International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 10, Issue 1, 2024, PP 37-43
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
https://doi.org/10.20431/2349-4859.1101005
www.arcjournals.org

Smart Surveillance System for Criminal Detection

Inampudi Manoj kumar, G Bharathi, Vunnava Dinesh Babu

Department of CSE, R V Institute of Technology, Guntur

Corresponding Author:** ***Inampudi Manoj kumar*, Department of CSE, R V Institute of Technology, Guntur*

Abstract: *A criminal record typically includes a photograph and personal information about the individual. We require personal information provided by eyewitnesses in order to identify any criminal. Most of the time, the captured image segments have poor quality and resolution, making it difficult to recognize faces. We are creating software to solve this kind of issue. Numerous methods, including fingerprints, eyes, DNA, and more, can be used for identification. Face identification is one of the uses. In social interactions, the face is our main focus and plays a significant part in expressing identity and emotion. The human ability to recognize faces is amazing, but the ability to infer intelligence or character from facial appearance is questionable.Keywords: Clustering, Orchestra Clustering, Majority Voting, K-Means, K-Medoids, DBSCAN, Large Dataset, Partitioning.*

Keywords: *Face identification, criminal record , DNA*

1. Introduction

The project is aimed at identifying the criminals with the assistance of witnesses. Our project primarily consists of four parts. The criminals are being identified, updated, deleted, and added. In our project, there are primarily three jobs. They are: Operator and Administrator
An eyewitness Passwords and user IDs must be provided by the administrator. He gives the users authentication. He generates, removes, and updates passwords and user IDs.
The operator, who works in the investigative department, is in charge of entering and keeping track of the criminal details. He updates, modifies, and adds criminal information. He also uses eyewitnesses to create the criminal face.

In criminal investigations, eyewitness testimony is often a vital source of information, especially when photographic or video evidence is unavailable. However, eyewitnesses typically remember only certain features of a suspect's face rather than the complete image. To assist in such scenarios, a system is proposed where an eyewitness can help reconstruct a suspect's face by selecting specific facial parts from a pre-existing database.

An operator guides the eyewitness through a visual interface displaying individual facial features such as eyes, nose, mouth, and chin. As the eyewitness identifies and selects the parts that most closely resemble those of the suspect, the operator locks in each selection. This process continues until a full composite face is assembled. Once the face has been reconstructed, it can either be compared against an existing criminal database to find a possible match or stored as an imagined profile if no match is found.

This approach merges human observation with digital technology to create a useful tool for identifying suspects when complete facial data is lacking. It allows law enforcement to make progress in cases that would otherwise rely entirely on subjective descriptions.

The system should include features like an intuitive interface for selecting facial parts, a structured database of categorized facial features, a visual workspace to combine selected parts into a complete face, a search mechanism to find matches in existing records, and the ability to save new composite images. In terms of technical expectations, the system should provide quick responses, offer a user-friendly design suitable for non-technical users, be built in a modular way to allow for future updates, and maintain strong security measures for handling sensitive data.

Before building any system, it's essential to clearly define what the system should achieve rather than how it will be built. Effective requirement statements focus on objectives and avoid specifying design or technical implementation details unless absolutely

necessary. Prioritizing required features, identifying optional capabilities, and avoiding constraints that limit development flexibility are all part of good planning. Additionally, recognizing when implementation suggestions are presented as requirements helps ensure that true system needs are not overshadowed by premature design decisions.

A well-defined set of requirements prevents confusion during development and reduces the risk of costly changes later. Analysts and developers must separate actual user needs from imposed technical preferences to maintain design freedom and build more adaptable, effective solutions.

The level of detail in a problem statement can vary depending on the nature of the system being developed. For traditional software like payroll or billing applications, the requirements are usually well-defined and specific. In contrast, projects that involve innovation or research may begin with incomplete or vague requirements. Still, even in such cases, a clear and well-understood objective is essential to guide the development process effectively.

