Kalpa Publications in Computing
Volume 19, 2024, Pages 455–466
Proceedings of 6th International Conference on Smart Systems and Inventive Technology

Child Monitoring and Data Analysis via Computer Vision

Shantanu Tripathi[1], Ashutosh Sharma[1], and Gokulakrishnan D[2]

[1]SRM Institute of Science & Technology, Computer Science & Engineering, Chennai, India
{st0674, as5279}@srmist.edu.in
[2]SRM Institute of Science & Technology, Assistant Professor,Department of Computing Technology, Chennai, India
gokulad@srmist.edu.in

Abstract.

Ensuring children's safety is a top priority for parents globally amidst urban and suburban complexities. Traditional supervision methods are increasingly impractical in today's fast-paced society, leading to a rise in missing children cases, such as the 47,000 reports in India in 2023. To combat this, an innovative solution harnessing computer vision technology, specifically OpenCV, has been developed. This system offers real-time monitoring within designated safe zones, utilizing sophisticated image processing and object tracking. Its user-friendly interface aims to encourage widespread adoption, catering to non-technical users. Beyond tracking, the system provides instant updates and alerts, transforming child safety management into a proactive and less stressful endeavor. This integration of advanced technology with user-centered design signifies a significant advancement in child safety, promising peace of mind for caregivers and a safer future for children worldwide.

Keywords: OpenCV, Child Monitoring, Kernelized Correlation Filter.

1 Introduction

The safety of children in today's fast-paced and often unpredictable world has become a paramount concern for parents, educators, and communities. Traditional approaches to ensuring child safety are increasingly challenged by the complexities of modern environments. Project addresses these concerns by harnessing the power of computer vision and real-time data processing to offer a sophisticated yet user-friendly child monitoring and tracking system. It represents a significant step forward in using technology to create safer spaces for children, whether at home, in school, or in public areas.

In an era of increasing technological advancements, ensuring the safety and well-being of children has become a paramount concern. This research proposes an innovative System that combines real-time tracking technology with analytical insights to offer comprehensive protection and informed decision-making for caregivers. The system employs geofencing and proximity alerts to notify caregivers when a child enters or exits predefined safe zones, enhancing proactive safety measures.

Existing child monitoring systems exhibit several challenges that limit their effectiveness and user adoption. Key issues include the invasive nature of some technologies, leading to privacy concerns; reliance on GPS signals, which can be unreliable indoors; and complex interfaces that deter use by non-technical guardians. Furthermore, many systems do not offer real-time video monitoring, a feature that can provide immediate context and assurance about a child's safety. This project aims to address these challenges by prioritizing privacy, utilizing indoor-compatible tracking technologies, and simplifying user interaction.

2 Literature Review

Introduction of Vision-Based Techniques in Child Monitoring Systems:

The study aims to enhance the detection and tracking capabilities of child monitoring systems through the implementation of advanced vision-based techniques. It introduces an algorithm designed for continuous monitoring of children with improved coordination and object tracking features, addressing limitations of stationary monitoring solutions.
Reference: John Anthony, C. Jose, Justine Veronica Basco et al. (2019) [1].
Predicting Risky Environments Using Deep Learning:

Naeem Ahmad, Sumedha Arya, Deepak Singh et al. (2023) propose a paper titled "Predicting Risky Environment for Child Inside House using Deep Learning," exploring the use of deep learning techniques to identify potentially risky environments for children within households. The aim is proactive risk mitigation to ensure child safety at home.
Reference: [2].
Computer Vision for Attention Monitoring in Online Meetings:

Tristram Decayan, Daehan Kwak, Xudong Zhang et al. (2022) contribute to the field with "Computer-Vision Based Attention Monitoring for Online Meetings," investigating the application of computer vision to monitor attention levels during online meetings. The study aims to enhance meeting experiences by providing real-time attention monitoring and feedback.
Reference: [3].
Secure Monitoring Systems:

Rameesa et al. [4] introduce a surveillance system anchored in Raspberry Pi and Pi camera, prioritizing security and reliability for child monitoring.
Comprehensive Monitoring Systems:

