Optical Flow Algorithm Picker

Map sparsity, latency, and accuracy requirements to the right primitive — and the right per-frame compute budget.

Decision tree — pick the algorithm

1. Do you need flow at every pixel?

- ☐ No (tracking, AR anchors, odometry) → Lucas-Kanade.
- ☐ Yes (stabilization, interpolation, VSR) → continue.

2. Is the pipeline real-time?

- ☐ Yes, CPU only → DIS (Kroeger 2016, 300–600 Hz).
- ☐ Yes, GPU available → SEA-RAFT (20+ FPS at 1080p).
- ☐ Yes, NVIDIA-only deployment → NVIDIA Optical Flow SDK.
- ☐ No, offline → continue.

3. Accuracy ceiling required?

- ☐ Standard → RAFT (Sintel EPE 2.855, KITTI F1 5.10%).
- ☐ State-of-the-art → FlowFormer / MegaFlow / DPFlow (2026).

4. Licence requirements?

- ☐ Commercial OK → all listed are BSD/Apache/MIT.
- ☐ NVIDIA hardware lock → Optical Flow SDK is GPU-locked.

Per-frame compute budget

Algorithm	Type	Hardware	Speed
Lucas-Kanade	Sparse	CPU	~200 FPS @ 720p
Farnebäck	Dense	CPU	~30 FPS @ 720p
DIS	Dense	CPU	300–600 Hz @ SD
RAFT	Dense	GPU	9 FPS @ 1088×436
SEA-RAFT	Dense	GPU	20+ FPS @ 1080p
FlowFormer	Dense	GPU	~4 FPS @ 1080p

Three-failure-mode audit

- ☐ Sparsity fits the feature: dense for stabilization / interpolation / VSR.
- ☐ Brightness assumption holds: photometric normalisation under flicker / low light.
- ☐ Latency budget respected: 33 ms/frame at 30 FPS — RAFT will not fit.

Library and licence quick reference

- ☐ Lucas-Kanade / Farnebäck: cv2 (Apache 2.0).
- ☐ DIS: cv2-contrib DISOpticalFlow (Apache 2.0).
- ☐ RAFT: torchvision raft_large (BSD 3-Clause).
- ☐ SEA-RAFT: GitHub (BSD 3-Clause).
- ☐ FlowFormer: GitHub (MIT).