Enhanced Sit-up Counter Application Documentation

This document provides comprehensive documentation for the Enhanced Sit-up Counter application, covering all technical aspects, features, and usage instructions.

# Table of Contents

1. Project Overview

2. System Architecture

3. Technical Specifications

4. Core Components

5. Features and Capabilities

6. Installation and Setup

7. Usage Instructions

8. API Endpoints

9. Configuration Parameters

10. Troubleshooting

11. Performance Considerations

12. Future Enhancements

# 1. Project Overview

The Enhanced Sit-up Counter is a real-time computer vision application that uses MediaPipe pose detection to automatically count sit-up repetitions. The system is built with FastAPI for web interface and provides a robust, user-friendly solution for fitness tracking.

## Key Features

• Real-time pose detection and tracking

• Automatic sit-up counting with fake rep detection

• Web-based interface with live video streaming

• Enhanced detection for low-light conditions

• Multiple detection strategies for improved accuracy

• Movement quality assessment

# 2. System Architecture

## Technology Stack

Backend Framework: FastAPI (Python)

Computer Vision: OpenCV, MediaPipe

Pose Detection: MediaPipe Pose

Web Server: Uvicorn

Image Processing: NumPy, OpenCV

Real-time Streaming: MJPEG over HTTP

## Architecture Components

The application follows a client-server architecture:

• Web Browser: Displays live stream and provides user interface

• FastAPI App: Handles pose detection, rep counting, and web interface

• Camera Input: Captures and processes video frames

# 3. Technical Specifications

## System Requirements

Operating System: Linux (tested on Ubuntu 6.12.10)

Python Version: 3.7+

Camera: USB webcam or built-in camera

Memory: Minimum 4GB RAM

Processing: Multi-core CPU recommended

## Dependencies

fastapi>=0.68.0

uvicorn>=0.15.0

opencv-python>=4.5.0

mediapipe>=0.8.0

numpy>=1.21.0

# 4. Core Components

## EnhancedPoseDetector Class

Handles all pose detection and landmark processing with multiple detection strategies for improved accuracy.

### Key Methods

• findPose(img, draw=True): Main pose detection method

• findPosition(img, draw=True): Extracts landmark positions

• findAngle(img, p1, p2, p3, draw=True): Calculates joint angles

• getDetectionQuality(): Assesses detection quality

• preprocess\_image(img): Enhances images for better detection

## RepCounter Class

Manages sit-up counting logic with fake rep detection and movement quality assessment.

### Key Methods

• updateState(angle, detection\_quality): Main counting logic

• \_isValidRep(duration, detection\_quality): Validates rep quality

• \_calculateMovementSmoothness(): Assesses movement smoothness

• getStats(): Returns current statistics

• reset(): Resets counter state

# 5. Features and Capabilities

## Advanced Pose Detection

• Multi-model Detection: Primary and fallback detection models

• Low-light Enhancement: CLAHE and noise reduction

• Robust Landmark Tracking: 33 MediaPipe pose landmarks

• Visibility Assessment: Quality scoring for each landmark

## Intelligent Rep Counting

• Fake Rep Detection: Prevents counting invalid movements

• Movement Quality: Assesses exercise form and smoothness

• Timing Validation: Ensures proper rep duration

• State Management: Tracks exercise progression

## Posture Discrimination

• Sit-up vs Squat Detection: Distinguishes between exercise types

• Lying Position Validation: Ensures proper starting position

• Movement Pattern Recognition: Identifies sit-up movement patterns

• Automatic Reset: Resets when standing up

## Real-time Feedback

• Live Video Stream: Real-time pose visualization

• Visual Indicators: Color-coded landmarks and connections

• Debug Information: Shows detection parameters

• Status Messages: User feedback and instructions

# 6. Installation and Setup

## Automated Installation

1. Run the installation script: ./install.sh

2. Follow the on-screen instructions

3. The script will create a virtual environment and install all dependencies

## Manual Installation

1. Create virtual environment: python3 -m venv situp\_counter\_env

2. Activate environment: source situp\_counter\_env/bin/activate

3. Install dependencies: pip install -r requirements.txt

4. Run application: python3 fast\_api\_integrate.py

## Access Application

1. Open web browser

2. Navigate to http://localhost:8000

3. Allow camera access when prompted

# 7. Usage Instructions

## Performing Sit-ups

1. Positioning: Lie down on your back with knees bent

2. Camera Setup: Ensure full body is visible in camera frame

3. Starting Position: Begin in down position (lying flat)

4. Exercise: Perform sit-ups with controlled movement

5. Counting: System automatically counts valid repetitions

## Interface Controls

• Clear Count: Reset counter to zero

• Get Stats: View current statistics

• Live Stream: Real-time video feed with pose detection

## Visual Feedback

• Green Landmarks: High visibility landmarks

• Orange Landmarks: Medium visibility landmarks

• Red Landmarks: Low visibility landmarks

• Angle Display: Shows current joint angle

• Quality Score: Detection quality indicator

# 8. API Endpoints

**GET /**: Returns HTML dashboard with live video stream and controls

**GET /video\_feed**: Provides MJPEG video stream with pose detection overlay

**POST /clear**: Resets the sit-up counter to zero

**GET /stats**: Returns current counter statistics

## Response Examples

Clear Count Response:

{"message": "Count cleared", "count": 0}

Statistics Response:

{"count": 15, "state": "up", "detection\_quality": 0.85, "last\_rep\_time": 1640995200.0}