Shreelatha et al. [3] expand monitoring scope to include environmental conditions like temperature and humidity, providing insights into child activities and sleeping patterns for parents.
Remote Monitoring Prototypes:

Patil and Mhetre present a GSM-based monitoring prototype monitoring critical health indicators such as body temperature and heart rate, with data sent conveniently to parents via the GSM network [5].
Wearable Technology Integration:

Ziganshin et al. [6] integrate smart wearable devices with Android applications to track the baby's position, temperature, and pulse, ensuring prompt alerts for parents in case of irregularities.
GPS and Acceleration Sensors for Child Safety:

Saranyaet [6] discusses the potential of Android phones equipped with GPS and acceleration sensors, leveraging mobile GIS technology for safeguarding children through advanced tracking technologies.
IoT-enabled Safety Systems:

Automated systems featuring IoT capabilities have been proposed to prevent infants from falling off beds, providing email notifications to parents or guardians with a photograph of the infant if the baby gets too close to the bed's edge, thus enhancing safety [7-11].
In conclusion, these research papers collectively highlight significant advancements in utilizing computer vision and deep learning technologies to enhance child monitoring, predict risky environments, and monitor attention levels in various contexts, contributing to the evolving landscape of robotics and computer vision applications for child safety and monitoring.

3 Proposed System

The proposed child monitoring system integrates various components to enhance its capabilities effectively. By leveraging OpenCV for real-time child tracking, the system benefits from the optimized C-based library's efficiency in computer vision tasks, including image processing, object detection, and tracking. This integration ensures accurate monitoring and tracking of children.

Additionally, the inclusion of a Streamlit interface provides an interactive platform for data input and visualization, enhancing user experience by offering a user-friendly interface for inputting child monitoring data and visualizing relevant information clearly.

The system securely stores user credentials and child details in an SQLite database, ensuring data integrity, security, and efficient retrieval of information related to guardians, children, and monitoring activities.

Integrating the Python Imaging Library (PIL) enables advanced image processing techniques within the system, enhancing the quality of visuals and data representation for improved monitoring outcomes.

The system incorporates dynamic safe zone definition features, allowing guardians to set and modify safe zones as needed. This functionality provides flexibility and control over safety parameters, enabling customization based on specific requirements or changing circumstances.

By combining these components, the proposed system offers a comprehensive solution for child monitoring that leverages advanced technologies for accurate tracking, interactive user interfaces, secure data management, enhanced image processing, and customizable safety settings. This integration ensures effective and reliable child monitoring capabilities, enhancing safety and peace of mind for guardians and caregivers.

The novelty of the project lies in its ability to allow users to define a specific safe/unsafe zone in the visible view of the camera. The system is designed to alert the User via live notification when the monitored entity enters or exits the predefined safe/unsafe zone. Furthermore, the data analysis performed at the end of each recording session provides the user with valuable, actionable insights which are highly useful in the long term.

4 Methodology

Proposed architecture diagram is shown in the below Fig 1.

