

# Hệ thống phát hiện và cảnh báo té ngã thời gian thực tích hợp cảm biến, xử lý ảnh và định vị

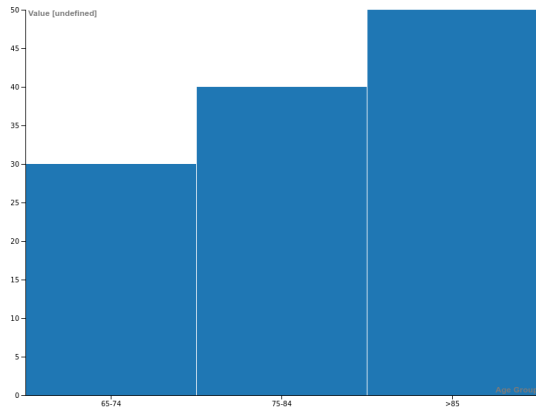
Trần Đức Hảo

Trường Đại học BK.HCM

Ngày 17 tháng 9 năm 2025

# Té ngã: Vấn đề chung toàn cầu

- Nguyên nhân chính gây chấn thương và tử vong không cố ý.
- **WHO**: ~ 646,000 ca tử vong/năm;  
> 80% ở các nước thu nhập trung bình/thấp.
- Người cao tuổi: **30%** té ngã/năm ở người > 65 tuổi, tăng lên **50%** ở người > 85 tuổi.



Hình: Tỷ lệ té ngã theo nhóm tuổi

# Tổng quan các phương pháp phát hiện té ngã

- **Dựa trên thị giác (Vision-based):** Sử dụng camera và thuật toán nhận diện tư thế người (MediaPipe, OpenPose).
- **Dựa trên cảm biến đeo (Wearable-based):** Dùng cảm biến quán tính (IMU, MPU6050) trên thiết bị.
- **Kết hợp đa phương thức (Multi-modal):** Tích hợp dữ liệu từ nhiều nguồn để tăng độ chính xác.

# Nghiên cứu trong và ngoài nước

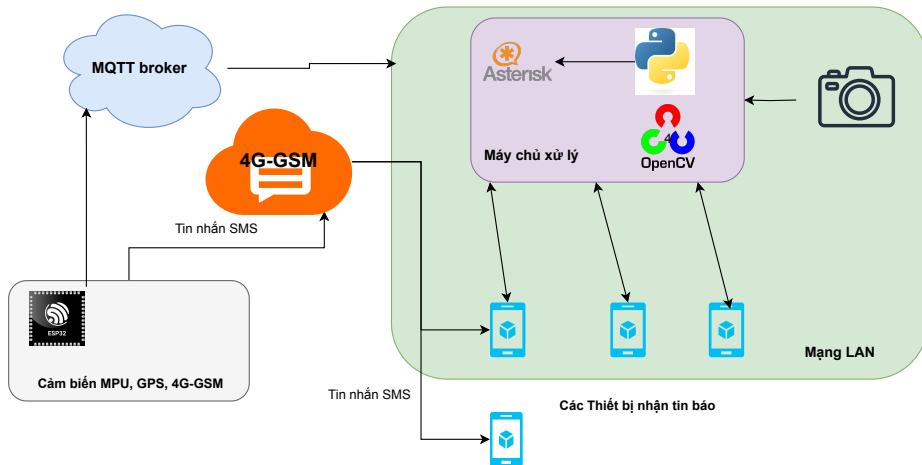
## Quốc tế

- **Xu hướng:** Sử dụng YOLO, Transformer, AI nhẹ, cảm biến mmWave.
- **Thành tựu:** Giảm false alarm, tối ưu cho thiết bị biên, Sensor Fusion.

## Trong nước

- **Thực trạng:** Chủ yếu mô hình thử nghiệm (PoC) với ESP32, Arduino.
- **Hạn chế:** Thiếu dữ liệu lớn, độ chính xác thấp (75-85%), thiếu tích hợp đa phương thức.

