Elderly Fall Detection System

Subject: Wagner: Safety Critical Computer Systems - SoSe 2024
Institution: Frankfurt University of Applied Sciences
Team Members:

- Akshay Dineshbhai Baldha
- Muhammad Usama Atiq
- Krupeshbhai Vipulbhai Pethani
- Jay Hemantkumar Mishra
- Asim Ali
- Saif Ahmed

Introduction

Problem Statement:

- **Objective:** Develop an application for Human Activity Recognition (HAR) focusing on gait analysis and fall detection.
- **Context:** Increasing elderly population and the need for systems to monitor their safety.
- **System Requirements:** Developed in MATLAB with a GUI for graphical simulation.

Scope:

- **Purpose:** Safety-critical application to monitor activities and falls.
- **Users:** Patients and doctors, each with specific functionalities.

Requirements Management

Information Gathering:

- **Methods:** Group discussions, literature reviews, and case studies.

Public Datasets:

- **Dataset Used:** MobiAct v2.0 for activity and fall detection.

MATLAB Tools:

- **Signal Processing Toolbox:** For preprocessing and analyzing sensor data.
- **Statistics and Machine Learning Toolbox:** For training and evaluating classifiers.
- **Data Acquisition Toolbox:** For real-time data processing.
- **App Designer:** For developing a user-friendly interface.
- **Parallel Computing Toolbox:** For speeding up computations.
- **Database Toolbox:** For future integration of data storage solutions.

Use Cases

Patient Use Cases:

- **View Activity Data**
- **Fall Alerts:** Notifications for detected falls with options to confirm or reject.

Doctor Use Cases:

- **Monitor Data:** View activity data of patient
- **Emergency Alerts:** Receive alerts for patient falls requiring immediate attention.

Process Model

Hybrid Model:

- **Agile-Scrum:** Iterative sprints for flexibility and continuous improvement.
- **V-Model:** Emphasis on verification and validation at each stage of development. which is crucial for a safety-critical system like an Elderly Care Monitoring System.

Benefits:

- **Flexibility:** Adapt to changes quickly.
- **Rigor:** Ensure thorough testing and reliability.

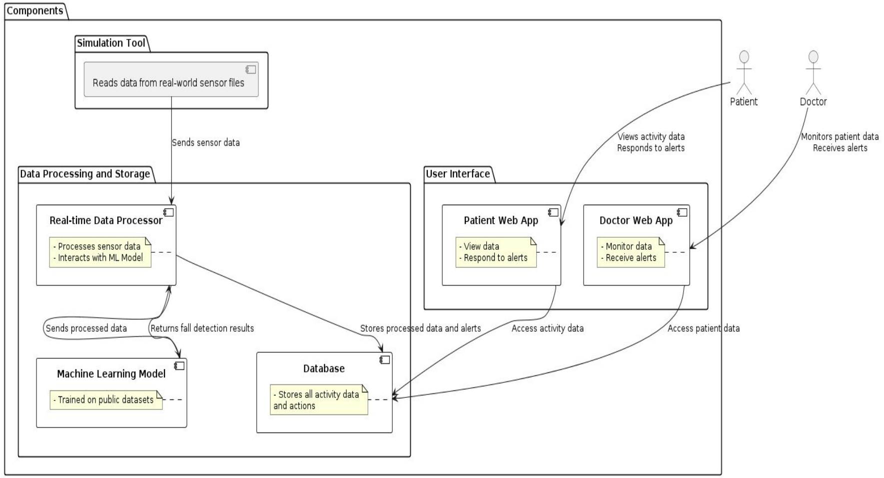
System Architecture

Actors:

- **Patient:** Views activity data and receives alerts.
- **Doctor:** Monitors patient data and receives emergency alerts.

Components:

- **Stream Data:** Read real-world sensor data.
- **Data Processing and Storage:** Real-time data processing, machine learning model interaction, and database storage.
- **Machine Learning Model:** Trained to detect falls using public datasets.
- **User Interface:** Patient and doctor web apps for data interaction.