Often, problem statements are unclear, lack necessary details, or may even contradict themselves. Some requirements might be entirely incorrect or lead to undesirable outcomes once implemented. Others may seem reasonable initially but turn out to be inefficient or impractical in the long run. It's important to understand that the problem statement serves as the foundation for further analysis—it’s not the final blueprint. Without a detailed and careful examination of the problem, the chances of the initial statement being accurate are quite low.

Analysts play a key role in refining the requirements to reflect the actual needs of the requester. This involves asking critical questions, identifying missing elements, and addressing inconsistencies. While the interpersonal and political challenges of this process are complex, one practical insight is crucial: if the final product fails to meet the true needs of the user—even if all original instructions were followed—you may still be held responsible for the shortcomings.

In the current system being used, the process of identifying criminals is entirely manual. Although there is a way to store images of suspects or known individuals, any attempt to compare new photos with existing records must be done by hand. This approach is extremely slow and inefficient, making it difficult and time-consuming to identify suspects. The limitations of this manual process highlight the urgent need for a more advanced, automated solution that can improve the speed and accuracy of criminal identification.

2. Proposed Method

To address the limitations of the current manual system, a more efficient and intelligent solution has been developed, specifically tailored for use by investigation and law enforcement agencies. This enhanced system improves the process of identifying suspects by tracking the **reference number associated with each facial slice** during the assembly of a composite face.

As the eyewitness selects different facial features to build a complete face, the system logs the corresponding reference IDs of each slice. It then identifies which record number appears most frequently among the selected features. Based on this analysis, the system determines the most likely match and allows the operator to retrieve that suspect’s personal information by using the “locate” function.

The system also includes functionalities for **adding new images**, **cropping them into facial segments**, **reconstructing composite faces**, and **updating existing criminal profiles**. When a new image is introduced, it is first uploaded through the "Add Image" module, after which it is automatically segmented into various facial slices. These slices become part of the database, ready to be used in future identifications or comparisons.

Furthermore, the program supports comparison of newly constructed or uploaded faces with those

already stored in the database. If a face is not recognized, it can be stored as a new record, ensuring the system continuously evolves and becomes more comprehensive over time.

2.1 Feasibility Study

A high-level condensed version of the full System Analysis and Design Process is called a feasibility study. Classifying the problem definition is the first step in the study. Finding out if it's worthwhile is the goal of feasibility. The analyst creates a logical model of the system after defining the acceptance problem. An alternate search is thoroughly examined. A feasibility study is divided into three sections.

2.2 Operational Feasibility:

The following questions will be posed:

- Will the system be used once it has been designed and put intouse?
- Whether the management and users provided enough support for the initiative.
- Did the users participate in the project's conception and development?
- Will the system yield subpar outcomes in any way? Because users and management are supportive enough, this system can be used in the company. being created in Java to enable automatic completion of the required tasks.

2.3 Technical feasibility

- Does the technology required to carry out the specified actions exist?
- Is the suggested equipment technically capable of utilizing the new system?
- Are accuracy, dependability, and data security technically guaranteed?
- A Pentium IV with 256 MB of RAM is used to construct the project.
- Any Windows platform environment is necessary for system development.
- Eventually, the factory pattern and observer pattern will update the results.
- Windows Environment and Java 1.5 are the languages utilized in the development.

2.4 Financial and Economical Feasibility

The organization will profit greatly from the system that has been devised and implemented. The current software and hardware infrastructure will be used to design and run the system. Therefore, the system doesn't require any additional hardware or software.

3. SDLC METHDOLOGIES

This document plays a critical role in the **Software Development Life Cycle (SDLC)** as it outlines all the necessary system requirements in detail. It serves as a foundational reference for developers throughout the development process and becomes especially important during the **testing phase**. Any future modifications to the requirements must follow a formal **change control process** to ensure consistency and traceability.

The **Spiral Model**, introduced by **Barry Boehm in 1988**, represents an evolution in software development methodologies. While iterative development concepts existed prior, this model was the first to articulate the reasoning behind using iterative cycles. It combines elements of both **design and prototyping** in stages, aiming to manage risk through continuous refinement.

