engineering proceedings

MDPI

Type of the Paper (Proceeding Paper, Conference Report, Editorial, etc.)

SIF—A Lightweight Optical Flow Algorithm for Micro Drones †

Che Liu [1], Chen-Fu Yeh [2] and Chung-Chuan Lo [3,*]

[1] Institute of Systems Neuroscience, National Tsing Hua University, Hsinchu, Taiwan; chadliu@lolab-nthu.org
[2] Institute of Systems Neuroscience, National Tsing Hua University, Hsinchu, Taiwan; chen_fu_yeh@lolab-nthu.org
[3] Institute of Systems Neuroscience, National Tsing Hua University, Hsinchu, Taiwan; cclo@mx.nthu.edu.tw
* Correspondence: cclo@mx.nthu.edu.tw; Tel.: +886-3-5742014
† Presented at the IEEE ICEIB, New Taipei, Taiwan, April 25-27, 2025

Abstract: In this study, we propose Selective Intersection Flow (SIF), a lightweight optical flow algorithm that enhances efficiency and accuracy by filtering out non-contributive pixels. SIF, derived from the differential category of algorithms, computes optical flow by analyzing intersections of equations from selected pixels rather than solving for all pixels. It replaces costly warping with a minimal computational procedure for initial flow estimate and employs a sliding window for optimized single-core performance. SIF runs 1.7–1.8× faster and achieves 1.2–1.4× higher accuracy than the single iteration of Lucas-Kanade method, showing promise for real-time micro drone navigation.

Keywords: optical flow; algorithm; image processing

1. Introduction

Optical flow estimation is central to image processing, with modern techniques using deep learning and Convolutional Neural Networks [1] to deliver high accuracy. However, these methods demand intensive GPU resources and high power consumption to achieve real-time performance. Traditional approaches, in contrast, offer a trade-off: the differential methods provide high speed but moderate accuracy, while block-matching methods [2] are more accurate yet slower.

This study aims to develop an optical flow algorithm for obstacle avoidance in micro drone navigation. Modern methods exceed the computational capacity of our in-tended platform, and while traditional differential techniques are well-suited for real-time use, they often compromise accuracy. The differential approach assumes that the displacement of the image between two consecutive frames is small and approximately constant within a neighborhood of the point *(x, y)* under consideration. This assumption leads to the classic optical flow equation $IxVx + IyVy = -It$. This is a linear equation of two variables, *Vx* and *Vy* represent the components of the optical flow vector in the *x* and *y* directions, while *Ix*, *Iy*, and *It* are the spatial and temporal gradient at *(x, y, t)*. For each pixel, there is an infinity number of solutions for the flow (*Vx* and *Vy*). By assuming neighboring pixels share the same flow and solving the simultaneous equations within a small window, the Lucas-Kanade algorithm [3] yields an estimated flow that can be refined through iterative processing. Enhanced versions employ pyramidal structures [4] and techniques like patch-and-stride [5] to lower computational costs.

Our study aims to develop an optical flow algorithm that is more accurate and

Academic Editor: Firstname Lastname

Published: date

Citation: To be added by editorial staff during production.

faster than the one-iteration Lucas-Kanade (simpleLK) variant. Optical flow techniques can also be classified as sparse or dense. Given that the motion in drone footage is primarily due to the camera rather than moving objects, our approach is designed to produce dense optical flow.

2. Materials and Methods

2.1 Basic Concept

Our study was inspired by observing the lines formed by $IxVx + IyVy = -It$. In a 5×5 window, 25 lines emerge on the *Vx*, *Vy* plane; while most lines converge near the true optical flow, some deviate significantly (Fig. 1(a)). Removing these outliers reduces both computational load and errors (Fig. 1(b)).

When enough qualified lines remain, we use a lightweight method to select the best ones and compute their intersections. The geometric center of these intersections approximates the true optical flow. Noticing that some intersections stray from the estimated flow (Fig. 1(b)), we apply a filter to remove them before the final calculations. The SIF algorithm achieves this efficiently in eight non-iterative steps (Fig. 1(c)).