UML Diagrams

Use Case Diagram:

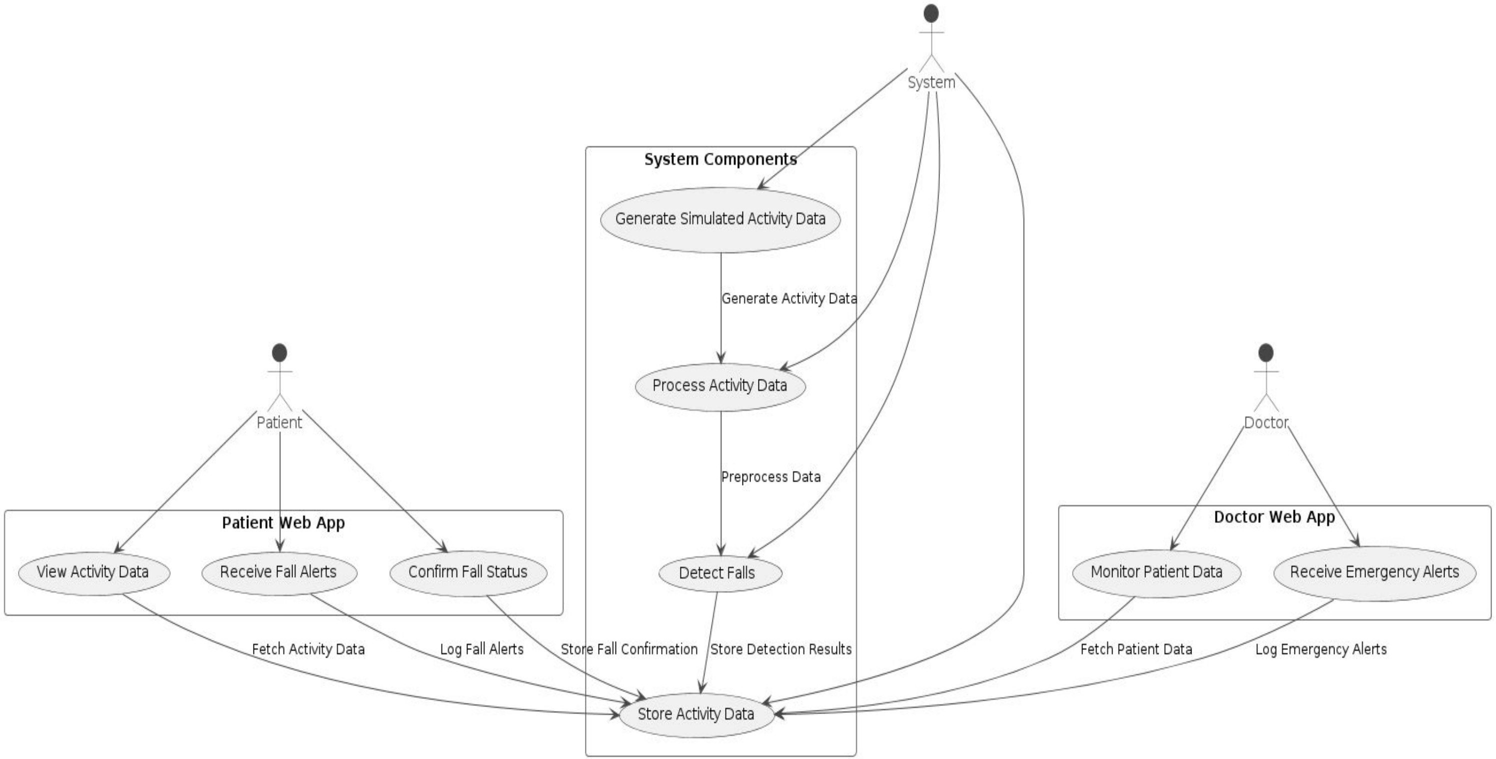
- **Description:** Shows interaction between users and the system.
- **Actors:** Patient and Doctor.

Sequence Diagram:

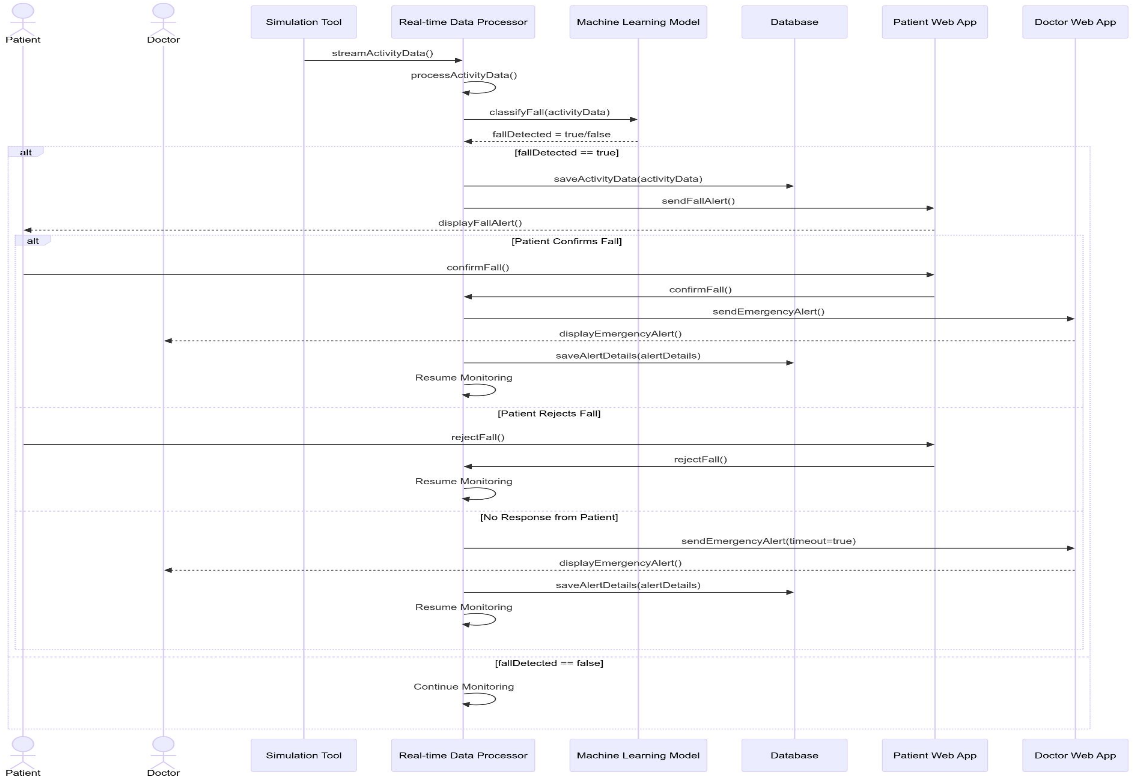
- **Flow of Data:** Streaming, processing, handling detected falls, and monitoring.

State Transition Diagram:

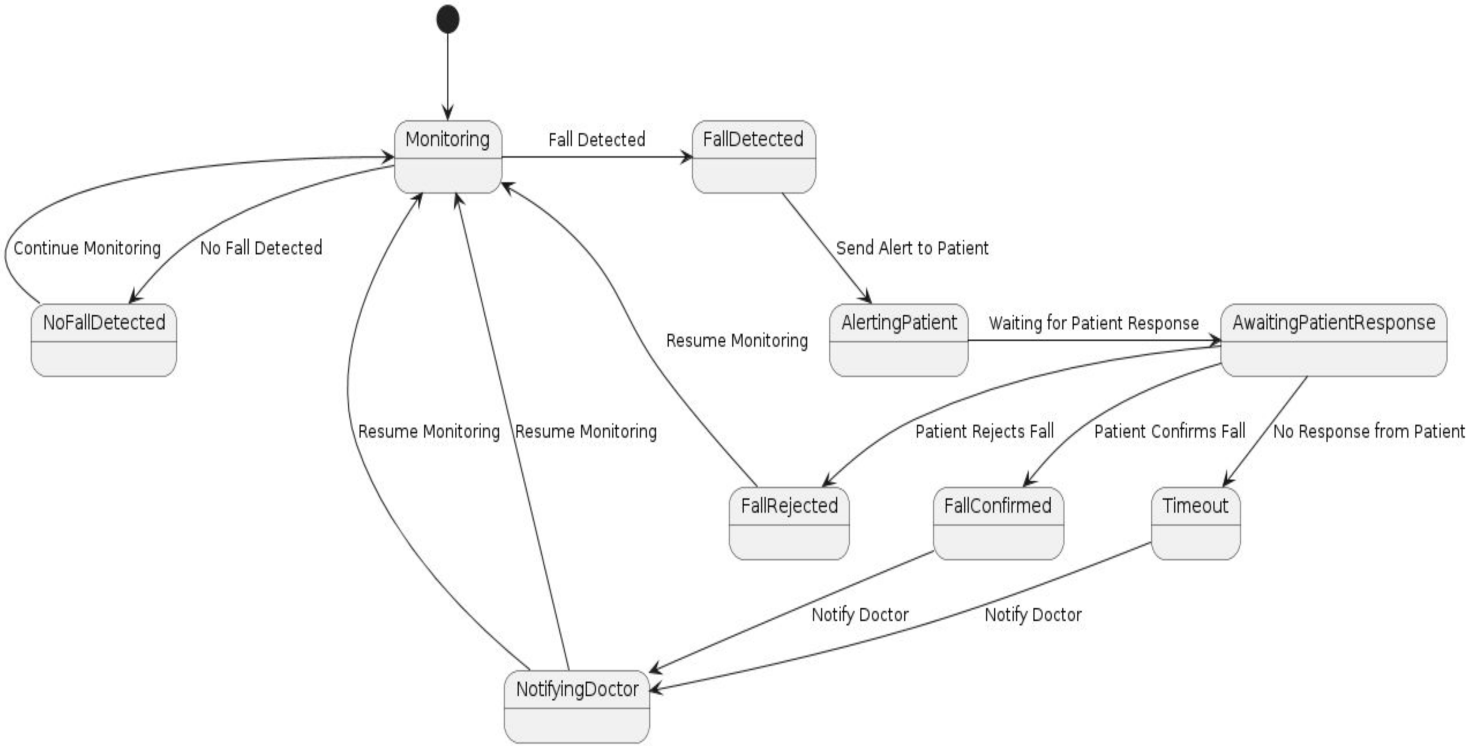
- **States:** Monitoring, Fall Detected, Alerting Patient, Awaiting Response, Fall Confirmed, Notifying Doctor, Fall Rejected, Timeout.
- **Transitions:** Detailed transitions between these states.



