Optical Flow

Chao Duan, Xuan Yang

Optical Flow

1. Basic idea
2. Correlation-based optical flow
3. Differential-based and Dense optical flow

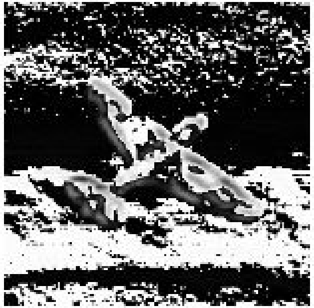
Basic idea

image1

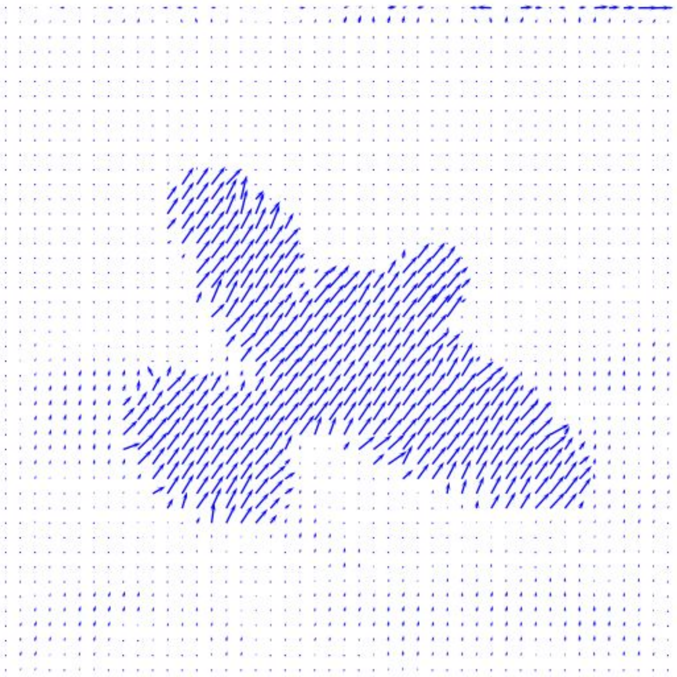
image2

(original image1and image2 are captured from [1])

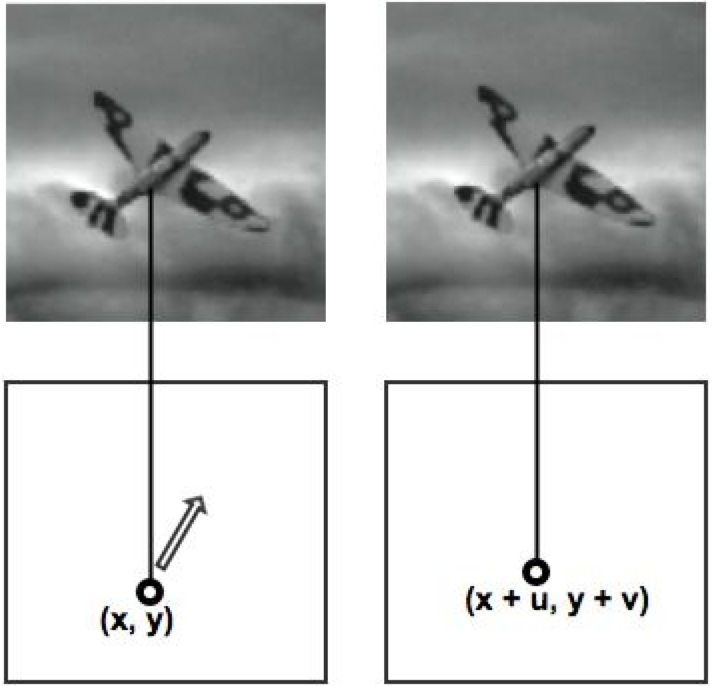
image2-1



Motion detecting result



Basic idea



Important assumption

1. Constant brightness

2. Similar velocity for a block

3. Small displacement

Correlation-based

Correlation

(x, y)

([x-d, x+d], [y-d,y+d])

$(2d+1)^2$ *possible* I_2

i

71	68	63
66	65	89
101	82	60

I_1 j

Differences measure methods:

$SAD(Sum\ of\ absolute\ differences)$

$$\sum_{(i,j)\in W} |I_1(i,j) - I_2(dx+i, dy+j)|$$

$ZSAD(Zero-mean\ Sum\ of\ absolute\ differences)$

$$\sum_{(i,j)\in W} |I_1(i,j) - \overline{I_1}(i,j) - I_2(dx+i, dy+j) + \overline{I_2}(dx+i, dy+j)|$$