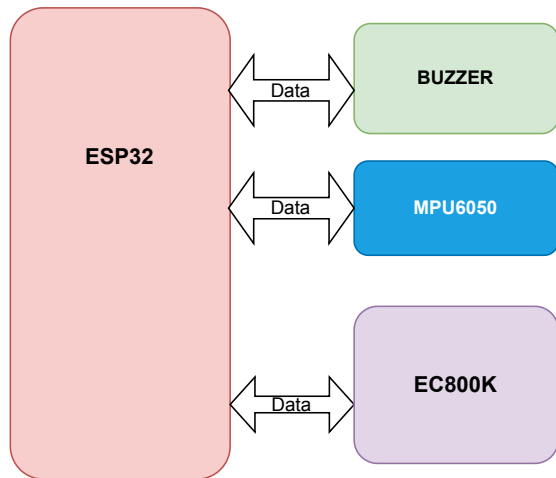
# Kiến trúc hệ thống tổng thể



Hình: Sơ đồ hệ thống tổng thể

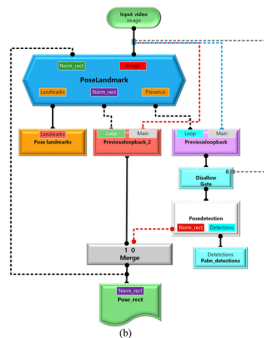
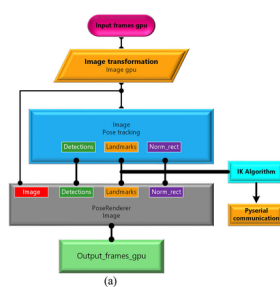
# Hệ thống nhúng (ESP32)

- **Phần cứng:** ESP32, MPU6050, GPS EC800K.
- **Nguyên lý:** Phát hiện té ngã dựa trên ngưỡng động học.
- **Giao tiếp:** Gửi cảnh báo qua **MQTT và SMS**.
- **Ưu điểm:** Thiết bị độc lập, tiết kiệm năng lượng, dễ mở rộng.



# Hệ thống phân tích hình ảnh

- **Công nghệ:** MediaPipe, OpenCV, YOLO để trích xuất các điểm khớp xương (keypoints) và phân tích tư thế.
- **Quy trình:** Phân tích góc nghiêng, vận tốc, tỉ lệ khung xương để nhận diện té ngã.
- **Thuật toán:** Sử dụng các mô hình học máy (ML) như SVM, Decision Tree.



Hình: Pipeline MediaPipe + YOLO

# Hiệu năng và Giới hạn

## Mục tiêu Hiệu năng

- Tổng độ trễ  $< 5$  giây.
- Độ chính xác  $> 90\%$ , False Alarm  $< 8\%$ .
- Uptime dịch vụ MQTT  $> 99\%$ .

## Giới hạn nghiên cứu

- Hoạt động trong nhà với điều kiện ánh sáng và mạng ổn định.
- Nguyên mẫu **ESP32** chưa tích hợp học sâu toàn phần.
- Không phát triển app di động/web phức tạp.



# Tóm tắt Giới thiệu

## Nội dung

Tổng quan

## Mô tả

Hệ thống phát hiện và cảnh báo té ngã thời gian thực tích hợp cảm biến, xử lý ảnh và định vị: Giải pháp giám sát sức khỏe chủ động cho người cao tuổi và bệnh nhân tích hợp **IMU** và **Thị giác Máy tính (CV)**.

Khoảng trống kỹ thuật

Thiếu giải pháp tích hợp **Human Pose Estimation** (MediaPipe Pose) với phần cứng nhúng chi phí thấp (**ESP32**). Kết hợp **độ chính xác cao (CV)** và **tính di động, tích hợp (IMU)**.

Mục tiêu nghiên cứu

Xây dựng hệ thống phát hiện té ngã **đáng tin cậy, hiệu quả**, phân tích sự kiện ở giai đoạn với dữ liệu **đa cảm biến**.

Tiếp theo

Chương *Cơ sở Lý thuyết*: Nguyên lý **CV**, **HPE**, **Hệ thống nhúng** cho thiết kế giải pháp.

# Tổng quan các phương pháp phát hiện té ngã

Phương pháp	Cơ chế	Ưu điểm	Nhược điểm
Đeo đeo	IMU (gia tốc kế, con quay hồi chuyển); phát hiện gia tốc/tư thế bất thường	Phản hồi nhanh; chính xác; chi phí thấp	Cần đeo liên tục; dễ false positive; pin/hiệu chuẩn
Môi trường	Cảm biến cố định: sàn áp suất, PIR, âm thanh; AI phân tích	Không xâm phạm; giám sát nhiều người; tích hợp smart home	Chi phí cao; phạm vi hạn chế; nhầm vật thể
Thị giác	Camera RGB/RGB-D/IR; pose estimation (OpenPose/MediaPipe)	Thông tin trực quan; không cần đeo; tích hợp giám sát	Quyền riêng tư; phụ thuộc ánh sáng; cần phần cứng mạnh
Đa phương thức	Kết hợp IMU + camera + môi trường; data fusion (Kalman/Deep Learning)	Độ chính xác cao; giảm cảnh báo sai; mở rộng phạm vi; kinh tế	Phức tạp; tốn năng lượng; đồng bộ khó