System
System Components
Generate Simulated Activity Data
Generate Activity Data
Process Activity Data
Preprocess Data
Detect Falls
Store Detection Results
Store Activity Data
Patient
Patient Web App
View Activity Data
Receive Fall Alerts
Confirm Fall Status
Fetch Activity Data
Log Fall Alerts
Store Fall Confirmation
Doctor
Doctor Web App
Monitor Patient Data
Receive Emergency Alerts
Fetch Patient Data
Log Emergency Alerts



Patient
Doctor
Simulation Tool
Real-time Data Processor
Machine Learning Model
Database
Patient Web App
Doctor Web App
streamActivityData()
processActivityData()
classifyFall(activityData)
fallDetected = true/false
alt
[fallDetected == true]
saveActivityData(activityData)
sendFallAlert()
displayFallAlert()
alt
[Patient Confirms Fall]
confirmFall()
confirmFall()
sendEmergencyAlert()
displayEmergencyAlert()
saveAlertDetails(alertDetails)
Resume Monitoring
[Patient Rejects Fall]
rejectFall()
rejectFall()
Resume Monitoring
[No Response from Patient]
sendEmergencyAlert(timeout=true)
displayEmergencyAlert()
saveAlertDetails(alertDetails)
Resume Monitoring
[fallDetected == false]
Continue Monitoring
Patient
Doctor
Simulation Tool
Real-time Data Processor
Machine Learning Model
Database
Patient Web App
Doctor Web App


Monitoring
Fall Detected
FallDetected
Continue Monitoring
No Fall Detected
Send Alert to Patient
NoFallDetected
AlertingPatient
Waiting for Patient Response
AwaitingPatientResponse
Resume Monitoring
Resume Monitoring
Resume Monitoring
Patient Rejects Fall
Patient Confirms Fall
No Response from Patient
FallRejected
FallConfirmed
Timeout
Notify Doctor
Notify Doctor
NotifyingDoctor

Project Estimates

Function Point Analysis (FPA):

- **Calculation:** Estimate software size and complexity.
- **Complexity Weights:** External Inputs (EI), External Outputs (EO), External Inquiries (EQ), Internal Logical Files (ILF), External Interface Files (EIF).

COCOMO II Estimation:

- **Effort Calculation:** Based on function points converted to Source Lines of Code (SLOC).
- **Summary:** 54 Function Points, 18.79 Person-Months, 10 weeks duration with 8 team members.

