AI Virtual Personal Fitness Coach — Technical Report

Introduction

The AI Virtual Personal Fitness Coach is a computer-vision-driven application that detects human pose, counts exercise repetitions, and provides real-time technique feedback. The system is designed for at-home fitness users who want objective, immediate guidance without specialized hardware. It leverages MediaPipe for robust body landmark detection and OpenCV for image processing, exposing a user-friendly web interface via Streamlit and Streamlit-WebRTC for real-time video.

The primary objectives are:

- Enable reliable pose estimation in consumer lighting and camera conditions
- Accurately count repetitions for supported exercises (pushups, squats, planks, lunges, burpees)
- Deliver actionable, low-latency feedback on form quality
- Log workout sessions and visualize progress over time

Abstract

This project implements a lightweight, real-time fitness coaching pipeline using Python. Pose detection is performed with MediaPipe, and key joint angles are computed to infer exercise state transitions (rest → down → up). A rule-based analysis layer evaluates range of motion and posture alignment to produce a per-rep form score and feedback messages. The application supports two interaction modes: a photo-based analyzer for quick feedback and a continuous real-time video mode for live workouts. Sessions are logged and visualized, enabling users to track performance and improvements.

The system maintains interactive throughput on commodity hardware while providing stable rep counting across a core exercise set (pushups, squats, planks, lunges, burpees). Configurable thresholds and temporal smoothing help mitigate jitter and double-counting. The resulting tool offers an accessible baseline for computer-vision fitness coaching, with a clear path to future ML-based personalization.

Tools Used

- Python 3.8+
- OpenCV (opencv-python): frame I/O, image preprocessing, drawing overlays
- MediaPipe: human pose estimation and landmark tracking
- Streamlit: web UI scaffolding and layout
- Streamlit-WebRTC: low-latency real-time webcam streaming to the app
- NumPy: vectorized math for angles and smoothing
- Pandas: session logging and tabular aggregation
- Plotly / Matplotlib / Seaborn: progress charts and analytics
- Pillow: basic image handling
- av: WebRTC media handling

Steps Involved in Building the Project

1. Requirements and Design

- Defined supported exercises: pushups, squats, planks, lunges, burpees
- Established core metrics: joint angles, rep count, phase detection, form score
- Chosen architecture: MediaPipe-based pose → rule-based analysis → UI overlays → logging
- Identified two UX modes: photo-based analysis (`app.py`) and real-time streaming (`realtime_app.py`)

2. Environment Setup

- Created a virtual environment and installed dependencies from `requirements.txt`
- Verified camera access and WebRTC connectivity in Streamlit

3. Pose Detection Core (`pose_detector.py`)

- Initialized MediaPipe Pose with appropriate model complexity and confidence thresholds
- Extracted landmark coordinates and confidence for key joints (shoulders, elbows, wrists, hips, knees, ankles)
- Implemented joint angle computation using vector geometry; derived angles and distances:

    - Elbow, shoulder, hip, knee angles (deg)
    - Hip–shoulder–ankle alignment and trunk inclination
    - Foot/step stance width for lunges and burpees
- Added configurable thresholds per exercise (example baseline ranges):
    - Pushup: elbow down <~ 90°, up >~ 160°; trunk deviation <~ 10–15°
    - Squat: knee down <~ 70–80°, up >~ 160°; knee-over-toe alignment within tolerance
    - Plank: elbow/shoulder locked, hip angle >~ 160–180°, trunk deviation <~ 10°
    - Lunge: front-knee down <~ 80–90°, hip depth threshold, upright torso
    - Burpee: composite phases (squat → plank → pushup optional → jump) with per-phase validity checks
- Implemented temporal filtering (moving average/min–max over sliding window) and cooldowns to reduce jitter and duplicate counts

4. Exercise Logic and Rep Counting

- Built finite-state machines per exercise with transition guards, cooldowns, and confidence checks
- Counted a rep on valid completion of state cycles; computed per-rep metrics (min/max angles, tempo, ROM)

Per-exercise state machines:

- Pushup: `rest → down (elbow angle decreases) → up (elbow angle increases past threshold)` → count
- Squat: `stand → descend (knee/hip angle decreases) → ascend (angles increase past threshold)` → count
- Plank: timed hold with stability criteria; count via duration or interval targets (e.g., 30 s)
- Lunge: `stand → descend (front-knee/hip depth) → ascend` per leg; optional side-tracking
- Burpee: `stand → squat → plank (optional pushup) → return → jump`; count on full cycle completion