- Kết hợp dữ liệu để xác nhận té ngã, giảm false positive.
- Chế độ linh hoạt: In-situ (cục bộ) + Mobile (edge device).
- Bảo mật: xử lý cục bộ, chỉ gửi dữ liệu tối thiểu, tùy chỉnh khu vực nhạy cảm.

# Các Giao Thức Truyền Thông trong Hệ Thống Cảnh Báo IoT

Hệ thống phát hiện ngã với ba giao thức chính.

- **SIP**: Truyền tải âm thanh/video cảnh báo thời gian thực
- **MQTT**: Vận chuyển dữ liệu cảm biến từ thiết bị IoT
- **JSON**: Định dạng cấu trúc dữ liệu trao đổi

## Mục tiêu

Xây dựng hệ thống cảnh báo không gián đoạn, độ trễ thấp từ cảm biến đến cuộc gọi VoIP

# Giao thức SIP - Khởi tạo Phiên

## Chức năng chính:

- Thiết lập cuộc gọi VoIP từ hệ thống cảnh báo
- Kết nối với Asterisk PBX để gọi điện thoại
- Truyền âm thanh cảnh báo qua RTP

## Các bước hoạt động:

- 1 REGISTER: Đăng ký thiết bị với server
- 2 INVITE: Khởi tạo cuộc gọi cảnh báo
- 3 ACK: Xác nhận kết nối thành công
- 4 RTP: Truyền dữ liệu âm thanh
- 5 BYE: Kết thúc cuộc gọi

# Phân biệt Đường tín hiệu và Đường truyền phương tiện trong SIP

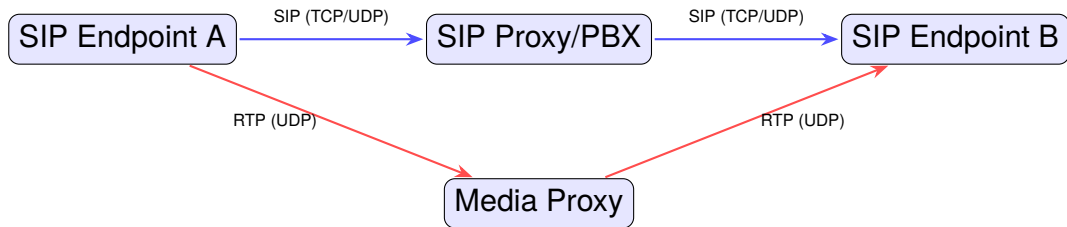
## Đường tín hiệu (Signaling Path)

- Mang các tin nhắn SIP (INVITE, BYE, 200 OK, v.v.)
- Thiết lập, quản lý và kết thúc cuộc gọi
- Sử dụng TCP hoặc UDP

## Đường truyền phương tiện (Media Path)

- Mang dữ liệu thoại/video thực tế
- Sử dụng RTP qua UDP
- Truyền trực tiếp giữa các điểm cuối

# Sơ đồ Đường tín hiệu và Đường truyền phương tiện



# Giao thức ICE (Interactive Connectivity Establishment)

## Vấn đề

Các thiết bị thường nằm sau NAT/firewall, ngăn cản truyền dữ liệu RTP trực tiếp

## Giải pháp ICE

- **Local IP:** Địa chỉ IP nội bộ của thiết bị
- **STUN:** Phát hiện địa chỉ IP công cộng và cổng NAT
- **TURN:** Máy chủ chuyển tiếp khi STUN thất bại

# SIP trong phần mềm mã nguồn mở Asterisk

## Lợi ích

- **Quản lý tập trung:** Đồng nhất cấu hình và quản lý thiết bị
- **Tương thích cao:** Hỗ trợ đa dạng nền tảng và thiết bị
- **Chuẩn mở:** Tích hợp dễ dàng với hạ tầng hiện có
- **Bảo mật:** Hỗ trợ TLS (SIP) và SRTP (RTP)