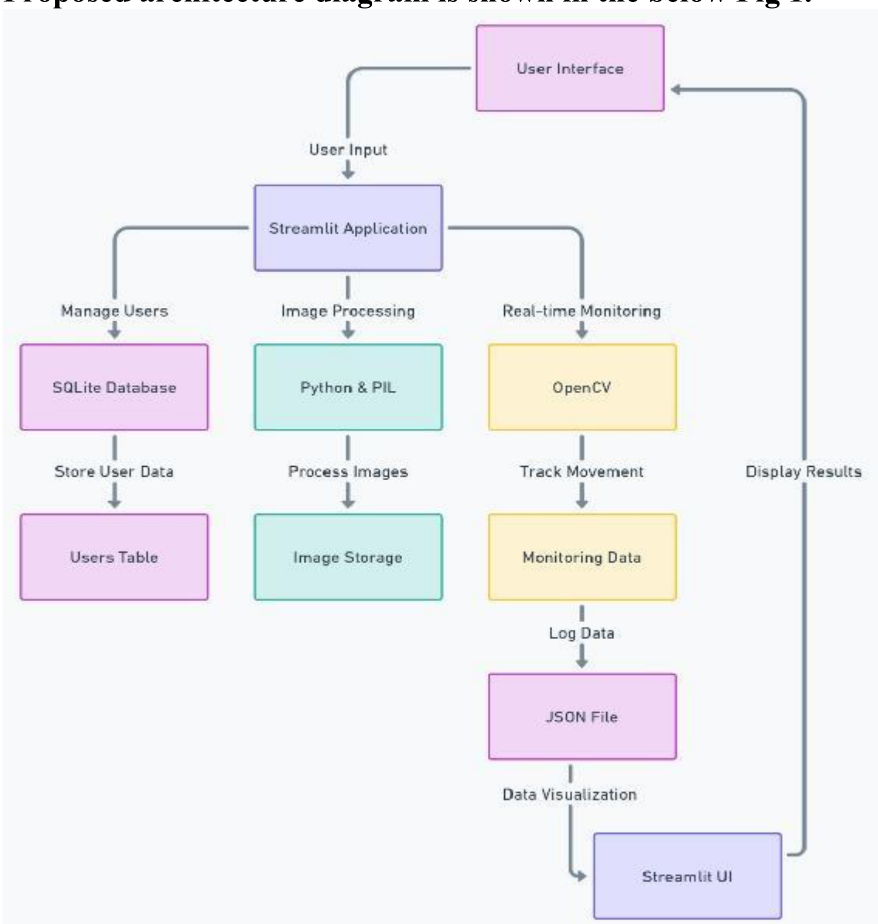


Figure. 1.: Proposed architecture diagram

The Child Monitoring System starts with taking user data as information to create a profile. Then, with the help of Region of Interest (ROI), we define our region where our subject lies. Which can be done via Haar cascade detecting the entity face or YOLO detecting the whole body and giving its coordinates in the image to the tracker as the target to track.

Next, we take our subject and define a safe region of interest. Process includes the following steps:

Saving Child Data and Image: The code saves uploaded image data along with the child's name, age, and place into a JSON file.

Analyzing an Image: It analyzes an image to identify regions of interest (ROIs) for the child and a safe region. It allows interactive selection of ROIs using OpenCV's cv2.selectROI function. It checks for overlap between the child's ROI and the safe ROI and stores the resulting data (ROIs and overlap status) in a JSON file.

Displaying Monitoring Data: It loads monitoring data from a JSON file, ensures consistency in data length, creates a pandas DataFrame from the data, and displays it.

Displaying Current Location Data: It creates a pandas DataFrame for current location data and displays it.

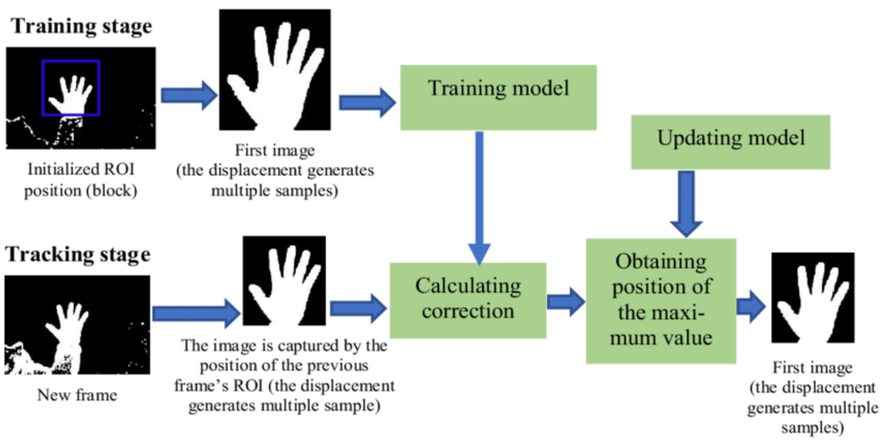


Figure. 2: Working of KCF(Kernelized Correlation Filter)

For ideal tracking, we use the KCF tracking algorithm(Fig 2). The KCF Tracking Algorithm is employed in computer vision for object tracking, utilizing Kernelized Correlation Filters (KCF) to track objects efficiently and robustly within a video sequence. It's commonly used in scenarios where continuous monitoring and tracking of objects are necessary, such as surveillance systems, video analytics, autonomous vehicles, and augmented reality applications.