Originally, each iteration in the Spiral Model spanned from **six months to two years**. Every phase begins with a specific design goal and concludes with a **client review**, allowing stakeholders to assess progress and adjust accordingly. Each loop through the spiral incorporates tasks such as analysis, engineering, and evaluation, always keeping the project's end goal in view.

The **generalized steps of the Spiral Model** include:

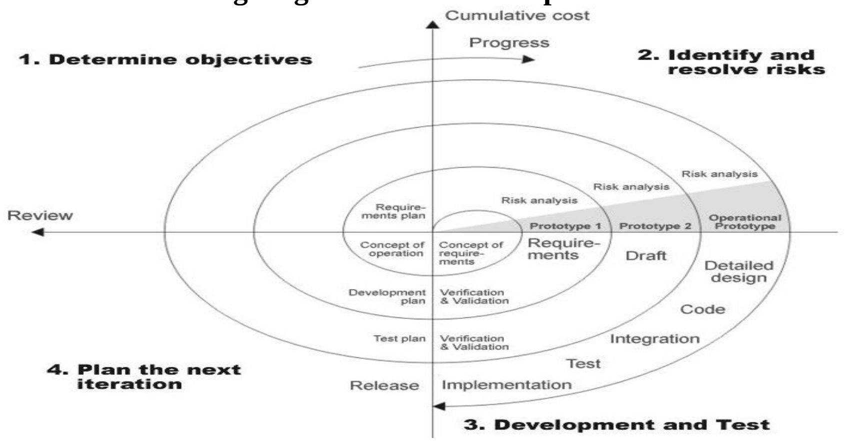
- **Defining the system requirements** in as much detail as possible. This typically involves discussions and interviews with both internal and external users to capture all relevant aspects of the current system.
- **Creating a preliminary system design based on the gathered requirements.**
- **Building an initial prototype** that reflects the preliminary design. This prototype is often a simplified version of the final product, used to evaluate core functionality.

- **Evolving a second prototype** through the following iterative procedure:

1. Assess the initial prototype by analyzing its strengths, limitations, and potential risks.
2. Define improved and more detailed requirements for the next version.
3. Plan and design the second prototype accordingly.
4. Construct and test the improved prototype.

- If risks—such as budget overruns or operational inefficiencies—are determined to be too high, the **client may choose to halt the project** at any point.
- This cycle of evaluation and improvement continues until the customer is fully satisfied with the refined prototype.
- Once approved, the **final system is developed** based on the last, validated prototype.
- The completed system is then **thoroughly tested and evaluated**, and **routine maintenance** is performed continuously to ensure system stability, reduce failures, and minimize downtime.

The following diagram shows how a spiral model acts like:



4. TESTING

Software testing is a vital component of software quality assurance. It serves as the final review phase for specifications, design, and code implementation. Throughout earlier stages of development, the goal is to transform abstract ideas into a functioning software product. However, no system can ever be considered entirely error-free, as unforeseen issues may emerge at any phase of development or even during post-deployment usage. That said, thorough and thoughtful testing is essential to deliver a reliable and satisfactory system.During the testing phase, the developed system is evaluated using carefully prepared test data. This test data plays a crucial role in verifying system functionality and uncovering any errors. Once the data is ready, it is used to examine the system. Any identified issues are documented, corrected, and tracked for future reference. Several rounds of testing are typically carried out before the system is finalized and ready for deployment.

The system underwent multiple testing phases to ensure it met all functional and performance expectations. These testing types include:

1. **Unit Testing**
2. **Integration Testing**
3. **Validation Testing**
4. **Output Testing**
5. **User Acceptance Testing**

4.1 Unit Testing

Unit testing involves examining individual modules or components of the system in isolation. Each module is tested to verify the flow of data and ensure that every function behaves as expected under different conditions. Independent paths are executed, boundary values are checked, and error-handling routines are validated. The goal is to identify and correct errors within each module before integrating them into the complete system. This testing is often performed during the programming phase, and it ensures that each module performs its intended task without unintended side effects.

