Design Overview

1. System Purpose

What is the system doing?

This system classifies **static and near-static two-hand gestures** from live camera input using **pose-based hand landmarks** extracted via MediaPipe.

Rather than recognizing gestures from raw images, the system operates on **geometric representations of the hands**, capturing:

- finger placement
- joint angles and relative positions
- symmetry and spatial relationships between both hands

Each gesture is classified into one of the following categories:

- **Positive (recognized gesture)**
- **Negative (explicit non-gesture / reject)**
- **Unidentified (insufficient confidence or unknown pose)**

The system is designed for **real-time inference**, prioritizing:

- stability
- predictability
- low latency
- interpretability

---

2. High-Level System Architecture

Data Flow Overview

Camera Input
↓
MediaPipe Hand Detection
↓
Landmark Normalization & Feature Extraction
↓
Two-Hand Feature Assembly

↓
Offline-Trained Classifier
↓
Gesture Intent + Confidence
↓
Runtime Stability Logic

deliberately separates **perception**, **feature engineering**, and **classification**, enabling modular development and easier debugging.

---

3. Major Components

3.1 MediaPipe (Hand Landmark Extraction)

Role

- Detects hands in camera frames
- Outputs 21 3D landmarks per hand with 129 features in total
- Provides handedness classification (Left / Right)

Why MediaPipe

- State-of-the-art hand tracking
- CPU-friendly, real-time capable
- Outputs semantically meaningful landmarks
- Eliminates need for training vision models

MediaPipe serves as the **perceptual front-end**, allowing the rest of the system to operate in a clean geometric space.

3.2 Camera Input (Webcam)

Role

- Provides live RGB frames
- Acts as the real-time input stream

Design Notes

- No dependency on depth cameras

- Compatible with any consumer webcam
- Keeps system accessible and portable
- Bad lighting can sometimes cause mediapipe model to mistake faces for hand landmarks

---

3.3 Feature Extraction Layer

Role

- Converts raw landmarks into normalized feature vectors
- Removes variance from:
    - hand position
    - scale
    - camera framing
- Produces deterministic, fixed-length vectors

Key Techniques

- Wrist anchoring (translation invariance)
- Scale normalization (bone length reference)
- Consistent Left | Right hand ordering
- Optional two-hand symmetry features

This layer defines the **contract** between perception and ML.

3.4 Machine Learning (Scikit-Learn)

Role

- Trains a lightweight gesture classifier
- Maps feature vectors → gesture labels + confidence

Model Choice

- RBF-kernel Support Vector Machine (SVM)

Why Scikit-Learn

- Strong performance on small, structured datasets
- Minimal hyperparameter tuning
- Deterministic, reproducible results
- Easy export and deployment

The ML component focuses strictly on **classification**, not representation learning.

3.5 Jupyter Notebooks (Model Development)

Role

- Offline training and evaluation
- Data validation and visualization
- Hyperparameter tuning
- Model export

Responsibilities

- Load captured landmark features
- Train and validate models
- Analyze confusion matrices and confidence distributions
- Simulate runtime thresholds
- Export deployment-ready artifacts

Notebooks are explicitly **not used in production**.

---

4. Design Tradeoffs & Decisions

4.1 Jupyter Notebooks vs Python Training Scripts

Chosen: Jupyter Notebooks

Rationale

- Enables rapid iteration and visualization
- Makes validation and diagnostics explicit
- Easier to reason about model behavior
- Encourages exploratory data analysis

Tradeoff

- Not suitable for production
- Requires discipline in version control

Mitigation

- Training is offline

- Final output is a serialized model
- Runtime uses pure Python modules

4.2 MediaPipe vs Custom Vision Models

Chosen: MediaPipe

Rationale

- Avoids training complex vision models
- Outputs structured semantic data
- Works reliably across environments
- Significantly reduces dataset requirements

Tradeoff

- Limited to poses MediaPipe can detect
- Less flexibility than custom vision pipelines

Conclusion
MediaPipe handles perception; the project focuses on intent.

4.3 Classical ML (Scikit-Learn) vs Deep Learning

Chosen: Classical ML (SVM - RBF)

Rationale

- Input features are already high-level
- Dataset size is limited - neural net would be overkill
- Gesture classes are well-defined
- Requires stability and interpretability
-

Rejected (for now): PyTorch / TensorFlow

Why

- Adds unnecessary complexity
- Risk of overfitting
- Longer iteration cycles
- No clear accuracy gain for this task

Future Path
Deep learning may be introduced if:

- gestures become temporal or sequential
- dataset scales significantly
- representation learning becomes necessary

---

5. Non-Goals

This system explicitly does **not** attempt to:

- recognize sign language grammar
- perform video-level action recognition
- operate on raw image data
- support hundreds of gestures

These constraints keep the system focused, stable, and maintainable.

---

6. Summary

This project is intentionally designed as a:

> **Pose-based, geometry-driven gesture classification system**

Key principles:

- leverage strong pretrained perception
- engineer invariant features
- use simple, robust models, don't overengineer

The result is a system that is:

- efficient
- interpretable
- deployable
- easy to extend