$SSD(Sum\ of\ squared\ differences)$

$$\sum_{(i,j)\in W} (I_1(i,j) - I_2(dx+i, dy+j))^2$$

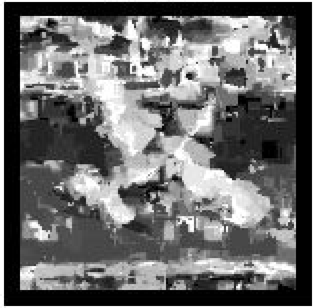
$ZSSD(Zero-mean\ Sum\ of\ squared\ differences)$

$$\sum_{(i,j)\in W} (I_1(i,j) - \overline{I_1}(i,j) - I_2(dx+i, dy+j) + \overline{I_2}(dx+i, dy+j))^2$$

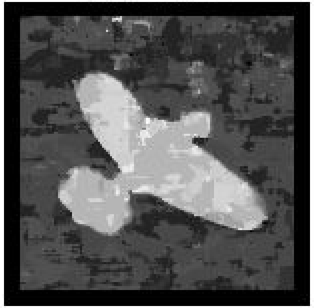
Correlation

No Gaussian filter:

SAD 5.48s

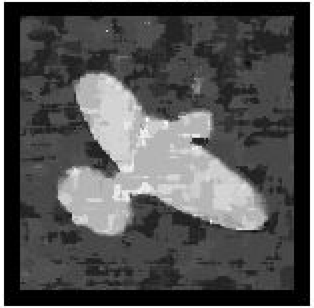


ZSAD 7.74s



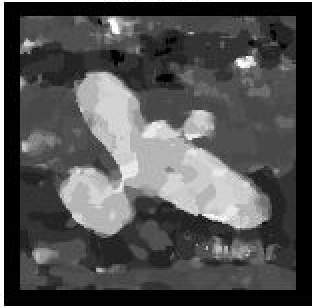
SSD 4.87s

ZSSD 8.51s



With Gaussian filter:

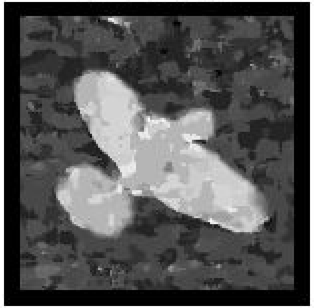
SAD 4.89s



ZSAD 7.92s

SSD 4.55s

ZSSD 8.37s



Correlation

Similarity measure methods:

$$CC(Cross\ correlation)$$

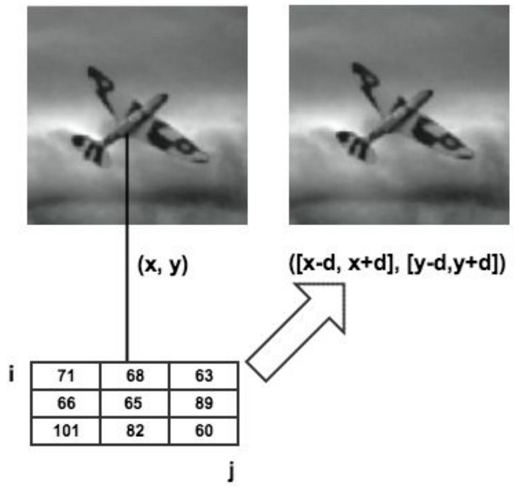
$$\sum_{(i,j)\in W} I_1(i,j) I_2(dx+i, dy+j)$$

$$NCC(Normalized\ Cross\ correlation)$$

$$\frac{\sum_{(i,j)\in W} I_1(i,j) I_2(dx+i, dy+j)}{\sqrt{\sum_{(i,j)\in W} I_1(i,j)^2 \sum_{(i,j)\in W} I_2(dx+i, dy+j)^2}}$$

$$ZNCC(Zero-mean\ Normalized\ Cross\ correlation)$$

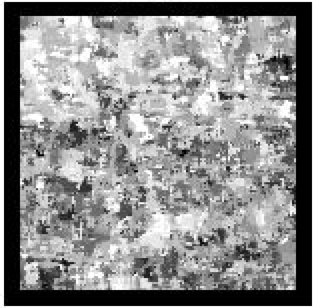
$$\frac{\sum_{(i,j)\in W}(I_1(i,j)-\overline{I_1}(i,j))(I_2(dx+i,dy+j)-\overline{I_2}(dx+i,dy+j))}{\sqrt{\sum_{(i,j)\in W}(I_1(i,j)-\overline{I_1}(i,j))^2 \sum_{(i,j)\in W}(I_2(dx+i,dy+j)-\overline{I_2}(dx+i,dy+j))^2}}$$