## Vai trò của Asterisk

Đóng vai trò như SIP server, xử lý đăng ký và định tuyến cuộc gọi



# Tích Hợp SMS qua SIP

## Cấu hình

- Kích hoạt textsupport=yes trong sip.conf
- Định nghĩa logic xử lý trong extensions.conf

## Thực hiện

- Sử dụng lệnh MessageSend
- Gửi tin nhắn SIP MESSAGE
- Tích hợp SMS gateway để chuyển tiếp sang mạng di động

# Giao thức MQTT - Tổng quan

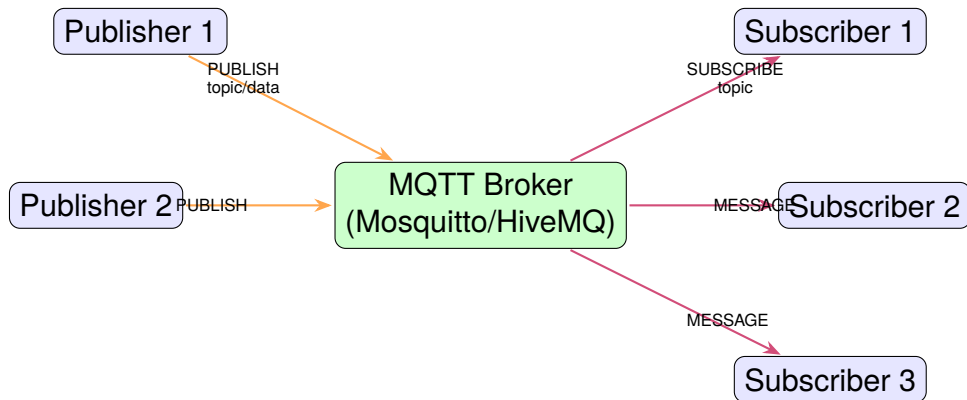
## Định nghĩa MQTT

- MQTT = Message Queuing Telemetry Transport
- Giao thức nhẹ, tối ưu cho IoT và M2M
- Hoạt động trên TCP/IP với cơ chế kết nối lâu dài

## Đặc điểm MQTT

- Thiết kế cho thiết bị có tài nguyên hạn chế
- Phù hợp với băng thông thấp, kết nối không ổn định
- Tiêu chuẩn OASIS cho IoT messaging

# Kiến trúc Publish/Subscribe của MQTT



# Lợi ích của mô hình Publish/Subscribe

Tách rời	Lợi ích
Không gian	Publisher và Subscriber không cần biết địa chỉ IP của nhau, giao tiếp qua
Thời gian	Không yêu cầu kết nối đồng thời, hỗ trợ retained messages và clean sess
Đồng bộ	Truyền/nhận độc lập, giảm độ trễ và tăng hiệu suất

# Quality of Service (QoS)

Mức	Đặc điểm	Ứng dụng
QoS 0	"Fire and forget", nhanh nhưng có thể mất message	Dữ liệu cảm biến thường
QoS 1	Đảm bảo gửi ít nhất một lần, có thể trùng lặp	Cân bằng độ tin cậy và
QoS 2	Đảm bảo gửi đúng một lần, tin cậy nhất nhưng chậm	Dữ liệu quan trọng

# Bảo mật trong MQTT

## Mã hóa Transport Layer

- Hỗ trợ TLS/SSL cho kết nối bảo mật
- MQTT over TLS (port 8883)
- Bảo vệ dữ liệu trong quá trình truyền

## Xác thực và Ủy quyền

- Username/Password authentication
- Client certificates cho xác thực mạnh
- Access Control Lists (ACL) kiểm soát quyền truy cập topic

# Các lệnh MQTT chính

## Connection Management

- CONNECT/CONNACK: Khởi tạo và xác nhận kết nối
- DISCONNECT: Ngắt kết nối graceful

## Messaging Operations

- PUBLISH: Gửi message tới topic
- PUBACK/PUBREC/PUBREL/PUBCOMP: QoS acknowledgments
- SUBSCRIBE/SUBACK: Đăng ký và xác nhận topic
- UNSUBSCRIBE: Hủy đăng ký topic

## Keep Alive

PINGREQ/PINGRESP: Duy trì kết nối

# MQTT trong Hệ thống IoT và Cảnh báo

## Ứng dụng trong IoT

- Thu thập dữ liệu sensors, điều khiển thiết bị
- Giám sát hệ thống, gửi thông báo

