TL;DR ViPE is a practical, open-source spatial AI engine that annotates raw videos with camera intrinsics, camera trajectories, and dense, near metric depth maps. It supports pinhole, wide angle, and 360° panoramas; runs via a simple CLI; and includes a sizeable public dataset for research and benchmarking.

What is ViPE?

ViPE (Video Pose Engine for 3D Geometric Perception) is a video processing pipeline that turns in the wild videos into structured 3D signals:

- Camera intrinsics and motion (per frame calibrated poses)
- Dense, near metric depth maps over time
- Robust handling of diverse content: selfie videos, cinematic footage, dashcams
- Support for multiple camera models: pinhole, wide angle, 360° panoramas

This helps bridge a key bottleneck for spatial AI systems: getting consistent, precise 3D supervision from raw, unconstrained videos.

References:

- GitHub: https://github.com/nv-tlabs/vipe
- Whitepaper: https://research.nvidia.com/labs/toronto-ai/vipe/assets/paper.pdf
- Project page: https://research.nvidia.com/labs/toronto-ai/vipe

---

Why it matters for production workflows

- Pre viz & blocking: Recover camera trajectories and depth layers to plan moves, composites, and CG inserts.
- VFX and match move: Export to COLMAP compatible formats for downstream 3D tools and pipelines.
- AR occlusion & relighting: Use dense depth for object placement and light estimation.
- Dataset bootstrapping: Automatically annotate large video corpora to train or fine tune 3D/SLAM/NeRF style models.

Note: Output quality varies by footage quality (motion blur, rolling shutter, low light) and chosen pipeline options. Always validate with your target tasks.

Quick Start (CLI)

After installation, the CLI processes mp4 videos end to end:

```
# 1) Create environment and install
conda env create -f envs/base.yml
conda activate vipe
pip install -r envs/requirements.txt
pip install --no-build-isolation -e .

# 2) Run inference on a single video
vipe infer YOUR_VIDEO.mp4 \
  --output vipe_results \
  --visualize \
  --pipeline default

# Option: memory friendlier depth path (less temporal smoothing, more 3D consistency)
vipe infer YOUR_VIDEO.mp4 --pipeline no_vda

# 3) Inspect results (viser UI)
vipe visualize vipe_results/
```

For multi video batches and fine grained Hydra config control, use the provided `run.py` workflow:

```
# Full pipeline on a directory of videos
python run.py pipeline=default \
  streams=raw_mp4_stream \
  streams.base_path=YOUR_VIDEO_OR_DIR

# Pose only (skip depth alignment)
python run.py pipeline=default \
  streams=raw_mp4_stream \
  streams.base_path=YOUR_VIDEO_OR_DIR \
  pipeline.post.depth_align_model=null
```

Export to COLMAP

ViPE can export results to a familiar COLMAP layout, including an option to unproject dense depth maps to a point cloud:

```
python scripts/vipe_to_colmap.py vipe_results/ --sequence demo

# Lightweight 3D consistent point cloud (requires SLAM map saved at inference)
python scripts/vipe_to_colmap.py vipe_results/ \
  --sequence demo \
  --use_slam_map
```

This makes it straightforward to mix ViPE outputs with existing photogrammetry/SLAM tooling.

Released datasets (approximate scale)

Alongside the code, the authors release several annotated sets with camera poses and dense depth maps:

- Dynpose 100K++: ~99.5K videos / ~15.8M frames (CC BY NC 4.0)
- Wild SDG 1M: ~966K videos / ~78.2M frames (CC BY NC 4.0)
- Web360: ~2.1K videos / ~212K frames (CC BY 4.0)

Download utility (Hugging Face mirrors; depth components are large):

```
python scripts/download_dataset.py \
  --prefix dpsp \
  --output_base ./vipe_datasets \
  --rgb --depth

# Tip: for Dynpose 100K++ RGB frames, the tool can fetch from YouTube
pip install yt_dlp ffmpeg-python
```

Licensing varies by subset; check dataset pages and third party licenses before commercial use.

Practical tips

- Footage quality: Stable exposure and less motion blur improve pose/depth stability.
- 360° and wide angle: Dedicated camera models are supported; follow repo instructions as they are released.
- Resources: Depth estimation models can be memory intensive; try `--pipeline no_vda` if you hit GPU pressure.

- Validation: Use `vipe visualize` to quickly spot depth tearing, drift, or intrinsics mismatches.

References

- ViPE GitHub: https://github.com/nv-tlabs/vipe
- Technical whitepaper (NVIDIA Research): https://research.nvidia.com/labs/toronto-ai/vipe/assets/paper.pdf
- Acknowledged components: DROID-SLAM, Depth Anything V2, Metric3D, PriorDA, UniDepth, Video Depth Anything, GeoCalib, Segment and Track Anything (see repo for licenses)

Notes on claims and numbers Counts (videos/frames) and qualitative robustness claims reflect the public README/whitepaper at publish time. Always consult the upstream repo for updates and evaluate against your own data.