Dance Motion Tracking & Animation — Project Draft

Overview

This project captures human dance/movement from video, extracts body and joint data using OpenCV + MediaPipe (or similar), converts that data into a format Blender can consume, and animates a 3D character to reproduce the motion. A later phase adds AI training so the system can learn from your videos and generate clean, retargeted animations automatically.

Goals

- Capture reliable 2D/3D joint tracking from consumer video (no suit).
- Produce a repeatable pipeline that maps tracked joints to a Blender rig.
- Create an MVP that demonstrates a recorded dance → Blender animation end-to-end.
- Later: train ML models to improve pose estimation, fill gaps, and synthesize animations from new video inputs.

Deliverables (MVP + Later AI Enhancement)

MVP (v1): - Video ingest (file upload) and simple viewer. - Pose extraction script (MediaPipe/OpenCV) that outputs joint data per frame (CSV/JSON/BVH). - Blender import script (bpy) that maps pose data onto a rig and keyframes bones. - Basic UI (CLI or small desktop app) to run pipeline and preview exported animation.

Phase 2 (v2) - Quality & UX: - Smoothing, filtering, and interpolation to reduce jitter. - Support for multiple camera angles (optional) and calibration tools. - Simple retargeting options for different rig proportions.

Phase 3 (AI enhancement): - Train a model to predict joint rotations from raw pose sequences (or from video frames directly). - Model-based denoising and temporal prediction (fill dropped frames, upsample FPS). - User-facing mode that takes a new video and outputs a cleaned, retargeted animation in Blender automatically.

High-level Pipeline (Steps)

1. **Pre-production (Recording setup)**
2. Choose camera (see recommendations below).
3. Mark a clean, uncluttered background and ensure even lighting.

4. Optional: place simple markers or use a calibration pose at the start for scale.

5. **Capture**

6. Record single or multiple takes.

7. Save as MP4 (H.264) or other lossless-ish codecs for best results.

8. **Ingest & Preprocess**

9. Use `ffmpeg` / OpenCV to transcode and, if needed, downsample.

10. Normalize frames, correct lens distortion if necessary.

11. **Pose Estimation**

12. Run MediaPipe (or chosen library) per frame to get joint locations and confidence scores.

13. Store joint 2D coordinates, visibility/confidence, timestamp/frame index.

14. **Post-processing / Filtering**

15. Smooth trajectories (e.g., low-pass filters, Savitzky–Golay, or Kalman filter).
16. Interpolate missing joints and correct implausible spikes.

17. Optionally reconstruct a 3D skeleton using single-view depth heuristics or a simple triangulation (if multiple cameras available).

18. **Retargeting & Mapping**

19. Convert 2D joint positions (or 3D if available) to rig bone rotations required by Blender.

20. Use an intermediate format (BVH/FBX/JSON) and a Blender `bpy` import script to apply and keyframe.

21. **Export & Preview**

22. Generate a Blender file or motion file and preview the animation.

23. Iterate on smoothing/retargeting parameters if needed.

24. **(Later) AI Training**

25. Prepare dataset: pose sequences (and optionally video frames) + ground-truth rotations/mocap if available.
26. Train temporal models to predict rotations, denoise, or upsample sequences.
27. Integrate model into pipeline to produce final animations.

Recommended Libraries & Tools

Core (MVP): - Python 3.9+ - OpenCV — video IO, pre/post-processing. - MediaPipe (Pose) — fast, practical body joint detection. - NumPy / Pandas — data handling. - ffmpeg (CLI) — transcoding and format handling. - Blender (with `bpy`) — import data and animate rigs. - matplotlib — quick plotting/visual checks.

Smoothing / Math: - SciPy — filtering and interpolation routines. - pykalman or filterpy — Kalman filters for smoothing (optional).

Machine Learning (Phase 3): - PyTorch or TensorFlow — model development and training. - tqdm — progress bars for training loops. - scikit-learn — utility functions, metrics. - Weights & Biases or TensorBoard — experiment tracking.

Optional / Advanced: - OpenPose (if you need multi-person/more accurate keypoints) — heavier but robust. - DeepLabCut / Detectron2 — if you need fine-tuned keypoint models for dance-specific cases. - BVH / FBX exporters (for industry-standard motion files).

Video Capture Recommendations (what to shoot)

Resolution: - **Start with 1080p (1920×1080).** It's the best tradeoff: clear joint detail without excessive storage and compute cost. - 4K gives more detail but increases processing time and storage — useful later for very fine-grained foot/hand work.

Framerate: - **60 FPS recommended** for dance and fast motion to reduce motion blur and make tracking smoother. - 30 FPS is acceptable for slow/moderate movement and faster to process, but 60 fps is a safer default. - Only go above 60 FPS (120–240 FPS) if you have ultra-fast moves you need to capture (and you have the storage and camera to do it).

Camera & Setup: - Use a camera with a decent global/shutter and avoid rolling-shutter artifacts if possible. - Stable mount on tripod, consistent lighting, and a single, uncluttered background. - If possible, shoot a T-pose or calibration pose at the start for scale. - Multiple angles (2+ cameras) help 3D reconstruction but increase complexity (sync + calibration required).

Data Formats & Example Schemas

Per-frame CSV/JSON schema (recommended):

```
{
  "frame": 1024,
  "timestamp_ms": 34133,
  "joints": {
    "nose": {"x": 0.512, "y": 0.232, "z": -0.03, "confidence": 0.98},
    "left_shoulder": {"x": 0.45, "y": 0.33, "confidence": 0.96},
```

```
    "left_elbow": {"x": 0.41, "y": 0.52, "confidence": 0.95}
  }
}
```

BVH — industry-standard for skeletal motion. If you can convert to BVH with correct bone hierarchy and rotation order it will be easiest to import to Blender.