## Lợi ích cho hệ thống cảnh báo

- Kết nối đáng tin cậy, truyền real-time
- Hỗ trợ offline messaging, scale tốt



# Giao thức MQTT - Truyền Dữ Liệu Cảm Biến

## Đặc điểm:

- Nhẹ, tiết kiệm băng thông cho IoT
- Mô hình Publish/Subscribe qua Broker
- Hỗ trợ 3 mức QoS đảm bảo độ tin cậy

**Ví dụ topic:** `sensor/room/temperature, alert/fall/detected`

# JSON - JavaScript Object Notation

## Định nghĩa và Đặc điểm

- Định dạng dữ liệu nhẹ, dễ đọc, độc lập ngôn ngữ
- Cấu trúc cặp `key:value`, dùng cho trao đổi dữ liệu
- Phổ biến trong IoT cho lưu trữ và truyền

# Cấu trúc JSON cơ bản

## Cấu trúc dữ liệu

- **Object:** {key: value}
- **Array:** [value1, value2]
- **Kiểu giá trị:** String, Number, Boolean, null, Object, Array

## Ví dụ JSON

```
1 {  
2   "device_id": "ESP32_001",  
3   "temperature": 25.5,  
4   "sensors": ["temp", "light"]  
5 }
```

# Ứng dụng JSON trong IoT

## Lưu cấu hình và Trao đổi dữ liệu

- Cấu hình thiết bị (Wi-Fi, MQTT), thông số cảm biến
- Định dạng payload cho MQTT, API, gửi cảnh báo

# Ví dụ: Cấu hình ESP32 và Payload MQTT

```
1 {  
2   "network": {  
3     "ssid": "IoT_Network",  
4     "mqtt_broker": "192.168.1.100"  
5   },  
6   "alert": {  
7     "fall_detected": true,  
8     "timestamp": "2025-09-17T12:00:00"  
9   }  
10 }
```

# Một số Thư viện JSON cho hệ thống nhúng

## json-c và FirebaseJson

- json-c: Phân tích cú pháp, tạo JSON cho C/C++
- FirebaseJson: Dễ dùng, hỗ trợ JSON phức tạp cho IoT

# JSON trong MQTT và SIP

## Tích hợp

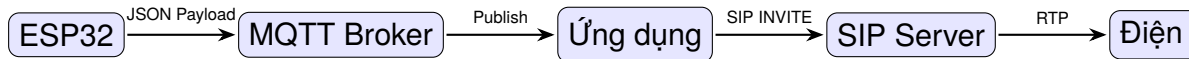
- Payload JSON trong MQTT topic sensor/data
- Dữ liệu JSON trong SIP MESSAGE hoặc custom headers
- Đảm bảo tương tác giữa các giao thức IoT

# JSON và Tích Hợp Hệ Thống

## Vai trò JSON:

- Định dạng nhẹ, tương thích, tối ưu payload

## Luồng tích hợp:



Hình: Luồng dữ liệu từ cảm biến đến cảnh báo



# Kết Luận và Tối Ưu Hóa

## Lợi ích kết hợp:

- MQTT: Thu thập dữ liệu hiệu quả
- JSON: Cấu trúc linh hoạt
- SIP: Cảnh báo âm thanh tức thì

## Tối ưu:

- Payload JSON nhỏ gọn
- QoS MQTT phù hợp
- Tự động reconnect
- Bảo mật TLS

# Định nghĩa và Mục tiêu

## Định nghĩa

**Thị giác Máy tính (CV):** Lĩnh vực AI cho phép máy tính xử lý, phân tích và diễn giải hình ảnh/video, mô phỏng thị giác con người.

## Mục tiêu

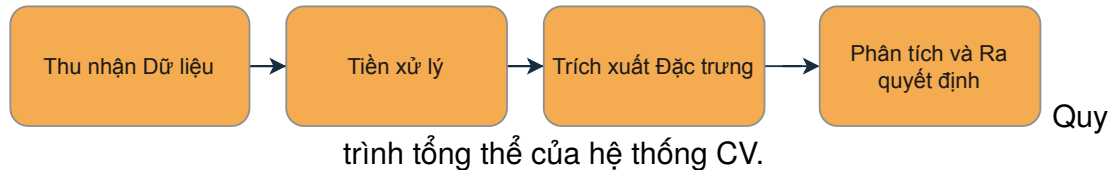
- Tái tạo khả năng nhận thức thị giác với tốc độ, độ chính xác và quy mô vượt trội.
- Ứng dụng trong Hệ thống phát hiện và cảnh báo té ngã thời gian thực tích hợp cảm biến, xử lý ảnh và định vị, đặc biệt là **Ước lượng Tư thế Người (HPE)**.

