Studi Kasus Pantai Gili Labak Sumenep),” *Jurnal Ilmiah Teknologi Informasi dan Robotika*, vol. 3, no. 2, pp. 39–43, Dec. 2021, doi: 10.33005/jifti.v3i2.66.

[8] M. Heri Saputra, D. Erwanto, and R. Fatkhur Rizal, “Penghitung Jumlah Pengunjung Objek Wisata Dengan Metode Deep Learning MobileNet-SSD,” *Techné : Jurnal Ilmiah Elektroteknika*, vol. 21, no. 2, pp. 145–154, Sep. 2022, doi: 10.31358/techne.v21i2.313.

[9] D. C. Rini, A. Z. Arifin, A. Fanani, G. B. D. Prasanda, and W. N. P. Sunaryo, “Penerapan Fuzzy Inference System dalam Pengoptimalan Suhu Ruangan pada Double Air Conditioner (AC) Secara Otomatis,” *MathVision : Jurnal Matematika*, vol. 1, no. 1, pp. 11–16, Mar. 2019, [Online]. Available: http://journal.unirow.ac.id/index.php/mv/article/view/52

[10] R. Prathivi and Y. Kurniawati, “Sistem Presensi Kelas Menggunakan Pengenalan Wajah Dengan Metode Haar Cascade Classifier,” *Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, vol. 11, no. 1, pp. 135–142, 2020.

[11] M. W. Septyanto, H. Sofyan, H. Jayadianti, O. S. Simanjuntak, and D. B. Prasetyo, “Aplikasi Presensi Pengenalan Wajah Dengan Menggunakan Algoritma Haar Cascade Classifier,” *Telematika: Jurnal Informatika dan Teknologi Informasi*, vol. 16, no. 2, pp. 87–96, 2020.

[12] S. P. Putra, I. Fitri, and S. Ningsih, "Absensi Pengenalan Wajah Menggunakan Menggunakan Algoritma Eigenface Berbasis Web," *Journal of Applied Informatics and Computing*, vol. 5, no. 1, Feb. 2021, doi: 10.30871/jaic.v5i1.2711.

[13] A. Jamhari, "Perancangan Sistem Pengenalan Wajah Secara Real-Time pada CCTV dengan Metode Eigenface," *INISTA (Journal of Informatics Information System Software Engineering and Applications)*, vol. 2, no. 2, pp. 20–32, 2020.

[14] H. Muchtar and R. Apriadi, "Implementasi pengenalan wajah pada sistem penguncian rumah dengan metode template matching menggunakan open source computer vision library (opencv)," *RESISTOR (elektRonika kEndali telekomunikaSI tenaga liSTrik kOmputeR)*, vol. 2, no. 1, pp. 39–42, 2019.

[15] R. A. Pratama, S. Achmadi, and K. Auliasari, "Penerapan Metode Convolutional Neural Network pada Aplikasi Deteksi Wajah Pengunjung Perpustakaan," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 6, no. 1, pp. 253–258, 2022.

[16] A. Adouani, W. M. Ben Henia, and Z. Lachiri, "Comparison of Haar-like, HOG and LBP approaches for face detection in video sequences," in *2019 16th International Multi-Conference on Systems, Signals & Devices (SSD)*, 2019, pp. 266–271. doi: 10.1109/SSD.2019.8893214.

[17] C. Rahmad, R. A. Asmara, D. R. H. Putra, I. Dharma, H. Darmono, and I. Muhiqqin, "Comparison of Viola-Jones Haar Cascade classifier and histogram of oriented gradients (HOG) for face detection," in *IOP conference series: materials science and engineering*, 2020, p. 12038.

[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, 2005, pp. 886–893.

[19] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[20] V. Thakur, "Celebrity Face Image Dataset," Sep. 13, 2022. https://www.kaggle.com/datasets/vishesh1412/celebrity-face-image-dataset (accessed Jul. 03, 2023).

[21] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1867–1874.

[22] G. Zhang, H. Qin, Y. Ke, J. Chen, and Y. Gong, "Phased groupwise face alignment," *IEEE Access*, vol. 8, pp. 62415–62422, 2020.

[23] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823. doi: 10.1109/CVPR.2015.7298682.

[24] S. Ghosh, A. Dasgupta, and A. Swetapadma, "A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification," in *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, 2019, pp. 24–28. doi: 10.1109/ISS1.2019.8908018.

[25] E. O. Akay, K. O. Canbek, and Y. Oniz, "Automated Student Attendance System Using Face Recognition," in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2020, pp. 1–5. doi: 10.1109/ISMSIT50672.2020.9255052.

[26] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, pp. I–I. doi: 10.1109/CVPR.2001.990517.

[27] Y. Zhang *et al.*, "Research and Application of AdaBoost Algorithm Based on SVM," in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2019, pp. 662–666. doi: 10.1109/ITAIC.2019.8785556.

[28] N. Francis Leena Mary and Sreenath, "Pre-processing Techniques for Detection of Blurred Images," in *Proceedings of International Conference on Computational Intelligence and Data Engineering*, N. and S. A. and D. N. C. Chaki Nabendu and Devarakonda, Ed., Singapore: Springer Singapore, 2019, pp. 59–66.