Algorithms are implemented through libraries like OpenCV, which provide functions and methods to initialize the tracker, update its position in each frame, and handle occlusions or changes in appearance. The value of KCF lies in its ability to run on minimal hardware while still providing a state-of-the-art level of accuracy in tracking, making the system highly adaptable for the masses.

The use of location data is helpful should the system be put into operation in a moving vehicle such as a car or a bus, and even aside from that, it is helpful for someone who's using the system on multiple locations. Allows clarity regarding which stream is coming from which location. The intended User Workflow is shown in Fig 3.

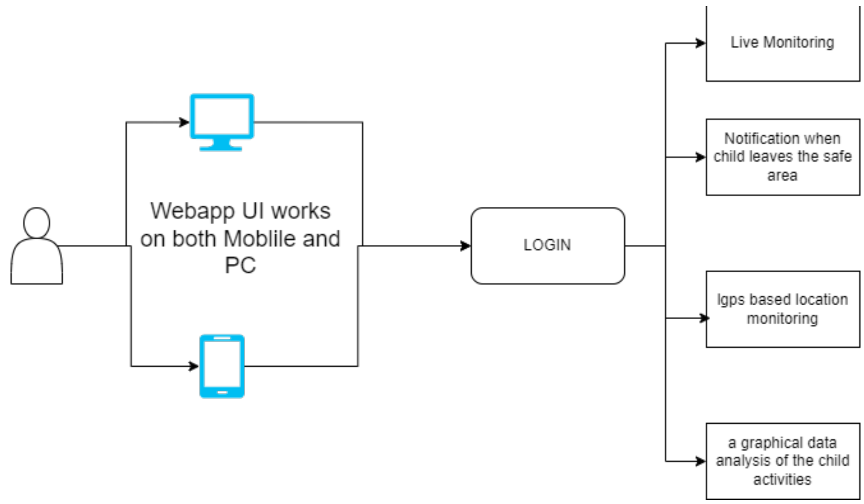


Figure 4: Intended User Workflow

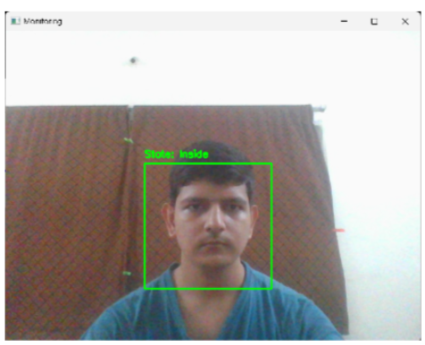
5 İmplementation & Result

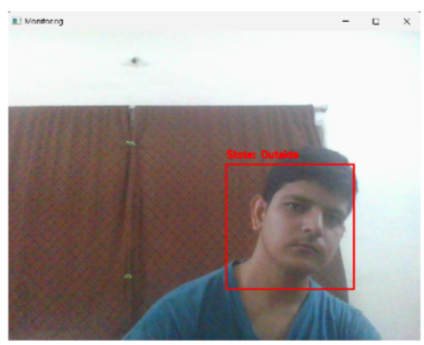
Validated coding and results are shown in Figure 4.