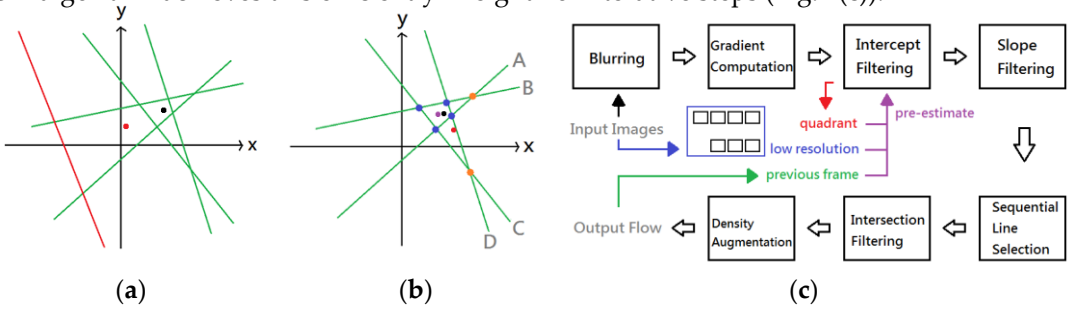

Figure 1. The basic concept of the proposed SIF algorithm. (**a**) For simplicity, only 5 lines from 5 neighboring pixels are shown. The green lines are close to the true optical flow (black dot), while the red line is an outlier and needs to be removed. The red dot indicates the estimated flow by the five lines. (**b**) After the outlier removal, the four lines form six intersections, giving rise to a new estimate (red dot). However, two intersections (orange dots) are far from the true optical flow and should be removed. The geometric center (purple dot) of the remaining four intersections represents a good estimate of the true optical flow. (**c**) The flow chart of the SIF algorithm.

2.2 SIF Step 1 - Blurring

Blurring is a pre-processing step used in many optical flow algorithms. The optical flow algorithm can better adapt to larger optical flows by incorporating pixels from a larger area. In SIF, unweighted square window blurring is applied with a window size of 5x5 pixels.

2.3 SIF Step 2 – Gradient Computation

Considering the gradient *Ix* (or *Iy*) at position (x, y) is typically calculated using values at x+1 (or y+1) and x–1 (or y–1), we also compute the gradient at *t* using values at t+1 and t–1(Eq. (2.1)), rather than the traditional t+1 and t as in most optical flow algorithms. This symmetric temporal and spatial gradient can enhance the accuracy of optical flow estimates. We can further improve the computation of gradients *Ix* and *Iy* by including the second terms of the Taylor expansion as shown below (Eq. (2.2) and Eq. (2.3)).

$$It = (I(x,y,t+1) - I(x,y,t-1))/2 \quad (2.1)$$

$$Ix = (8 \cdot I(x+1) - 8 \cdot I(x-1) - I(x+2) + I(x-2))/12 \quad (2.2)$$

$$Iy = (8 \cdot I(y+1) - 8 \cdot I(y-1) - I(y+2) + I(y-2))/12 \quad (2.3)$$

2.4 SIF Step 3 -- Intercept Filtering

The filtering process removes lines that could distort the optical flow computation by eliminating those far from the true solution in the Vx–Vy plane. In practice, since the true optical flow is unknown, we begin by pre-estimating a potential flow—if unavailable, the coordinate origin serves as a reference.

Instead of calculating the computationally expensive point-to-line distance, we use the line's intercepts with the coordinate axes as a surrogate. The absolute value of an intercept approximates the distance from the reference point, while its sign indicates the quadrants through which the line passes, an information useful in a later step.

A key parameter, InterceptFilter (CF), determines qualification: if both the absolute x- and y-intercepts are below CF, the line is kept (Fig. 2(a)); otherwise, it is discarded. When a pre-estimated optical flow is available, shifting the coordinate origin to that flow markedly improves accuracy (Fig. 2(b)).

