

Algorithms Overview

Theory

The temporally filtered Lucas-Kanade algorithm is another approach to extract optic flow over the temporal domain. While the combination of the [Lucas-Kanade algorithm](#) with a Kalman Filter included temporal filtering as a post-processing step the following approach includes the temporal filter directly in the estimation step.

In the [Lucas-Kanade algorithm](#) the optic flow constraint is solved by weighted least-squares estimation over the spatial domain. Here we extend the estimation to the temporal domain leading to the following minimization problem:

$$\min \sum_{x,y,t} w(x,y,t) (I_x u + I_y v + I_t)^2 \quad (1)$$

$$\mathbf{G} \begin{pmatrix} u \\ v \end{pmatrix} = \sum_{x,y,t} w(x,y,t) \begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \sum_{x,y,t} w(x,y,t) \begin{pmatrix} I_x I_t \\ I_y I_t \end{pmatrix} = \mathbf{B} \quad (2)$$

$$w(x,y,t) = \tilde{w}(x,y) (1 - \alpha) \alpha^{-t} \quad (3)$$

The solution to this minimization problem is given by equation 2. By choosing an exponential temporal weighting function (equation 3) we avoid to fully recompute \mathbf{G} and \mathbf{B} in every step. Instead we can compute

\mathbf{G}
and
 \mathbf{B}
iteratively
(equations 4 and 5)

$$\mathbf{G}_t = \mathbf{G}_{t-1} - \alpha \mathbf{G}_{t-1} + \alpha \tilde{\mathbf{G}}_t \quad \mathbf{B}_t = \mathbf{B}_{t-1} - \alpha \mathbf{B}_{t-1} + \alpha \tilde{\mathbf{B}}_t$$
$$\tilde{\mathbf{G}}_t = \sum_{x,y} \tilde{w}(x,y) (I_x \ I_y)^T (I_x \ I_y) \quad \tilde{\mathbf{B}}_t = \sum_{x,y} \tilde{w}(x,y) (I_x \ I_y)^T I_t$$

The properties of the temporal filtering are controlled by parameter α which must lie between 0 and 1 where larger values mean less filtering over the temporal domain.

References:

Fleet, D., & Langley, K. (1995). Recursive filters for optical flow. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(1), 61-67

Algorithm

This method is available as an [implementation for microcontroller](#) as well as a generic C algorithm in the Library in

[visual_processing/optic_flow/TemporalLucasKanade.cpp](#)

[visual_processing/optic_flow/TemporalLucasKanade.hpp](#)