Manual del Desarrollador (README)- Proyecto LSM

Versión: 1.0

Fecha: 11/04/2025

Contenido

1. Introducción

1.1 Propósito del Documento

Este manual proporciona una guía técnica completa para desarrolladores que trabajen en el Proyecto LSM (Lenguaje de Señas), incluyendo estructura del proyecto, funcionalidades clave, requisitos y procesos de desarrollo.

1.2 Alcance

Cubre los módulos de captura, reconocimiento, interfaz gráfica y utilidades, con instrucciones para ejecución, extensión y mantenimiento.

2. Estructura del Proyecto

2.1 Diagrama de Directorios

```
PROYECTO_LSM/
├── capture/              # Módulo de captura de señas
│   ├── capture.py        # Script principal
├── datasets/             # Almacenamiento de imágenes/JSON
├── reconocimiento/       # Módulo de ML
│   └── reconocimiento.py  # Clasificador SVM
├── app.py                # Interfaz gráfica (Tkinter)
└── utilidades.py         # Funciones matemáticas auxiliares
```

2.2 Descripción de Componentes

Archivo	Responsabilidad
capture.py	Captura imágenes de manos y genera datasets con metadatos (keypoints, ángulos, distancias)
reconocimiento.py	Entrena modelo SVM y clasifica señas en tiempo real
app.py	Interfaz gráfica para usuarios finales
utilidades.py	Cálculos de normalización, ángulos y distancias

3. Configuración del Entorno

3.1 Requisitos
Python 3.11+

Bibliotecas necesarias:

```
pip install opencv-python mediapipe scikit-learn numpy tkinter
```

3.2 Instalación
Clonar repositorio:

```
git clone [URL del repositorio]
```

Instalar dependencias:

```
pip install -r requirements.txt
```

4. Módulo de Captura (capture.py)

4.1 Flujo de Trabajo
1. Entrada de datos: Solicita nombre y categoría de la seña.
2. Detección de manos: Usa MediaPipe para tracking en tiempo real.
3. Procesamiento: Normaliza keypoints, calcula ángulos y distancias.
4. Almacenamiento en datasets/[nombre_seña]/: imagen (.jpg) y metadatos (.json).

4.2 Uso

```
python capture.py
```

Teclas:
s: Capturar muestra actual.
q: Salir.

5. Módulo de Reconocimiento (reconocimiento.py)

5.1 Pipeline de ML
1. Cargar datasets → 2. Extraer características → 3. Entrenar SVM → 4. Clasificar en tiempo real

5.2 Características Utilizadas

Tipo	Cantidad	Descripción

Keypoints	63	Coordenadas x, y, z de 21 puntos normalizados
Ángulos	10	Entre falanges de cada dedo
Distancias	7	Entre puntos clave (ej. muñeca-pulgar)

5.3 Ejecución

```
python reconocimiento/reconocimiento.py
```

6. Interfaz Gráfica (app.py)

6.1 Funcionalidades

Botón "Reconocer señas": Inicia el módulo de clasificación.
Diseño responsive con Tkinter.

6.2 Captura de Pantalla

[Incluir imagen de la interfaz aquí]

7. Extensión del Proyecto

7.1 Añadir Nuevas Señas

1. Ejecutar capture.py.
2. Proporcionar al menos 20 muestras por seña.

7.2 Mejorar el Modelo

```
Ajustar hiperparámetros del SVM:

SVC(kernel="linear", C=1.0)
```

8. Referencias Técnicas

MediaPipe Hands: Modelo de detección de manos con landmarks 3D.
SVM: Clasificador lineal con kernel para características no lineales.

Anexos

A.1 Ejemplo de JSON Generado

```
{
  "keypoints": [{"x": 0.1, "y": 0.2, "z": 0.3}, ...],
  "angles": [45.2, 30.1, ...],
  "distances": [0.15, 0.22, ...]
}
```

A.2 Estructura de Datasets

```
datasets/
└──── Hola/
    ├──── imagen_1.jpg
    ├──── imagen_1.json
    └──── ...
```