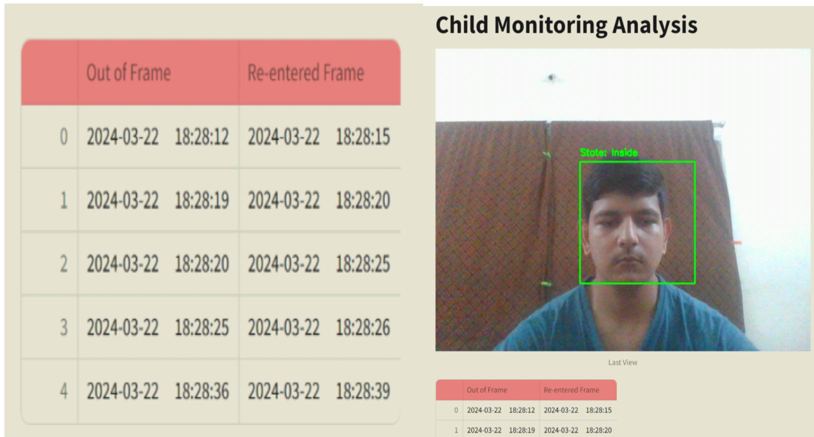
```
import cv2
import json
import datetime
import os

data = {
    "out_of_frame": [],
    "re_entered_frame": [],
    "additional_info": []
}

def start_monitoring():
    with open("child_data.json", "r") as f:
        child_data = json.load(f)

    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    roi = cv2.selectROI("Select the child", frame, False, False)
    tracker = cv2.TrackerKCF_create()
    tracker.init(frame, roi)

    safe_region = cv2.selectROI("Select the safe region", frame, False, False)

    out_of_frame = False

    while True:
        ret, frame = cap.read()
        success, roi = tracker.update(frame)
        (x, y, w, h) = tuple(map(int, roi))

        if success:
            inside_safe_region = all([x >= safe_region[0],
                              y >= safe_region[1],
                              x + w <= safe_region[0] + safe_region[2],
                              y + h <= safe_region[1] + safe_region[3]])


            if inside_safe_region:
                if out_of_frame:
                    data["re_entered_frame"].append(str(datetime.datetime.now()))
                    out_of_frame = False
                color = (0, 255, 0)  # Green
                state_text = "Inside"
            else:
                if not out_of_frame:
                    data["out_of_frame"].append(str(datetime.datetime.now()))
                    out_of_frame = True
                color = (0, 0, 255)  # Red
                state_text = "Outside"

            cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
            cv2.putText(frame, f"State: {state_text}", (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
        else:
            if not out_of_frame:
                data["out_of_frame"].append(str(datetime.datetime.now()))
                out_of_frame = True

        cv2.imshow("Monitoring", frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            if out_of_frame:  # Check if currently outside the safe zone
                data["re_entered_frame"].append(str(datetime.datetime.now()))
            cv2.imwrite("last_view.jpg", frame)
            break

    cap.release()
    cv2.destroyAllWindows()

    with open("monitoring_data.json", "w") as f:
        json.dump(data, f)

    os.system("streamlit run display.py")

if __name__ == "__main__":
    start_monitoring()
```

```python
# streamlit_display.py
import streamlit as st
import json
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
def plot_out_of_frame_correlation(data):
    # Convert strings to datetime objects
    out_of_frame = [datetime.strptime(time, "%Y-%m-%d %H:%M:%S.%f") for time in data["out_of_frame"]]
    re_entered_frame = [datetime.strptime(time, "%Y-%m-%d %H:%M:%S.%f") for time in data["re_entered_frame"]]

    # Calculate durations and corresponding hours
    durations = [(re_entered_frame[i] - out_of_frame[i]).total_seconds() for i in range(len(out_of_frame))]
    hours = [time.minute for time in out_of_frame]

    # Plotting
    fig, ax = plt.subplots(figsize=(10, 6))
    ax.scatter(hours, durations, color='blue', alpha=0.5)
    ax.set_title('Correlation between minute of Day and Duration Out of Frame')
    ax.set_xlabel('minute of Day')
    ax.set_ylabel('Duration Out of Frame (seconds)')
    ax.grid(True)
    return fig
def main():