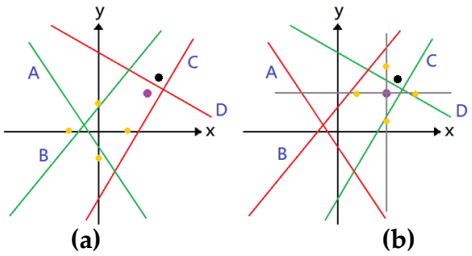

Figure 2. The concept of intercept filtering. For simplicity, only four lines are shown. The black and purple dots represent the true and pre-estimated optical flow, respectively. The yellow dots represent CF. (**a**) When using the origin as the reference point, lines A and B are qualified, while lines C and D are unqualified. (**b**) When shifting the origin to the pre-estimate, lines C and D are qualified, while lines A and B are unqualified, resulting in a more accurate optical flow estimation.

2.5 SIF Step 4 -- Slope Filtering

This step involves another key parameter: SlopeFilter (abbreviated as SF), which is used to eliminate lines with slopes that are nearly horizontal or vertical. For example, when SF = 10, lines with absolute slopes greater than 10 or less than 0.1 are deemed unqualified. The detailed rationale for slope filtering will be explained in Section 2.7 .

2.6 SIF Step 5 – Sequential Line Selection

If a sufficient number of lines pass the intercept and slope filters, we then select a subset to reduce the intersection computation load in a cost-effective manner. Two parameters guide this process: MinLine and MaxLine. If qualified lines number fewer than MinLine, the optical flow is deemed unreliable and (0, 0) is returned; if there are more, only up to MaxLine lines are chosen.

The selection prioritizes lines originating near the center pixel, based on a fixed prearranged order that eliminates extra distance calculations (Fig. 3(a) and Fig. 3(b)). To further improve quality, we ensure diversity in slopes by dividing the plane into three sections for positive slopes and three for negative slopes (Fig. 3(c)). Qualified lines are assigned to their respective slope queues and then selected in a round-robin fashion.

2.7 SIF Step 6 -- Intersection Filter

We observed that intersections between two lines with similar slopes can stray far from the main cluster (Fig. 1(b)). Consequently, we accept only intersections formed by one positive-slope and one negative-slope line. This rule, along with prior slope filtering, ensures sufficient slope difference and reduces the number of calculated intersections.

Accordingly, during line selection, we count positive- and negative-slope lines separately relative to the MinLine and MaxLine thresholds.

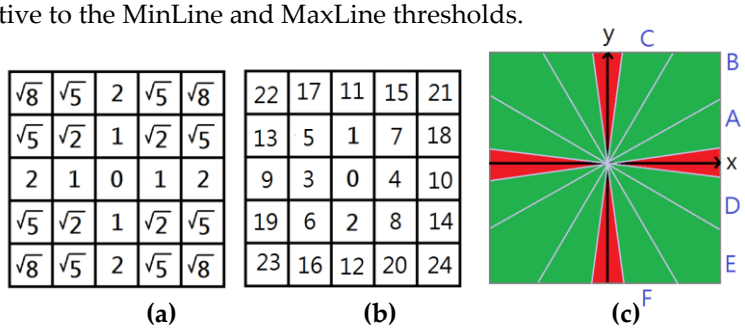

Figure 3. Sequential line selection. (**a**) Distance of each pixel from the center pixel. (**b**) The pixels are order based on their distances from the center pixel. (**c**) The slope filtering stage removes the slopes that fall in red areas. The green area with a positive slope is divided into three sections: A, B, and C. The green area with a negative slope is divided into three sections: D, E, and F.

2.8 SIF Step 7 -- Density Augmentation

After processing every pixel, a dense optical flow field is theoretically obtained. However, due to filtering and the MinLine requirement, some regions may lack computed flow, resulting in lower density than methods like simpleLK. To remedy this, post-processing increases density: for any pixel without computed flow, the average of the flows from the pixels in the surrounding region (defined by the parameter windowsize) is used as the optical flow for that pixel.