# Pipeline Cơ bản của Hệ thống CV

## Quy trình

- 1 **Thu nhận dữ liệu:** Thu thập ảnh/video từ camera.
- 2 **Tiền xử lý:** Chuẩn hóa kích thước, điều chỉnh sáng/tương phản, giảm nhiễu.
- 3 **Trích xuất đặc trưng:** Chuyển pixel thành đặc trưng trừu tượng (cạnh, góc, kết cấu).
- 4 **Phân tích và quyết định:** Phân loại, nhận dạng hoặc ước lượng tư thế.

## Minh họa



# Phân loại Bài toán CV

## Các bài toán cốt lõi

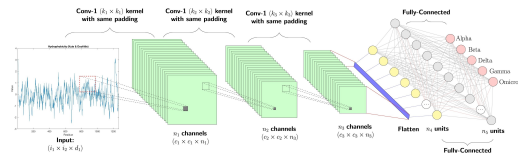
- **Phân loại Ảnh:** Gán nhãn cho toàn bộ ảnh (VD: "Người", "Xe").
- **Phát hiện Đối tượng:** Xác định vị trí và nhãn bằng hộp giới hạn.
- **Phân đoạn Ảnh:**
  - **Ngữ nghĩa:** Gán nhãn từng pixel (VD: Đường, Cây).
  - **Thể hiện:** Phân biệt các cá thể cùng lớp.
- **Ước lượng Tư thế Người (HPE):** Xác định tọa độ **khớp keypoint** để phân tích chuyển động.

# Mô hình Học sâu: CNN

- **Mạng Nơ-ron Tích chập (CNN):** Kiến trúc chủ đạo cho xử lý ảnh.
- **Phép tích chập:** Trích xuất đặc trưng cục bộ:

$$(I * K)(i, j) = \sum_m \sum_n I(i - m, j - n) K(m, n) \quad (1)$$

- **Phép gộp:** Giảm kích thước, tăng tính bền vững (VD: Max Pooling).



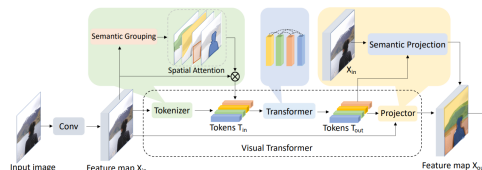
Hình: Phép tích chập và gộp.

# Mô hình Học sâu: Vision Transformer

- **Vision Transformer (ViT):** Chia ảnh thành **miếng vá**, xử lý như token.
- **Tự chú ý (Self-Attention):**

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \quad (2)$$

- Học quan hệ toàn cục, vượt giới hạn của CNN.



Hình: Kiến trúc ViT.

# Tập dữ liệu và Metrics Đánh giá

## Tập dữ liệu

- **ImageNet**: Phân loại ảnh (>14 triệu ảnh).
- **COCO**: Phát hiện, phân đoạn đối tượng.
- **MPII, COCO Keypoints**: Ước lượng tư thế người (HPE).

## Metrics đánh giá

- **IoU**: Đo độ trùng khớp hộp giới hạn.
- **mAP**: Trung bình độ chính xác cho phát hiện đối tượng.
- **F1-score**: Cân bằng Precision và Recall.
- **OKS**: Đo độ chính xác khớp trong HPE.

# Nhận diện Tư thế Người và Phát hiện Té ngã

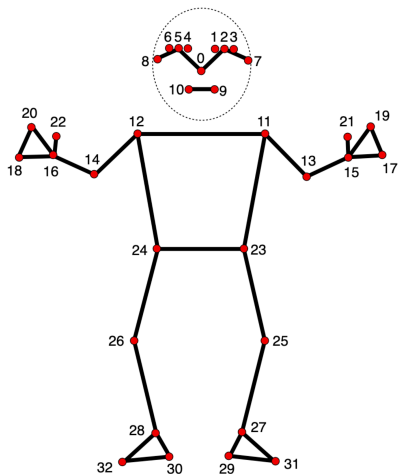
## Tổng quan

Hệ thống tích hợp nhận diện tư thế (MediaPipe Pose) và phát hiện té ngã dựa trên đặc trưng động học/tư thế.

