Ideal Optical Flow Images

Reproducible Farnebäck Settings and Visual Comparison for Motion Flow Generation

September 14, 2025

1 Ideal Optical Flow Images

To ensure that the optical flow images generated by the Farnebäck method from the `cv2` library are optimized for use in machine learning models, multiple versions of the optical flow were generated and compared.

Template call. The template for the Farnebäck function call is:

```
flow = cv2.calcOpticalFlowFarneback(prev_gray, next_gray, None,
                                    pyr_scale, levels, winsize,
                                    iterations, poly_n,
                                    poly_sigma, flags)
```

where parameters are:

- `prev_gray`, `next_gray`: consecutive grayscale frames (previous and next).
- `pyr_scale`: pyramid scale factor between layers; higher values emphasize smaller motions.
- `levels`: number of pyramid layers (more layers capture larger motions across scales).
- `winsize`: neighborhood window size; smaller values capture fine detail, larger values smooth.
- `iterations`: iterations per pyramid level; more can refine small movements but add compute time.
- `poly_n`: pixel neighborhood size for polynomial expansion; smaller is more sensitive, larger is smoother.
- `poly_sigma`: Gaussian smoothing std. of the neighborhood; higher means more smoothing.
- `flags`: optional flags (e.g., border handling); used for fine-tuning.

Five sensitivity variants were evaluated for the TF-66 dataset by adjusting `cv2.calcOpticalFlowFarneback` parameters to control how much motion appears in the resulting optical-flow image. The sensitivity levels and parameters are listed in Table 1.

Table 1: Sensitivity settings and corresponding `cv2.calcOpticalFlowFarneback` parameters.

Note: "G" = `cv2.OPTFLOW_FARNEBACK_GAUSSIAN`

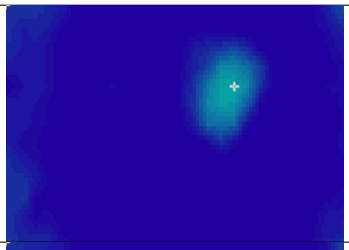
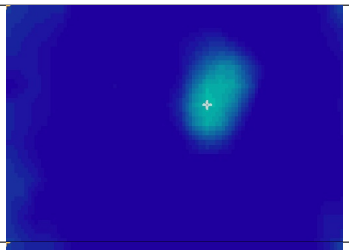
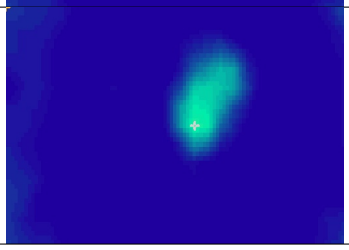
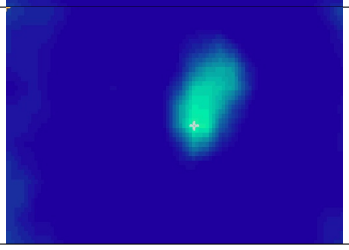
Sensitivity	pyr_scale	levels	winsize	iterations	poly_n	poly_sigma	flags
High	0.8	5	15	5	5	1.1	0
Medium-High	0.7	4	19	4	7	1.2	0
Balanced	0.5	3	21	3	5	1.2	G
Low-Medium	0.4	3	25	3	7	1.3	G
Low	0.3	2	31	2	7	1.5	G

Visual evaluation protocol. For each sensitivity level, two optical-flow images are presented:

1. **Motion:** optical flow between frames 38 and 39 of video "01-Fall-04" (the subject is actively falling).
2. **No motion:** optical flow between frames 86 and 87 of the same video (the subject is stationary, prone).