    st.title("Child Monitoring Analysis")

    with open("monitoring_data.json", "r") as f:
        data = json.load(f)

    # Display the last view image
    st.image("last_view.jpg", caption="Last View", use_column_width=True)

    # Ensure both lists have the same length
    max_len = max(len(data["out_of_frame"]), len(data["re_entered_frame"]))
    while len(data["out_of_frame"]) < max_len:
        data["out_of_frame"].append(None)
    while len(data["re_entered_frame"]) < max_len:
        data["re_entered_frame"].append(None)

    # Display out of frame and re-entered frame times
    df = pd.DataFrame({
        "Out of Frame": data["out_of_frame"],
        "Re-entered Frame": data["re_entered_frame"]
    })
    # Convert strings to datetime objects
    df["Out of Frame"] = pd.to_datetime(df["Out of Frame"])
    df["Re-entered Frame"] = pd.to_datetime(df["Re-entered Frame"])

    # Extract and format date, hours, minutes, and seconds
    df["Out of Frame"] = df["Out of Frame"].dt.strftime("%Y-%m-%d     %H:%M:%S")
    df["Re-entered Frame"] = df["Re-entered Frame"].dt.strftime("%Y-%m-%d     %H:%M:%S")
    st.write(df)
```

```python
    ax.bar(["Out of Frame", "Re-entered Frame"], [len(data["out_of_frame"]), len(data["re_entered_frame"])])
    ax.set_ylabel('Counts')
    ax.set_title('Out of Frame vs Re-entered Frame Counts')
    st.pyplot(fig)
    # Convert strings to datetime objects
    out_of_frame = [datetime.strptime(time, "%Y-%m-%d %H:%M:%S.%f") for time in data["out_of_frame"]]
    re_entered_frame = [datetime.strptime(time, "%Y-%m-%d %H:%M:%S.%f") for time in data["re_entered_frame"]]

    # Calculate total time outside and inside frame
    total_outside = sum((re_entered_frame[i] - out_of_frame[i]).total_seconds() for i in range(len(out_of_frame)))
    total_inside = sum((out_of_frame[i+1] - re_entered_frame[i]).total_seconds() for i in range(len(re_entered_frame)-1)

    # If the last event is a re-entered_frame, calculate until the last recorded time
    if len(re_entered_frame) > len(out_of_frame):
        total_inside += (re_entered_frame[-1] - out_of_frame[-1]).total_seconds()
    elif len(out_of_frame) > len(re_entered_frame):
        # This means the person ended outside the frame, you might need the current time or some logic to handle it
        pass  # You could handle this case if necessary

    # Prepare data for pie chart
    counts = [total_outside, total_inside]
    labels = ["Outside Frame", "Inside Frame"]

    # Create a pie chart
    fig, ax = plt.subplots()
    ax.pie(counts, labels=labels, autopct='%1.1f%%', startangle=90)
    ax.axis('equal')
    # Title for the pie chart
    plt.title('Out of Frame vs Re-entered Frame Counts')

    # Display the plot in the Streamlit app
    st.pyplot(fig)
    # Calculate counts
    out_of_frame_count = len(data["out_of_frame"])
    re_entered_frame_count = len(data["re_entered_frame"])
    counts = [out_of_frame_count, re_entered_frame_count]
    categories = ["Out of Frame", "Re-entered Frame"]

    # Plot
    fig, ax = plt.subplots()
    ax.barh(categories, counts, color=['skyblue', 'orange'])
    ax.set_xlabel('Counts')
    ax.set_title('Out of Frame vs Re-entered Frame')

    # Create the plot
    fig = plot_out_of_frame_correlation(data)

    # Display the plot
    st.pyplot(fig)

if __name__ == "__main__":
    main()
```