- Ứng dụng: Giám sát an toàn, phát hiện té ngã.
- Nền tảng: Thị giác máy tính thời gian thực.



# Nhận diện Tư thế Người



## Khái niệm

Ước lượng vị trí khớp từ hình ảnh/video:

$$\mathcal{K} = \{k_i = (x_i, y_i, z_i, c_i)\}$$

( $c_i$  là confidence score cho mỗi keypoint).

## Phương pháp

- **Top-down:** Phát hiện người trước, sau đó keypoints (MediaPipe).
- **Bottom-up:** Keypoints trước, nhóm thành người sau (OpenPose).

Keypoints cơ bản trong HPE (Human

# MediaPipe Pose – Kiến trúc BlazePose

## Kiến trúc BlazePose

BlazePose tối ưu HPE 3D với:

- **Nodes:** Các module xử lý tín hiệu hình ảnh.
- **Edges:** Luồng dữ liệu đồng bộ giữa các module.

*(Nodes = các bước tính toán; Edges = kết nối dữ liệu giữa các bước).*

# MediaPipe Pose – Thành phần Hậu xử lý

## Thành phần chính

- **Detection:** ROI từ ảnh RGB, phát hiện người.
- **Landmark:** 33 keypoints 3D, Loss:  $\mathcal{L} = \sum \lambda_i \mathcal{L}_i$ .
- **Tracking:** Dự đoán vị trí ROI cho khung tiếp theo.

*(Landmark 3D giúp đánh giá tư thế và động tác).*

## Hậu xử lý

- **One Euro Filter:** Làm mịn nhiễu trong dữ liệu keypoints.
- **Chuẩn hóa  $z$ :** Dựa trên hông, tăng độ chính xác 3D.

# Thuật toán Phát hiện Té ngã

## Đặc trưng Động học

- **Vận tốc COM:**  $\vec{v}_{\text{COM}} = \frac{\Delta \vec{p}}{\Delta t}$
- **Gia tốc:**  $a = \frac{\|\Delta \vec{v}\|}{\Delta t}$

## Đặc trưng Tư thế

- **AR (Aspect Ratio):** Tăng khi người nằm ngang
  - $\theta_{\text{body}}$ : Góc vai-hông
  - $\Delta h_{\text{head}}$ : Giảm chiều cao đầu
- (AR,  $\theta$ ,  $\Delta h$  giúp xác định tư thế bất thường).*

## Ba Giai đoạn Phát hiện

- 1 **Sớm:** Tốc độ/gia tốc COM cao
- 2 **Xác nhận:** AR,  $\theta_{\text{body}}$  chỉ nằm ngang
- 3 **Bất động:** Chuyển động  $< M_{th}$

# Thuật toán Phát hiện Té ngã

## Đặc trưng Động học

- **Vận tốc COM:**  $\vec{v}_{\text{COM}} = \frac{\Delta \vec{p}}{\Delta t}$
- **Gia tốc:**  $a = \frac{\|\Delta \vec{v}\|}{\Delta t}$

## Đặc trưng Tư thế

- **AR (Aspect Ratio):** Tăng khi người nằm ngang
  - $\theta_{\text{body}}$ : Góc vai-hông
  - $\Delta h_{\text{head}}$ : Giảm chiều cao đầu
- (AR,  $\theta$ ,  $\Delta h$  giúp xác định tư thế bất thường).*

## Ba Giai đoạn Phát hiện

- 1 **Sớm:** Tốc độ/gia tốc COM cao
- 2 **Xác nhận:** AR,  $\theta_{\text{body}}$  chỉ nằm ngang
- 3 **Bất động:** Chuyển động  $< M_{th}$

# Tổng quan Kiến trúc Hệ thống Phát hiện Té ngã

## Phân loại hệ thống

- **Dựa trên Camera:** Xử lý hình ảnh cố định, yêu cầu máy chủ mạnh
- **Dựa trên Thiết bị đeo:** Cảm biến IMU, ESP32, truyền thông di động

## Ba thành phần cốt lõi

- 1 **Thiết bị Thu thập Dữ liệu:** IMU, Camera, GPS
- 2 **Máy chủ/Xử lý:** Phân tích dữ liệu, Học sâu
- 3 **Truyền thông:** Wi-Fi, 4G/LTE đảm bảo kết nối