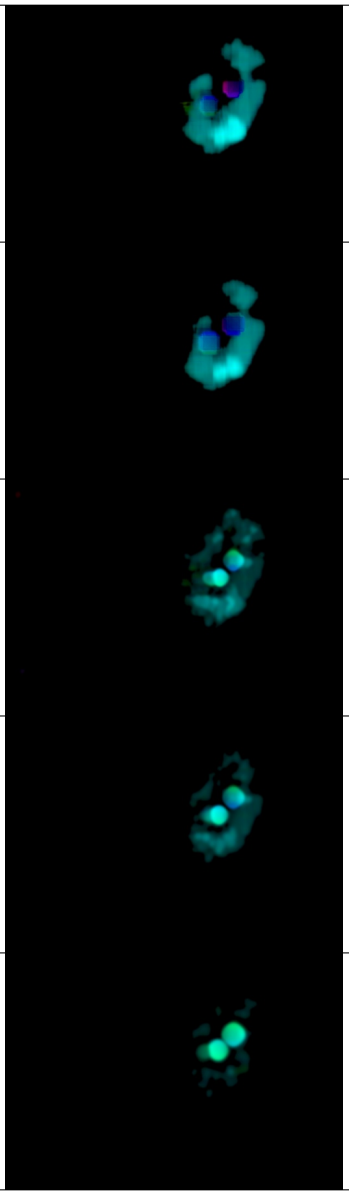
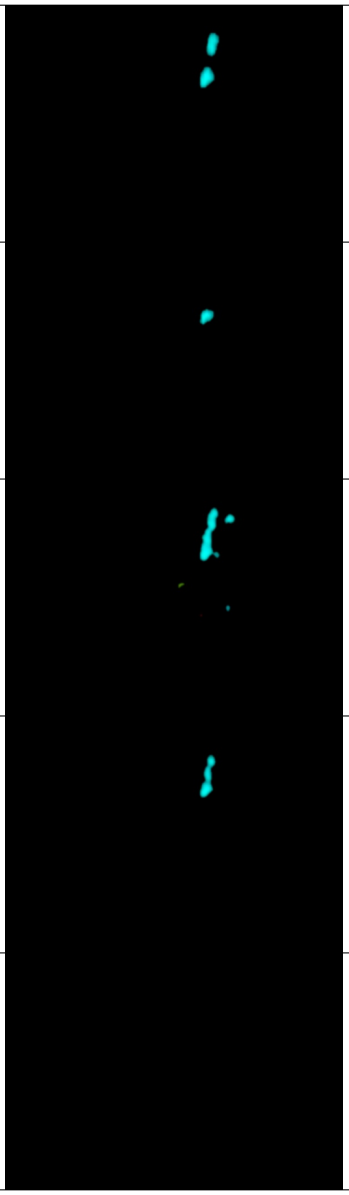
The corresponding original frames are shown in Table 2; the resulting optical-flow images for each sensitivity are shown in Table 3.

Table 2: Consecutive frames illustrating motion (top) and no motion (bottom) in "01-Fall-04" (TF-66).

Description	Frame `X`	Frame `X+1`
Consecutive frames demonstrating motion		
Consecutive frames demonstrating no motion		

Selection and rationale. In all sensitivity settings except *Low*, spurious motion was extracted when the subject was already prone (no movement). This behavior is undesirable. The *Low* setting suppressed such false motion while still producing a strong, localized flow signal for the true fall event. Consequently, the parameters for `cv2.calcOpticalFlowFarneback` were set to the *Low* sensitivity in Table 1 for dataset-wide generation.

Table 3: Optical-flow outputs at varying sensitivities for frames 38–39 (middle, motion) and 86–87 (right, no motion) of "01-Fall-04".

Sensitivity	Optical Flow 38–39	Optical Flow 86–87
High		
Medium-High		
Balanced		
Low-Medium		
Low		

2 Future Development

Planned extensions include automating per-scene sensitivity selection, adding temporal consistency checks to suppress residual noise in low-motion segments, and exporting flow fields in a standardized format (e.g., two-channel `.npy`) alongside the rendered color images to support a broader range of deep learning architectures.

Minimal OpenCV snippet (for reference)

```
 import cv2
 import numpy as np
 prev_gray = cv2.imread('frame38.jpg', cv2.IMREAD_GRAYSCALE)
next_gray = cv2.imread('frame39.jpg', cv2.IMREAD_GRAYSCALE)
flow = cv2.calcOpticalFlowFarneback(prev_gray, next_gray, None,
                                     pyr_scale=0.3, levels=2, winsize=31,
                                    iterations=2, poly_n=7, poly_sigma=1.5,
                                    flags=cv2.OPTFLOW_FARNEBACK_GAUSSIAN)
```