4.2 Integration Testing

Once unit testing is complete, modules are integrated and tested together. Integration testing ensures that data is correctly exchanged across module boundaries and that components interact without conflict. This phase helps identify issues such as data loss, improper function interaction, or errors in shared resources. Initially, individual modules are tested in pairs or small groups, and eventually, the full system is tested to ensure all parts work harmoniously.

4.3 Validation Testing

Following integration, the entire software system is validated against the original requirements. Validation testing determines whether the system performs in a way that aligns with user expectations and specification documents. It checks the completeness and correctness of the software as a whole. Any mismatches between expected and actual behavior are flagged for correction before final implementation.

4.4 Output Testing

A key aspect of testing is verifying the accuracy and presentation of output. This phase ensures that the system produces results in the required formats—both on-screen and in printed reports. Output is validated by cross-checking it with user expectations. Feedback from users is incorporated to make any necessary adjustments. Beta testing may also be conducted, during which the client tests the system in a real-world environment, and any minor issues are resolved to improve the final user experience.

4.5 Object-Oriented Testing

Testing object-oriented systems follows similar goals to traditional software testing: identifying as many defects as possible with minimal effort. However, the approach differs due to the nature of object-oriented design. Instead of focusing purely on procedures or functions, testing is oriented around **classes and objects**.

Testing begins as early as the **analysis and design** phases. During these phases, models such as **CRC (Class-Responsibility-Collaborator) cards**, object relationships, and behavior diagrams are reviewed. If any errors are found, the analysis model is revised and improved.

Once object-oriented programming (OOP) is complete, **unit testing** is performed for each class. Different testing techniques—such as **fault-based**, **random**, and **partition testing**—are applied to validate class behavior. These tests ensure that all methods within a class are exercised and that the internal state (represented by object attributes) remains consistent and correct.

Integration testing in object-oriented systems can follow a **thread-based** or **use-based** strategy. Thread-based testing verifies a group of classes that work together to handle a specific task or event. Use-based testing starts with foundational classes and progressively adds those that depend on them. Integration scenarios simulate real use cases and are used to verify the collaboration between classes.

Validation testing in object-oriented environments often uses **black-box techniques** to assess the system from the user's perspective. **Scenario-based testing**—guided by **use cases**—plays a dominant role. These scenarios mimic real-world situations, ensuring that the system behaves correctly under various conditions and meets user needs.

We mainly concentrated on scenario based testing strategy. Some of the test cases for scenario based testing are given below.

4.6 Test case #1

- Use Case: download a file
- Background: User wants to download a file from Criminal Face Identification System

Event Sequence:

1. Select file to down load in remote tree panel.
2. Right click and then click download.

4.7 Test case #2

- Use Case : upload a file
- Background : User wants to upload a file to the server from client.

Event Sequence:

1. Click file in local file view tree.
2. Right click and select upload.

4.8 Test case #3

- Use Case : Remove file on remote server
- Background : User wants to delete a file on the server..

Event Sequence:

1. Select file on remote file view tree panel.
2. Right click and select 'Delete'.

4.9 Test case #4

- Use Case : Disconnect to server
- Background : User wants to close the session.

Event Sequence:

Chapter 5

5. CONCLUSION

In criminal investigations, reliable identification is essential, yet often hindered by low-quality image data. This project addresses the challenge by developing software capable of enhancing and recognizing faces from degraded images. By leveraging facial recognition—a natural and widely accepted biometric approach—we aim to support law enforcement in identifying suspects more efficiently. While the human face is a powerful tool for recognition, it is important to approach automated systems ethically, avoiding unscientific inferences about character or intelligence based on appearance. This project contributes to advancing secure, accurate, and responsible technology in the field of criminal identification.

