Tutorial Report: 3D Face Reconstruction from 2D Images using Depth Mapping

1. Introduction

This project focuses on reconstructing a **3D model of a human face** from a single 2D image using **depth estimation** and **point cloud visualization**. It leverages modern computer vision techniques and provides an interactive UI through **Streamlit**, allowing users to explore the 3D face structure in real time.

---

2. System Overview

The system performs the following key steps:

1. **Image Upload** – Accepts a frontal 2D image of a human face.
2. **Depth Estimation** – Uses the MiDaS model to predict relative depth for each pixel.
3. **3D Model Generation** – Converts the depth map into a 3D point cloud.
4. **Interactive Visualization** – Displays the 3D model using Plotly within the web app.

---

3. Tools & Technologies

Tool/Library	Purpose
Python	Core programming language
Streamlit	Web app framework
OpenCV	Image handling
PyTorch	Running MiDaS depth model
NumPy	Matrix and array operations
Plotly	3D interactive visualization

---

4. System Architecture

```
User → Streamlit UI → Image Upload
                ↓
       MiDaS Depth Estimation (PyTorch)
                ↓
      Depth Map → NumPy Point Cloud
                ↓
       Plotly 3D Scatter Plot Viewer
```

5. Project Structure

```
3D_Modeling_Project/
├── app.py                  # Streamlit app logic
├── depth_estimation.py    # MiDaS model loading and prediction
├── generate_3d_model.py   # Converts depth map to 3D point cloud
├── assets/                # Input/output image and depth assets
├── requirements.txt       # pip dependencies
```

6. Functional Modules

6.1 Image Upload (Streamlit UI)

```python
uploaded_file = st.file_uploader("Upload an image", type=["jpg", "p
if uploaded_file:
    image = Image.open(uploaded_file)
    st.image(image, caption="Uploaded Image", use_column_width=True
```

Function: Allows users to select a facial image. Image is displayed immediately after upload.

6.2 Depth Estimation with MiDaS

```python
def estimate_depth(image):
    model_type = "DPT_Large"
    midas = torch.hub.load("intel-isl/MiDaS", model_type)
    midas.eval()

    transform = torch.hub.load("intel-isl/MiDaS", "transforms").dpt_
    input_batch = transform(image).to(device)

    with torch.no_grad():
        prediction = midas(input_batch)
        depth = prediction.squeeze().cpu().numpy()

    return cv2.normalize(depth, None, 0, 255, cv2.NORM_MINMAX).asty
```

Function: Loads and runs the MiDaS model to compute a **relative depth map**, normalized to grayscale.

6.3 Depth to 3D Point Cloud Conversion

```
def generate_point_cloud(depth_map):
    h, w = depth_map.shape
    x, y = np.meshgrid(np.arange(w), np.arange(h))
    z = depth_map / 255.0
    x = x.flatten()
    y = y.flatten()
    z = z.flatten()
    return x, y, z
```

Function: Creates 3D coordinates from 2D pixel positions and depth values.

6.4 Plotly 3D Visualization

```
def plot_3d_face(x, y, z):
    fig = go.Figure(data=[go.Scatter3d(
        x=x, y=y, z=z,
        mode='markers',
        marker=dict(
            size=1,
            color=z,
            colorscale='gray',
            opacity=0.8
        )
    )])
    fig.update_layout(height=700, scene=dict(aspectmode='data'))
    st.plotly_chart(fig)
```

Function: Displays the face model as an interactive **3D scatter plot** in the Streamlit app.

---

7. Usage Instructions

Step 1: Upload a Face Image

- Use the **Upload** button to provide a frontal face image.

- Supported formats: .jpg, .png.

Step 2: Generate Depth Map

- Click **Generate Depth Map**.
- The system will show a grayscale map, with bright regions being closer.

Step 3: View 3D Model

- Click **Generate 3D Model**.
- An interactive Plotly visualization appears for zoom, rotate, and pan operations.

---

8. Sample Output

Input Image	Depth Map	3D Reconstruction

---

9. Advantages

- Converts **any single face image** into 3D structure.
- Fully **interactive web UI**.
- Uses **open-source MiDaS** for accurate monocular depth estimation.
- Visualization is smooth and high-resolution with **Plotly 3D**.

---

10. Future Enhancements

Feature	Description
Facial landmark alignment	Align faces before depth estimation
Denoising & smoothing	Use surface reconstruction to refine noisy outputs
Mesh output	Export point cloud as OBJ or STL for 3D printing
Batch mode	Support uploading and processing multiple images
Mobile support	Package app using Streamlit Mobile or PWA

---

11. Conclusion

This project successfully demonstrates the transformation of a 2D facial image into a **3D model** using AI-powered depth estimation and visual analytics. Such systems have far-reaching applications in **biometrics, entertainment, virtual reality**, and **3D printing**.

By using open-source tools and real-time visualization, the project strikes a balance between **technical robustness** and **user accessibility**.