5. Real-time Feedback and UI (`realtime_app.py`, `app.py`)

- Overlayed pose skeleton and metrics using OpenCV drawing functions
- Provided real-time textual cues:
    - Pushup: "Lower more", "Keep body straight", "Lock out at top"

    - Squat: "Go deeper", "Knees out", "Chest up"
    - Plank: "Hips up", "Hips down", "Maintain straight line"
    - Lunge: "Step longer", "Knee over ankle", "Upright torso"
    - Burpee: "Full plank", "Explosive jump", "Control descent"
- Exposed controls to select exercise type and start/stop sessions
- Implemented two run modes:
    - Photo-based: single-frame capture and analysis with immediate feedback
    - Continuous video: live video with on-the-fly analysis and rep counting via WebRTC

6. Data Logging and Visualization (`workout_logger.py`)

- Logged session metadata: timestamps, exercise type, per-rep counts, and scores
- Saved to CSV/JSON for portability
- Implemented progress dashboards using Plotly/Seaborn for trends (weekly/monthly summaries, total reps, average form score)
- Added per-exercise summaries (e.g., best form score, average tempo, time-in-plank)

7. Testing and Validation (`test_pose_detector.py`)

- Unit tests for angle computation and state transitions across edge cases
- Exercise-specific tests for thresholds and phase detection (pushup, squat, plank, lunge, burpee)
- Regression checks for threshold updates to prevent drift in counting accuracy
- Manual validation on varied webcam positions and lighting conditions

8. Configuration and Tuning (`config.py`)

- Centralized thresholds (angles, cooldown timing, confidence) per exercise for easy adjustment
- Provided sane defaults and documented how to adapt for different users or cameras
- Example structure:

```
ANGLE_THRESHOLDS = {
    "pushup": {"down": 90, "up": 160, "trunk_dev_deg": 12},
    "squat": {"down": 75, "up": 160, "knee_valgus_tol_deg": 15},
    "plank": {"min_hip_angle": 165, "max_trunk_dev_deg": 10},
    "lunge": {"down": 85, "up": 160, "torso_upright_deg": 15},
    "burpee": {
        "squat_angle": 90,
        "plank_hold_ms": 200,
        "pushup_optional": True,
        "jump_min_extension_deg": 165
    }
}
```

9. Deployment and Documentation (`README.md`)

- One-command run for each mode via Streamlit
- Usage instructions, troubleshooting, and future enhancement roadmap

Exercise Specifications

Pushups

- Detection: elbow and shoulder angles; trunk alignment (shoulder–hip–ankle)
- Form checks: full depth, locked-out top, straight body line
- Rep logic: `down → up` with cooldown and min-depth enforcement
- Metrics: min elbow angle, tempo, trunk deviation, form score

Squats

- Detection: knee and hip angles; ankle–knee–hip alignment; torso pitch
- Form checks: adequate depth, knees tracking over toes, chest up
- Rep logic: `descend → ascend`; optional pause-at-bottom validation
- Metrics: min knee/hip angles, stance width, tempo, form score

Planks

- Detection: hip angle and trunk deviation stability over time
- Form checks: straight line from shoulders to ankles, minimal sway
- Rep logic: time-based hold with continuous stability checks
- Metrics: hold duration, average deviation, stability score

Lunges

- Detection: front-knee angle, hip depth, torso uprightness; side differentiation
- Form checks: knee over ankle, adequate depth, balanced stance
- Rep logic: per-leg `descend → ascend`; optional alternating-leg tracking
- Metrics: per-leg reps, min knee angle, torso pitch, balance consistency

Burpees

- Detection: composite phases with pose validity (squat, plank, optional pushup, jump)
- Form checks: full plank extension, controlled transitions, full extension on jump
- Rep logic: full cycle completion with phase ordering and time constraints
- Metrics: cycle time, jump extension, optional pushup depth, consistency

Conclusion

The AI Virtual Personal Fitness Coach demonstrates a pragmatic, real-time approach to computer-vision exercise analysis using commodity hardware and open-source libraries. With MediaPipe-based pose estimation and rule-driven form assessment, the system reliably counts reps and offers actionable feedback across pushups, squats, planks, lunges, and burpees. The Streamlit/WebRTC interface enables accessible deployment in a browser with minimal setup, while also supporting logging and visualization for ongoing user progress tracking.

Future work includes expanding datasets and calibration for more body types, learning-based form scoring and personalization, voice feedback, and integration with wearables. These enhancements can increase robustness across diverse environments, further improving coaching quality and user engagement.