# 9. Configuration Parameters

## Rep Counting Parameters

DOWN\_THRESHOLD: 150 degrees for down position

UP\_THRESHOLD: 110 degrees for up position

MIN\_REP\_DURATION: 0.3 seconds

MAX\_REP\_DURATION: 5.0 seconds

MIN\_TIME\_BETWEEN\_REPS: 0.5 seconds

fake\_rep\_threshold: 0.5 minimum quality score

MIN\_MOVEMENT\_QUALITY: 0.3 minimum movement smoothness

## Pose Detection Parameters

model\_complexity: 2 (primary), 1 (fallback)

smooth\_landmarks: True (primary), False (fallback)

min\_detection\_confidence: 0.3 (primary), 0.2 (fallback)

min\_tracking\_confidence: 0.3 (primary), 0.2 (fallback)

# 10. Troubleshooting

## Camera Not Detected

Symptoms: "System Initialization Failed" message

Solutions:

• Check camera connection

• Verify camera permissions

• Try different camera index (0, 1, 2)

## Poor Detection Quality

Symptoms: Low quality scores, missed reps

Solutions:

• Improve lighting conditions

• Ensure full body is visible

• Check camera positioning

• Clean camera lens

## False Positives

Symptoms: Counting non-sit-up movements

Solutions:

• Adjust posture detection parameters

• Ensure proper lying position

• Check for interference in camera view

## Performance Issues

Symptoms: Low FPS, laggy video

Solutions:

• Close other applications

• Reduce camera resolution

• Use lower model complexity

• Check system resources

## Debug Information

The application provides real-time debug information:

• Angle: Current joint angle

• Quality: Detection quality score

• Hip-Leg diff: Height difference between hips and legs

• Sit-up mode: Whether sit-up pattern is detected

• FPS: Current frame rate

# 11. Performance Considerations

## Resource Usage

CPU: Moderate to high usage during pose detection

Memory: ~500MB-1GB RAM usage

GPU: Optional acceleration with compatible hardware

Network: Minimal usage for local deployment

## Optimization Strategies

• Model Complexity: Balance accuracy vs performance

• Frame Processing: Optimize image preprocessing

• Memory Management: Regular garbage collection

• Camera Settings: Optimize resolution and FPS

# 12. Future Enhancements

## Cloud Deployment & Mobile Integration

The next major phase of development will focus on deploying the pose detection model to cloud servers and creating a comprehensive mobile application for real-time fitness tracking.

• Cloud Server Deployment: Alibaba Cloud or Tencent Cloud hosting for scalable model serving

• Flutter Mobile Application: Cross-platform mobile app for iOS and Android

• Real-time Video Processing: Phone camera to server pipeline for live pose detection

• JPEG Frame Processing: Efficient image transmission and processing between mobile and server

• Live Video Streaming: Real-time pose detection results displayed on mobile devices

• Cloud-based Model Serving: Scalable pose detection API with load balancing

• Multi-user Support: Concurrent user processing and session management

• Offline Capability: Local processing when server unavailable

## Technical Architecture Improvements

• GPU Acceleration: CUDA/OpenCL support for cloud servers to improve processing speed

• Machine Learning: Improved detection models with cloud-based training capabilities

• Real-time Analytics: Advanced performance metrics and user insights collection

• API Expansion: RESTful API for mobile app integration with authentication

• Load Balancing: Multiple server instances for scalability and high availability

• Database Integration: User data storage, workout history, and analytics

• CDN Integration: Global content delivery for mobile apps worldwide

• Microservices Architecture: Modular cloud deployment for better maintainability

## Mobile App Features

• Cross-platform Support: iOS and Android compatibility using Flutter framework

• Real-time Camera Integration: Live video capture and processing on mobile devices

• Cloud Communication: Secure API communication with cloud servers

• Offline Mode: Local processing capabilities when internet is unavailable

• User Authentication: Secure login and profile management system

• Workout History: Personal exercise tracking and detailed statistics

• Social Features: Share workouts and compete with friends

• Push Notifications: Workout reminders and achievement notifications

## Cloud Infrastructure Specifications

• Alibaba Cloud ECS: Elastic Compute Service for model hosting with GPU instances

• Tencent Cloud CVM: Cloud Virtual Machine for processing with auto-scaling

• Load Balancer: Traffic distribution across multiple server instances

• Auto Scaling: Dynamic resource allocation based on user demand

• Object Storage: Video and image storage using Alibaba OSS or Tencent COS

• CDN Services: Global content delivery for mobile apps worldwide

• Database Services: User data and analytics storage with high availability

• API Gateway: Secure API management, rate limiting, and monitoring

## Implementation Roadmap

1. Phase 1: Cloud Server Setup - Deploy pose detection model to Alibaba/Tencent Cloud

2. Phase 2: API Development - Create RESTful APIs for mobile app communication

3. Phase 3: Flutter App Development - Build cross-platform mobile application

4. Phase 4: Real-time Integration - Implement live video streaming and processing

5. Phase 5: User Management - Add authentication, profiles, and data storage

6. Phase 6: Performance Optimization - Implement caching, CDN, and load balancing

7. Phase 7: Testing & Deployment - Comprehensive testing and production deployment

# Conclusion

The Enhanced Sit-up Counter application provides a robust, user-friendly solution for automated fitness tracking. With its advanced pose detection, intelligent rep counting, and comprehensive web interface, it offers a complete solution for sit-up exercise monitoring.

The system's modular architecture, extensive error handling, and configurable parameters make it suitable for both individual use and potential commercial deployment.