# Real-Time Depth Estimation API

## Overview

This project implements a real-time depth estimation API using FastAPI. The API takes video footage from a camera as input and converts it to depth maps in real-time. The system leverages a pre-trained deep learning model for depth estimation and streams the output with minimal latency using WebSockets.

## Approach

1. Model Selection: The project uses the MiDaS model for depth estimation, which is a state-of-the-art open-source 3D depth estimation model. It is loaded using PyTorch and adapted for real-time processing on CPU.  
  
2. WebSocket Streaming: Video frames are captured from the camera, sent to the server over a WebSocket connection, processed to generate depth maps, and then sent back to the client.  
  
3. Client-Side Implementation: The client-side application captures video using the getUserMedia API and displays the depth map using HTML <video> and <img> elements.  
  
4. FastAPI Server: The server is built with FastAPI, which handles WebSocket connections and processes video frames for depth estimation.  
  
5. Real-Time Processing: The system is designed to handle real-time video streams, processing each frame and sending the corresponding depth map back with minimal latency.

## Prerequisites

- Python 3.6 or higher  
- FFmpeg  
- Webcam or other camera device for capturing video

## Installation

### Python Environment

Set up a virtual environment:

python -m venv venv  
source venv/bin/activate # On Windows use `venv\Scripts\activate`

Install required packages:

pip install fastapi uvicorn opencv-python opencv-python-headless numpy torch torchvision

## File Structure

- `app.py`: Contains the FastAPI server implementation, including the WebSocket endpoint for processing video frames and generating depth maps.  
  
- `static/index.html`: A client-side HTML file that captures video from the user's camera and sends frames to the server via WebSockets. It also displays the received depth maps in real-time.  
  
- `offline\_depth\_estimation.ipynb`: A Jupyter Notebook for offline testing of depth estimation on a pre-recorded video file. This notebook can be used to process videos and generate depth map videos without real-time streaming.

## Running the Application

1. Start the FastAPI Server:

python app.py

2. Access the Client:  
Open your web browser and navigate to `http://localhost:8000/static/index.html` to start capturing video and viewing depth maps in real-time.

## Jupyter Notebook for Offline Testing

- \*\*Notebook Name\*\*: `offline\_depth\_estimation.ipynb`  
  
- \*\*Functionality\*\*: This notebook allows for testing the depth estimation model on a pre-recorded video file. The output is saved as a video with depth maps.  
  
- \*\*Steps to Run\*\*:  
 1. Open the Jupyter Notebook.  
 2. Ensure you have the video file path specified correctly.  
 3. Run the cells to generate the depth map video.

## Future Enhancements

1. GPU Support: Enhance the server to utilize GPU processing for faster depth estimation and improved performance.  
  
2. Enhanced Models: Experiment with other state-of-the-art depth estimation models for potentially better accuracy.  
  
3. Multilingual Support: Extend the API to support video streams in multiple formats and handle diverse input conditions.  
  
4. User Interface Improvements: Develop a more interactive user interface for better visualization and interaction with the depth maps.  
  
5. Scalability: Optimize the backend architecture to handle multiple simultaneous video streams efficiently.  
  
6. Batch Processing: Implement support for processing multiple videos at once for batch depth map generation.  
  
7. Integration with Flask: Provide an alternative implementation using Flask to accommodate different developer preferences and requirements.

## Third-Party Libraries

- \*\*FastAPI\*\*: For building the API server.  
- \*\*Uvicorn\*\*: ASGI server for running FastAPI applications.  
- \*\*OpenCV\*\*: For capturing and processing video frames.  
- \*\*NumPy\*\*: For numerical operations.  
- \*\*Torch\*\*: Machine learning library for loading and using the depth estimation model.  
- \*\*TorchVision\*\*: Provides utilities for image transformations required by the MiDaS model.