# Vi điều khiển ESP32

## Đặc điểm chính

- Lõi kép Xtensa LX6, FreeRTOS
- Wi-Fi + Bluetooth tích hợp
- Hỗ trợ MQTT, HTTP

## Phân công nhiệm vụ

- **Lỗi 1:** Xử lý thời gian thực (IMU, Kalman Filter)
- **Lỗi 2:** Truyền thông không dây

Lỗi 1

Lỗi 2

**ESP32**

# Cảm biến IMU và GPS

## IMU

- Gia tốc kế  $a = [a_x, a_y, a_z]$
- Con quay hồi chuyển  $\omega = [\omega_x, \omega_y, \omega_z]$
- Từ kế – xác định hướng
- Fusion: Kalman/Madgwick

## GPS

- Module NEO-6M / EC800K
- Định vị NMEA, tọa độ cứu hộ
- Kết hợp truyền thông SMS/4G



# Thuật toán Phát hiện Té ngã

## Shock Event

$$\|a\| > a_{\text{shock}}$$

Gia tốc tăng vọt đột ngột

## Post-fall State

- Gia tốc  $\approx 1g$  (nằm yên)
- Tốc độ góc thay đổi lớn

# Xử lý tại Biên (Edge) và Máy chủ

## Edge (ESP32)

- Xử lý IMU thời gian thực
- Phát hiện té ngã sơ cấp
- Truyền dữ liệu JSON/MQTT

## Cloud/Server

- Xử lý ảnh từ Camera (ESP32-S3 + OV5640)
- TensorFlow/PyTorch, OpenCV

# Hệ thống Truyền thông

## Wi-Fi (chính)

- Truyền tải dung lượng lớn (ảnh/video)
- MQTT với máy chủ
- Độ trễ thấp

## 4G/LTE (dự phòng)

- SMS/cuộc gọi khẩn
- Định vị GPS
- Hoạt động khi Wi-Fi lỗi

# Logic Hoạt động Hệ thống

- 1 ESP32 thu thập dữ liệu IMU/Camera
- 2 Phát hiện té ngã sơ cấp tại biên
- 3 Truyền dữ liệu lên máy chủ (ưu tiên Wi-Fi, dự phòng 4G)
- 4 Máy chủ xử lý tin và phát cảnh báo
- 5 Kích hoạt cảnh báo (SMS/cuộc gọi)

# Môi trường Phát triển (ESP-IDF)

## Đặc trưng của ESP-IDF

- **Build system CMake + Kconfig** – cấu hình linh hoạt, dễ mở rộng component
- **FreeRTOS tích hợp sẵn** – quản lý đa nhiệm trên 2 lõi Xtensa
- **Driver cấp thấp** – I2C, SPI, UART, PWM, GPIO được tối ưu cho ESP32
- **Hỗ trợ mạng phong phú** – Wi-Fi, Bluetooth, TCP/IP stack, MQTT, HTTP(S)

## Lợi ích cho hệ thống Phát hiện Té ngã

- Quản lý **đa component** (IMU, SIM4G, LED, Fall Logic) độc lập
- Thực thi **song song**: lõi 1 xử lý cảm biến, lõi 2 lo truyền thông
- Hỗ trợ **OTA update** để nâng cấp firmware từ xa
- Debug chuyên nghiệp: `idf.py monitor`, `gdbstub`, logging

# 06<sub>background</sub><sub>summary</sub>

# 01<sub>a</sub>*rchitecture*

# 02<sub>hardware</sub>



# 03<sub>s</sub>oftware<sub>o</sub>verview

# 04<sub>m</sub>odule<sub>a</sub>st

# 05<sub>module<sub>i</sub></sub>

# 06<sub>m</sub>odule<sub>i</sub>

# 07<sub>s</sub>erver

# 08<sub>m</sub>*ethod*<sub>s</sub>*ummary*

# 01<sub>setup</sub>

## 02<sub>results<sub>detection</sub></sub>



# 03<sub>latency</sub>

# 04<sub>s</sub>*tability*

# 05\_results\_summary

# 01 *summary*

## 02<sub>c</sub>ontribution

## 03 *imitations*

# 04<sub>futurework</sub>

# 05<sub>o</sub>verall



# 01<sub>t</sub>*hankyou*