Function Point Calculation

1. **External Inputs (EI)**: User inputs that the system processes.
    - Patient confirmation of fall
    - Patient rejection of fall
    - Streaming activity data from the Simulation Tool
2. **External Outputs (EO)**: User outputs generated by the system.
    - Display fall alert to patient
    - Notify the doctor of the fall
    - Display emergency alert to doctor
3. **External Inquiries (EQ)**: User-initiated inquiries that do not change the system state.
    - View activity data (patient)
    - View patient data (doctor)
4. **Internal Logical Files (ILF)**: Logical groups of data maintained within the system.
    - Activity data
    - Alert details
5. **External Interface Files (EIF)**: Logical groups of data used for reference that are maintained by other systems.
    - Public datasets for training the Machine Learning Model

Assigning Complexity Weights

For simplicity, we'll assume all functions have average complexity.

Function Type	Count	Complexity Weight (Average)	Total FP
External Inputs (EI)	3	4	12
External Outputs (EO)	3	5	15
External Inquiries (EQ)	2	4	8
Internal Logical Files (ILF)	2	7	14
External Interface Files (EIF)	1	5	5

Total Function Points Calculation

Total Unadjusted Function Points (UFP) = 54

Value Adjustment Factor (VAF)

VAF = 0.65 + (sum of degree of influence) / 100

Assuming a typical degree of influence summing to 35,
VAF = 0.65 + 0.35 = 1.00

Adjusted Function Points (AFP)

AFP = UFP * VAF = 54 * 1.00 = 54

Using the COCOMO II model, we can estimate the effort required to complete the project based on the function point count.

Source Lines of Code (SLOC) Conversion

The conversion factor from function points to SLOC.

For MATLAB, a typical conversion factor is around 100 SLOC per function point.

Adjusted SLOC Calculation:

- SLOC = AFP * MATLAB Conversion Factor
- SLOC = 54 * 100 = 5.4 KSLOC

Effort Calculation

Using the COCOMO II model, Let's calculate the effort required.

COCOMO II formula:
{Effort (Person-Months)} = A * [{SLOC})^B * E]

For MATLAB and nominal parameters:

- A=2.94: This is the base constant for the COCOMO II model in nominal mode.
- B = 1.05B=1.05: This exponent reflects the scale factor.
- E = 1.0E=1.0: This indicates a nominal effort multiplier, assuming typical project attributes.

Effort = 2.94 * (5.4 KSLOC)^1.05 * 1.0 ≈ **18.79 person-months.**

Team and Duration

Given 8 team members:

Duration (Months) = Effort / Number of Team Members = 18.79 / 8 ≈ 10 Weeks

Risk Management

Identified Risks:

- **Resource Allocation Issues:** Mitigation with clear task assignments and monitoring.
- **Delays in Task Completion:** Detailed timeline and progress reviews.
- **Scope Creep:** Clear project scope and change management process.
- **Communication Breakdowns:** Regular team meetings and clear communication channels.
- **Data Security Breaches:** Follow best practices and regular security audits.
- **Usability Challenges:** Simplify UI and conduct usability testing.

Mitigation Strategies:

- **Regular Reviews:** Weekly reviews to monitor and adjust tasks and resources.

Safety Plan

STAMP/STPA Analysis:

- **Identify Unsafe Control Actions (UCAs):** Data stream inaccuracies, processing delays, data cleansing issues, fall detection errors, alert delivery failures.
- **Safety Constraints and Mitigation:** Accurate data, timely processing, robust data validation, model improvement, real-time alert delivery.

Safety Requirements:

- **Accurate Fall Detection:** Using robust algorithms and diverse training datasets.
- **Timely Alerts:** Ensure real-time alerts to patients and doctors.
- **Usability:** Design user-friendly interfaces.

Safety Measures:

- **Data Validation and Cleansing:** Implement rigorous processes.
- **Algorithm Optimization:** Continuous refinement.

Security Plan

STPA-SafeSec Analysis:

- **System Level:**
  - **Threats:** Unauthorized access, data interception.
  - **Requirements:** Authorized access, data confidentiality.
- **Component Level:**
  - **Simulation Tool:** Malicious data injection.
  - **Real-time Data Processor:** Unauthorized access.
  - **Machine Learning Model:** Model poisoning attacks.
  - **Database:** Data breaches.
  - **Web App:** Cross-site scripting attacks, data leakage.