Blender Integration Notes

- Use Blender's Python API (`bpy`) to create or match an armature and keyframe bone rotations per frame.
- Important: bone rotation order and coordinate system conversions are a frequent source of errors — test with a simple 2-joint example first.
- Implement retargeting to scale joint positions to different rig proportions (you may use constraints or intermediate solving).
- Add smoothing in Blender (graph editor modifiers like "Smooth" or apply interpolation curves) for polished motion.

MVP Feature List (concrete)

1. **Video Uploader / Loader** (supports MP4, MOV)
2. **Run Pose Extraction** (save JSON/CSV per take)
3. **Pose Viewer** (overlay joints on video for quick QA)
4. **Filtering Module** (Kalman or Savitzky–Golay smoothing)
5. **Export to BVH/JSON** (intermediate format)
6. **Blender Import Script** (apply keyframes and preview)
7. **Basic CLI or tiny UI** to run pipeline steps in sequence

---

Later AI Enhancements (how and what to train)

Problems AI will solve: - Denoise raw pose estimates and remove temporal jitter. - Predict missing joints or entire poses when occluded. - Map sequences of 2D poses (or frames) to 3D joint rotations suitable for animation rigs. - Generate stylistic variations or retarget to different rig sizes automatically.

Approaches & model types: - **Sequence models**: LSTM / GRU, Temporal Convolutional Networks (TCN), or Transformers for temporal mapping from pose sequences → joint rotations. - **Encoder–Decoder**: CNN (frame-level) encoder + Transformer/LSTM temporal decoder to produce pose sequences or rotations. - **Graph Neural Networks (GNNs)**: Model skeletal structure explicitly (joints as nodes, bones as edges) — good for respecting kinematic constraints. - **Diffusion / Generative models**: For creative motion synthesis and sampling new dance variations.

Losses / Metrics: - MPJPE (Mean Per Joint Position Error) — standard for 3D joint accuracy. - Angle-based losses for bone rotation accuracy (to reduce twisting). - Smoothness / temporal-consistency losses. - Perceptual evaluation (user study or automated heuristics for naturalness).

Data for training: - Start with generated data: use existing mocap datasets (e.g., CMU MoCap) and reproject to 2D, or generate synthetic renders in Blender to create paired `(video/frame sequence) -> (joint rotations)` training pairs. - Augment: add noise, occlusions, lighting changes, and cropping. - If possible, collect labeled personal dataset (your recorded videos + curated retargeted Blender outputs) to fine-tune models.

Training infrastructure: - GPU (NVIDIA recommended). For initial experiments a single 8–12GB GPU (RTX 2070/3060) is workable; for larger models use 16GB+ GPUs or cloud GPUs. - Use mixed precision training (AMP) and batch-size tune.

Inference integration: - Package model as a TorchScript/ONNX model for speed, then integrate into the Python pipeline to post-process pose data and output final rotations for Blender.

Practical Tips & Gotchas

- **Coordinate systems**: MediaPipe uses a normalized image coordinate system. Blender uses a 3D coordinate system with different axes — always test conversions on a simple rig.
- **Bone orientations & rest pose**: Ensure your Blender rig has a consistent rest pose (T-pose or A-pose). Retargeting becomes much easier with a known rest pose.
- **Occlusions**: Hands and feet are frequently occluded in dance — plan to implement robust interpolation and confidence-thresholding.
- **Frame sync for multi-camera**: If you add more than one camera later, ensure tight frame synchronization (audio clap or hardware sync) and camera calibration.
- **Data hygiene**: Keep raw video, processed frames, and final keyframes organized and versioned.

Suggested Development Roadmap (4–8 weeks MVP)

- **Week 1**: Recording tests & capture guidelines. Implement video ingest and run MediaPipe on test clips. Save pose outputs.
- **Week 2**: Build filtering & pose QA viewer. Iterate on smoothing and interpolation.
- **Week 3**: Implement Blender import script and test mapping for a simple rig. Debug coordinate problems.
- **Week 4**: Polish MVP workflow (simple CLI/UI) and produce 2–3 demo animations.

Phase 2 (4–8 weeks) - Add multi-take aggregation, retargeting options, better UI, and optional multi-camera support.

Phase 3 (AI, 8+ weeks) - Collect/prepare training data, prototype models (LSTM/Transformer/GNN), evaluate, iterate and integrate.

Minimal Example: Quick sanity checks to implement

- Visualize MediaPipe keypoints over the original video (overlay) — first check.
- Plot X/Y trajectories of key joints across frames — detect jumps/outliers.
- Export 10–20 frames to BVH and confirm bone rotations map logically in Blender.

Final Recommendations

- **Start simple**: single camera, 1080p@60fps, MediaPipe → CSV → Blender `bpy` script.
- **Iterate**: once pipeline is stable, add smoothing & retargeting, then consider ML for improving quality.
- **Use synthetic data** from Blender mocap to bootstrap AI training before collecting lots of real labelled data.
- **Keep experiments reproducible**: log parameters and keep small datasets for quick iteration.

If you want, I can now: - Export this as a Canva-ready layout (I can build the text & suggested slide structure so you can paste into Canva), or - Create a downloadable PDF or Word doc summarizing the plan.

Tell me which format you prefer and I'll prepare it next.