Optical Flow

Functions: computer vision/image processing, corner detection, Lucas-Kanade algorithm
Inputs: paired images (same scene and camera, different positions)
Metrics: per-stage and full-pipeline time

1. Overview

Optical flow is a means of detecting the relative motion of a scene and an observer. This is particularly useful for (1) detecting motion in a scene observed by a stationary platform, or (2) monitoring and controlling the motion of the observer (e.g. a drone or a car) through space, which can yield useful data on both the observer's movement relative to fixed reference points and other moving objects' motion relative to the observer's trajectory. For further reading, see https://nanonets.com/blog/optical-flow/ or https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html.

MilSpec's optical flow reference implementation is based on https://github.com/mbeyeler/opencv-python-blueprints/blob/master/chapter4/scene3D.py. First, corner detection is run on the input images to find points of interest whose motion can be easily tracked. Next, the Lucas-Kanade algorithm is used to locate the corresponding points in each image and estimate motion. This is then used to generate an output image that shows optical flow vectors at the previously identified points. The MilSpec version of this code ports the Python implementation to C++, adds performance metrics, and increases configurability with respect to varying test scenarios. The corner detection method is configurable between Harris and minimum-eigenvalue (selected in testcase-specific configuration files). A future revision will further improve the facilities for testcase definition, output image generation, and result

validation, and may incorporate additional algorithms to create a three-dimensional version of optical flow.

2. System requirements

Platform: Ubuntu 18.04 LTS with g++ 7.4.0. Code may build and run successfully on other versions/platforms, but has not been tested with them.
Storage: ~35MB for code and sample input images.
Dependencies: OpenCV 3.2.0 (may work with later versions, but not tested).

3. Build and run

To benchmark:

- Install OpenCV: `sudo apt-get update` and `sudo apt-get install libopencv-dev`
- Verify compatible OpenCV version: `pkg-config --modversion opencv`
- Download and extract the zipfile from www.adacenter.org/milspec
- From the `OpticalFlow/` directory, `make clean && make`
- Choose a testcase: `standard.in` (simple baseline testcase) or `batch.in` (processes multiple sequential images in a single directory).
- From the `OpticalFlow/` directory, `./opticalflow [testcase]`
- Results are displayed in the terminal, as below:

```
OpticalFlow Batch
Total number of image pairs: 7
Total time to find corners of features (method = HARRIS, keypoint_threshold = 0.
0004): 1.64821s
Average number of corners found per pair: 4376
Total time to apply Lucas-Kanade algorithm: 0.341164s
Total time taken: 2.63932s
```

4. Code structure

Coming soon!

5. MilSpec development notes, errata, changelog

v0.9:

- Changes from baseline optical flow code for MilSpec:
  - Translated from Python to C++
  - Created makefile
  - Added basic configuration options and file-based testcase input
  - Created basic testcases
  - Added performance metrics
  - Added configurable corner detection method

6. Acknowledgements

MilSpec is a project under development at the University of Michigan – Ann Arbor by Pete Ehrett, Bing Schaefer, Adrian Berding, John Paul Koenig, Pranav Srinivasan, Todd Austin, and Valeria Bertacco. This project is supported by the Applications Driving Architectures Center, one of six centers of JUMP, a Semiconductor Research Corporation program co-sponsored by DARPA.

The original optical flow implementation upon which the MilSpec version is based was written by Michael Beyeler,[1] and is available at https://github.com/mbeyeler/opencv-python-blueprints/blob/master/chapter4/scene3D.py. Input images are drawn from https://drivingstereo-dataset.github.io/.

[1] M. Beyeler, (2015). OpenCV with Python Blueprints: Design and develop advanced computer vision projects using OpenCV with Python. Packt Publishing Ltd., London, England, 230 pages, ISBN 978- 178528269-0.