2.9 SIF Step 0 -- Guessing Pre-flow

In the Intercept Filter section, the term "pre-estimate optical flow" refers to a quick initial guess that can significantly boost accuracy if computed with minimal cost. We explored three methods for generating this pre-estimate:

1. **Temporal Smoothing:** If optical flow changes smoothly, use the flow from the previous frame.
2. **Low-Resolution Estimate:** Compute the optical flow on a downscaled image (without a pre-estimate), then upscale it for the full-resolution computation.
3. **Quadrant Penalty:** Analyze each line's slope and intercept to determine which quadrant is least likely to contain the true flow. Assign penalties based on this analysis, then adjust the qualified range of CF according to the quadrant with the lowest total penalty.

For temporally smooth flows, the first method offers the best speed and accuracy. For rapid motions—such as from a fast-moving camera—the low-resolution approach (method 2) yields better results, while the third method is useful in other scenarios.

2.10 Evaluation

The performance of our optical flow algorithm is evaluated using three metrics: endpoint error (EPE), normalized endpoint error (NEPE), and density (den). Among these, NEPE is the primary metric, which is a scalar calculated by dividing EPE with the length of ground truth.

2.11 Dataset

MidAir Dataset is used in the present study [6]. It contains 5 videos that simulate an observer (camera) flying in different environments under four weathers, with 325 frames in each video, and ground truth of optical flow between each frame.

3. Results

3.1. Windowsize of Blur & Improved Grdient

Blurring primarily improves the algorithm's ability to handle larger optical flows. We tested window sizes from 3 to 9 and optical flow lengths from 0.5 to 4.0 pixels using both simpleLK and SIF (Fig. 4(a) & 4(b)). To generate flows of specific lengths, we translated an image in three directions and used the average NEPE as the performance metric for each parameter combination. By combining blurring with enhanced gradient computation, both algorithms can reliably process flows up to 3 pixels. Because accuracy improvements plateau with window sizes over 5, we set windowsize = 5 in subsequent experiments.

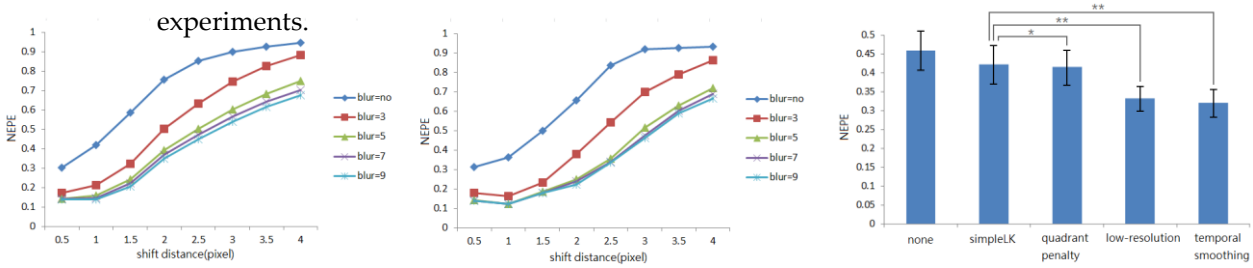

Figure 4. Experimental results of five different blurring window sizes over eight translation distances. (**a**) Results for simpleLK. (**b**) Results for SIF. (**c**) The performance of different ways of optical-flow pre-estimation.

3.2 Value of Other Parameters

Through similar experiments as above, we have found the optimal set of parameters : Minline=3, Maxline=7, CF=1.5 (with pre-estimate) or 8.5 (without pre-estimate), SF=10.

3.3 Different Guesses of Pre-flow

We evaluated various optical-flow pre-estimation methods using a video segment from the MidAir Dataset. Five methods were tested—three from section 2.9, a baseline with no pre-estimate (all zeros), and simpleLK for comparison. Each method was applied to five video segments with nine frames per segment (45 data points total). The results (Fig. 4(c)) confirmed the effectiveness of our pre-estimation strategies.

3.4 Performance on MidAir Dataset

We visualized the optical flow direction and magnitude using colors and compared the original images from the MidAir Dataset, the ground truth, and the results of the SIF algorithm both before and after density enhancement (Fig. 5).