Security Measures:

- **Input Validation:** Strict validation of inputs.
- **Role-Based Access Control (RBAC):** Restrict access.
- **Data Encryption:** Secure data transmission and storage.

Roadmap and Schedule

Key Phases and Milestones:

- **Setup and Preliminary Development:** MATLAB environment setup, simulation tool development.
- **Core System Development:** Real-time data processor and machine learning model.
- **Web Application Development:** Patient and doctor web apps.
- **Integration and Testing:** Unit testing, system testing.
- **Deployment and Evaluation:** System deployment.

Current Status:

- **Completed:** Setup, real time streaming of data, real-time data processor, machine learning model training.
- **In Progress:** web UI development.
- **Pending:** Database integration.

Implementation Details

Code Demonstration:

Challenges and Adjustments

Initial Simulation Tool:

- Planned to generate synthetic data and stream to the model.
- **Adjustment:** Switched to real-world sensor data as synthetic data didn't match real-world use cases.

Dataset Access Delay:

- Planned to use MobiAct dataset and contacted BMI lab for access.
- **Adjustment:** Used a backup dataset temporarily. Repeated preprocessing when access was granted, causing delays.

Multi-Classification to Binary Classification:

- Initially aimed for multi-classification.
- **Adjustment:** Switched to binary classification for better fall detection accuracy.

Challenges and Adjustments

LLM Model Complexity:

- Tried LSTM models.
- **Adjustment:** Dropped due to high computational requirements, opted for simpler models.

Team Reduction:

- Initially planned with 8 members.
- **Adjustment:** Two members dropped out early, continued with 6 members, affecting workload distribution.

Unfinished Components

Database Component:

- **Design:** Intended design for storing activity data and system actions.
- **Role:** Central storage for all data, supporting real-time processing.

Web UI:

- **Planned Features:** User interfaces for patients and doctors, role-based access.
- **Future Work:** Complete development, integrate with backend.

Conclusion

Summary of Achievements:

- **Core Functionalities:** Developed key components for fall detection and monitoring.

Next Steps:

- **Complete Remaining Components:** Database and web UI.
- **Further Testing:** Ensure reliability and usability.
- **Deployment:** Finalize and deploy the system for real-world use.

Q&A

References

S. Vavoulas, M. Pediaditis, E. Spanakis, and M. Tsiknakis, "The MobiAct Dataset: Recognition of Activities of Daily Living using Smartphones," International Journal of Computer Science and Information Technologies, vol. 7, no. 3, pp. 1-9, 2016.

B. Boehm, "COCOMO II Model Definition Manual," Center for Software Engineering, USC, 2000.

N. G. Leveson, "Engineering a Safer World: Systems Thinking Applied to Safety," MIT Press, 2011.

T. R. Morgan, A. S. Adams, and N. G. Leveson, "Integrating STPA with Cyber Security Analysis Using STPA-SafeSec," in Proceedings of the International Conference on Software Engineering, Montreal, Canada, 2019, pp. 120-129.

M. C. Huang, E. T. Hsiao, C. Y. Chu, and C. T. Sun, "Fall Detection Using Accelerometer and Gyroscope Sensors," in Proceedings of the International Conference on Medical Information and Communication Technology (ISMICT), La Jolla, CA, USA, 2014, pp. 1-6.

C. A. Ericson, "Hazard Analysis Techniques for System Safety," 2nd ed., Wiley, 2015.

N. G. Leveson, "A New Accident Model for Engineering Safer Systems," Safety Science, vol. 42, no. 4, pp. 237-270, April 2004.

N. G. Leveson, "Engineering a Safer World: Systems Thinking Applied to Safety," MIT Press, 2011.

K. Suzuki, S. Misaki, and K. Kawakami, "A Fall Detection System for the Elderly Using Triaxial Acceleration Sensors," in Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Osaka, Japan, 2013, pp. 3505-3508.

X. Yu, "Approaches and Principles of Fall Detection for Elderly and Patient," in Proceedings of the International Conference on HealthCom, Biopolis, Singapore, 2008, pp. 42-47.

S. J. Preece, J. Y. Goulermas, L. P. Kenney, and D. Howard, "A Comparison of Feature Extraction Methods for the Classification of Dynamic Activities from Accelerometer Data," IEEE Transactions on Biomedical Engineering, vol. 56, no. 3, pp. 871-879, March 2009.