	Out of Frame	Re-entered Frame
0	2024-03-22 18:28:12	2024-03-22 18:28:15
1	2024-03-22 18:28:19	2024-03-22 18:28:20
2	2024-03-22 18:28:20	2024-03-22 18:28:25
3	2024-03-22 18:28:25	2024-03-22 18:28:26
4	2024-03-22 18:28:36	2024-03-22 18:28:39

Figure. 4: Partial Code & Result

6 Conclusion

The Child Monitoring System offers users the capability to designate a target and define a safe zone, enabling them to effectively track the child's movements. Through the system, users receive alerts if the child strays outside the predetermined safe area, ensuring enhanced security and peace of mind.

Upon concluding the monitoring session, the system delivers a comprehensive analysis of the accumulated data. The analysis includes the child's most recent location, a detailed log documenting instances of entering and exiting the safe zone, and insightful data analysis presented through intuitive graphical representations.

By providing real-time monitoring and insightful data analysis, the Child Monitoring System equips users with the necessary tools to ensure the safety and well-being of their children while offering valuable insights into their activities.

A cutting-edge child monitoring system has been developed to reliably track and analyze children's movements within controlled environments. This innovative system provides comprehensive data analysis at the conclusion of monitoring sessions, offering actionable insights to users. Leveraging the advanced capabilities of the KCF algorithm, the system ensures smooth operation even on hardware with limited resources, thus enhancing its potential for widespread adoption among users.

7 Future Enhancements

Advanced Machine Learning Algorithms: Exploring advanced machine learning algorithms would enhance the system's capability to recognize patterns in the child's behavior, offering personalized insights and recommendations.

Two-Way Communication: Incorporating two-way communication features would enable caregivers to interact with the child remotely, providing support or guidance as needed.

Cloud Integration: Integrating with cloud services would securely store monitoring data and allow remote access from anywhere, facilitating convenient monitoring and analysis.

Improved Data Visualization: Developing interactive and customizable data visualization tools would provide caregivers with a clear understanding of the child's movements and behavior trends over time.

User Profiles and Permissions: Implementing user profiles and permission settings would enable multiple caregivers or family members to securely access and manage the system, with varying levels of authorization based on their roles.

Emergency Response Integration: Integrating with emergency response services or mobile apps would enable swift and coordinated responses in emergencies or unexpected situations.

Feedback Mechanism: Incorporating a feedback mechanism would gather input from users and caregivers, allowing continuous improvement based on their experiences and requirements.

By incorporating these enhancements, the Child Monitoring System can further ensure the safety and well-being of children while providing caregivers with valuable insights and peace of mind.

References

[1] John Anthony, C. Jose, Justine Veronica Basco et al. (2019), "Vision-Based Detection and Tracking Algorithm for a Child Monitoring Robot."

[2] Naeem Ahmad, Sumedha Arya, Deepak Singh et al. (2023), "Predicting Risky Environment for Child Inside House using Deep Learning."

[3] Tristram Decayan, Daehan Kwak, Xudong Zhang et al. (2022), "Computer-Vision Based Attention Monitoring for Online Meetings."

[4] Rameesa. O, C Periasamy and Priyanka. M. T. “Real Time Child Monitoring System based on Raspberry Pi and Beacon Technology using Android App”. Int. J. Adv. Res.6(3), pp. 1410- 1416, ISSN:2320-5407

[5] S. Patil and M. Mhetre, "Intelligent Baby Monitoring System", ITSI Transactions on Electrical and Electronics Engineering, vol. 2, no. 1, pp. 11-16, 2014.

[6] E. Ziganshin, M. Numerov and S. Vygolov, "UWB Baby Monitor", 2010 5th International Conference on Ultrawideband and Ultrashort Impulse Signals, pp. 159-161, 2010.

[7] A. Saranyaet “Design and implementation of automatic child monitoring (ACM) system using wireless network”, International Journal of Computer Science and Mobile Computing, 5(4) April2016, pp. 356-363.

[8] A.N. Ansari, M.Sedky, N.Sharma and A.Tyagi, "An Internet of Things Approach for Motion Detection using Raspberry Pi," 2015.

[9] Sagar S Bachhav, Dr.Nilkanth B Chopade, “IoT Based HealthyBaby Cradle System”, 2018, IJRIEE

paper.

[10] Kumar, S., Rao, P. R., & Prasad, C. R. (2019). Internet of things based pollution tracking and alerting system. International Journal of Innovative Technology and Exploring Engineering, 8(8), 2242-2245.

[11] SavitaP.Patil, Manisha R .Mhetre, INTELLIGENT BABY MONITORING SYSTEM, ITSI Transactions on Electrical and Electronics Engineering ISSN 2320-8945 Volume-2 (2014)