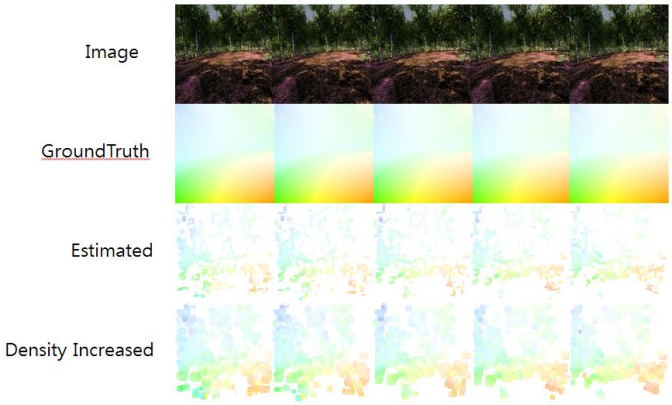

Figure 5. Visualized performance of SIF algorithm on a section of MidAir Dataset.

3.5 Comparison of Speed, Error, and Density

To compare the speed of the algorithms, we ran the C++ implementation repeatedly 10 times on an Intel i7 single-core CPU and took the average runtime (excluding image reading, error calculations, etc.) for each frame. The results confirmed that SIF is faster than simpleLK (Table 1). The SIF variation here uses *low resolution* as the pre-estimate.

Table 1. Comparison of simpleLK and SIF.

Metric	simpleLK	SIF
NEPE (ratio)	0.432 (1)	0.333 (0.771)
Time(ms) (ratio)	157.6 (1)	84.4 (0.536)
Density (ratio)	0.970 (1)	0.786 (0.810)

4. Discussion

In this study, we introduce Selective Intersection Flow (SIF), a lightweight optical flow algorithm that balances efficiency with accuracy. Unlike traditional differential methods, SIF filters out non-contributory pixels and computes flow by analyzing intersections of filtered equations rather than solving equations for every pixel, thereby reducing computational overhead. We further enhance efficiency by replacing the costly warping step in pyramidal techniques with a lightweight initial flow estimate. SIF runs 1.7–1.8× faster and achieves 1.2–1.4× higher accuracy than a single iteration of the Lu-cas-Kanade method.

The primary application for the SIF algorithm is obstacle avoidance in micro drones. Given their low flight speeds and limited camera resolution, its inability to process flows larger than 3 pixels and its somewhat lower density are acceptable drawbacks. Nonetheless, there is still potential to improve accuracy, speed, and flow density. A further limitation is the algorithm's inflexibility; its symmetric computation of *It* prevents the use of warping, which means iterative or pyramidal techniques cannot be applied to boost accuracy or handle larger flows.

In conclusion, the proposed SIF algorithm provides a lightweight method for optical flow estimation. Its integration into an optical flow-based obstacle avoidance system for micro drones highlights its potential for real-time navigation.

References

1. Ilg, Eddy & Mayer, Nikolaus & Saikia, Tonmoy & Keuper, Margret & Dosovitskiy, Alexey & Brox, Thomas. (2017). FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. 1647-1655, doi: 10.1109/CVPR.2017.179
2. A H. Gharavi and M. Mills, "Blockmatching motion estimation algorithms-new results," in IEEE Transactions on Circuits and Systems, vol. 37, no. 5, pp. 649-651, May 1990, doi: 10.1109/31.55010
3. Lucas, B. and Kanade, T. 1981. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674– 679, doi:10.5555/1623264.1623280
4. Bergen, J.R., Anandan, P., Hanna, K.J., and Hingorani, R. 1992. Hierarchical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, pp. 237–252, doi: 10.1007/3-540-55426-2_27
5. Shum, H.-Y. and Szeliski, R. 2000. Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 16(1):63–84, doi: 10.1023/A:1008195814169
6. M. Fonder and M. Van Droogenbroeck, "Mid-Air: A Multi-Modal Dataset for Extremely Low Altitude Drone Flights," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Long Beach, CA, USA, 2019, pp. 553-562, doi: 10.1109/CVPRW.2019.00081