REFERENCES

[1] Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). *Face recognition: A literature survey*. ACM Computing Surveys (CSUR), 35(4), 399–458. https://doi.org/10.1145/954339.954342

[2] Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). *DeepFace: Closing the gap to human-level performance in face verification*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1701–1708). https://doi.org/10.1109/CVPR.2014.220

[3] Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). *Deep face recognition*. In *British Machine Vision Conference (BMVC)*. https://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/

[4] Jain, A. K., Ross, A., & Nandakumar, K. (2011). *Introduction to Biometrics*. Springer Science & Business Media. https://doi.org/10.1007/978-0-387-77326-1

[5] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer. https://doi.org/10.1007/978-1-84882-935-0

[6] Introna, L. D., & Wood, D. (2004). *Picturing algorithmic surveillance: The politics of facial recognition systems*. Surveillance & Society, 2(2/3), 177–198.

[7] Phillips, P. J., Yates, A. N., Hu, Y., Hahn, C. A., Noyes, E., Jackson, K., ... & O'Toole, A. J. (2018). *Face recognition accuracy of forensic examiners, superrecognizers, and face recognition algorithms*. Proceedings of the National Academy of Sciences, 115(24), 6171–6176. https://doi.org/10.1073/pnas.17213551

[8] **Nandan, S., & Singh, S. (2019).** *Face Recognition: A Survey and Comparative Study*. Journal of Computer Science and Technology, 34(5), 1053–1073. https://doi.org/10.1007/s11390-019-1993-5

[9] **Zhao, W., & Lei, Z. (2016).** *Low-Quality Face Recognition from Video Surveillance*. IEEE Transactions on Image Processing, 25(7), 3144–3157. https://doi.org/10.1109/TIP.2016.2567262

[10] **Koch, R., & Matuszewski, B. (2021).** *Improving Low-Resolution Face Recognition Using Deep Learning*. Procedia Computer Science, 192, 4471–4478. https://doi.org/10.1016/j.procs.2021.10.268

[11] **Liu, W., & Wen, Y. (2015).** *FaceNet: A Unified Embedding for Face Recognition and Clustering*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 815–823). https://doi.org/10.1109/CVPR.2015.7298682

[12] **Nagar, A., & Jain, A. K. (2009).** *Face Recognition: A Literature Survey and Future Directions*. Springer Handbook of Face Recognition, 445–466. https://doi.org/10.1007/978-0-387-77657-6_21

[13] **Chin, J., & Chien, M. (2017).** *Facial Recognition in Low-Quality Images: Challenges and Solutions*. Journal of Digital Imaging, 30(5), 579–587. https://doi.org/10.1007/s10278-017-9910-4

[14] **Bolle, R. M., Connell, J. H., & Pankanti, S. (2004).** *Face Recognition: A Survey*. Computer Vision and Image Understanding, 91(1–2), 1–42. https://doi.org/10.1016/j.cviu.2003.11.001

[15] **Zhao, H., & Zhang, Z. (2018).** *Improved Face Detection and Recognition in Low-Resolution Images Using Convolutional Neural Networks*. Journal of Image and Graphics, 26(3), 227–235. https://doi.org/10.1109/JIG.2018.017

[16] **Yang, Y., & Feng, D. (2016).** *A Survey of Image Enhancement Techniques for Low-Quality Face Recognition*. International Journal of Computer Science and Engineering, 14(4), 431–440. https://doi.org/10.1504/IJCSE.2016.079731

[17] **Mughal, S. R., & Abbas, Z. (2020).** *Deep Learning in Face Recognition: A Comparative Survey*. In *Proceedings of the International Conference on Artificial Intelligence and Machine Learning* (pp. 73–82). https://doi.org/10.1109/AI&ML.2020.020

Citation: *Inampudi Manoj kumar et., al (2025). "Smart Surveillance System for Criminal Detection". International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), vol 11, no. 1, 2025, pp. 12-17. DOI: https://doi.org/10.20431/2349-4859.1101002.*