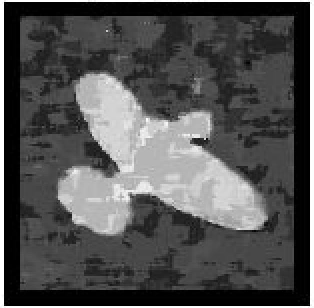
Correlation

No Gaussian filter:

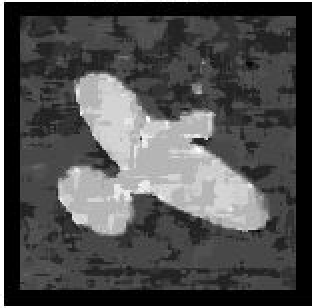
CC 6.23s



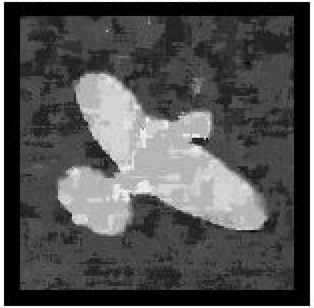
NCC 58.10s



ZNCC 19.08s

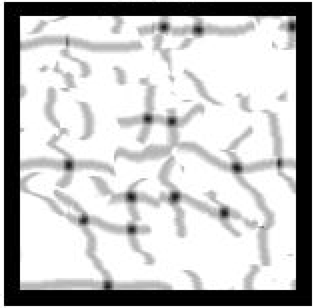


ZNCC (speedup) 10.67s

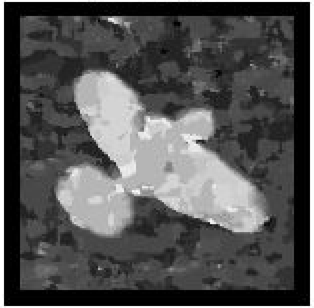


With Gaussian filter:

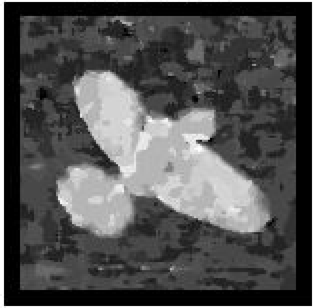
CC 5.06s



NCC 54.73s

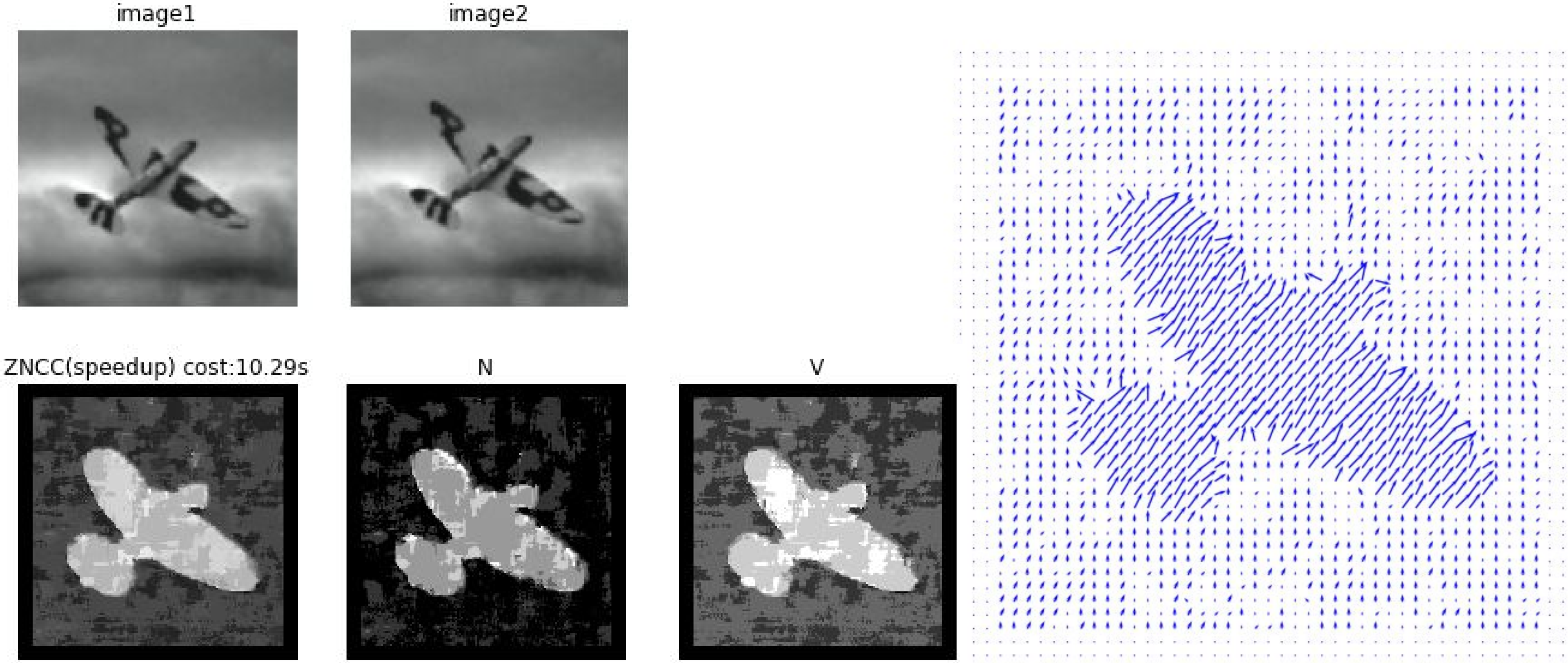


ZNCC 18.02s

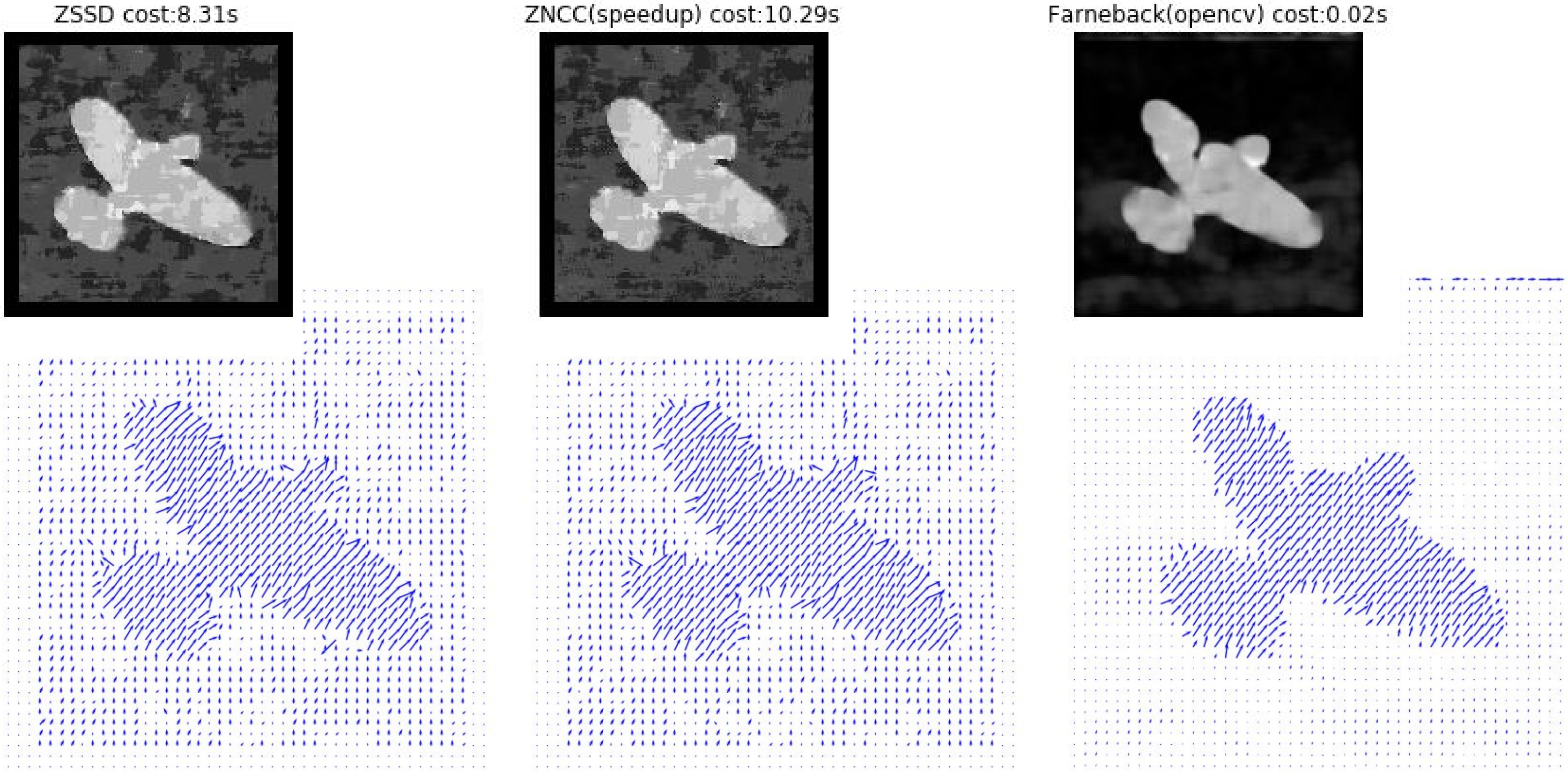


ZNCC (speedup) 11.13s

Belive me I am flying!



Comparison


ZSSD cost:8.31s
ZNCC(speedup) cost:10.29s
Farneback(opencv) cost:0.02s

Differencial-based

& Dense optical flow

Differencial-based Derive

Image brightness constancy[5]:

➡ from a short interval t1 to t2, while an object may change position, the reflectivity and illumination will remain constant.

$$f(x+\triangle x, y+\triangle y, t+\triangle t) \approx f(x, y, t)$$

➡ According to Taylor series expansion:

$$f(x+\triangle x, y+\triangle y, t+\triangle t) = f(x, y, t)\frac{\partial}{\partial x}\triangle x + \frac{\partial}{\partial y}\triangle y + \frac{\partial}{\partial z}\triangle z + h.o.t$$

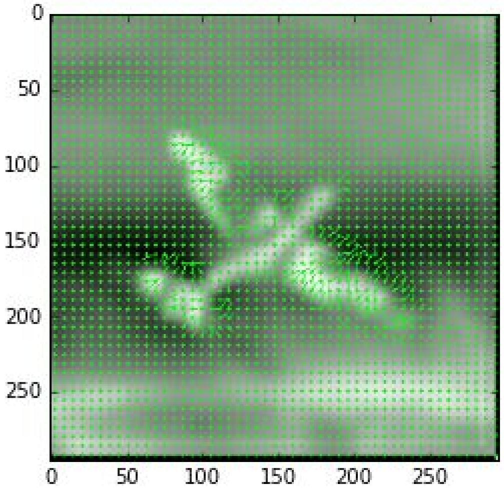
$$\bigtriangledown I \bullet + I_t = 0$$

HORN & SCHUNCK METHOD[6]

use masks to calculate fx, fy,ft:

$$\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$$

mask1 mask2 mask3 mask4



$$e_s = \iint ((u_x^2 + u_y^2) + (v_x^2 + v_y^2))dxdy$$

$$e_c = \iint (I_x u + I_y v + I_t)^2 dxdy$$

minimize $$e = \lambda e_c + e_s$$

LUCAS & KANADE METHOD[6]

$$\min \sum_i (f_{xi}u + f_{yi}v + f_t)^2$$

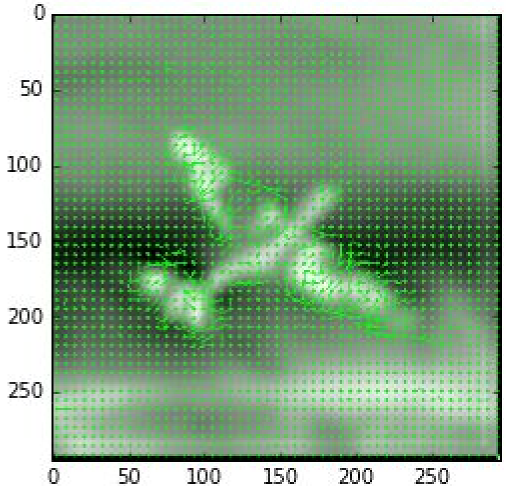
$$\sum (f_{xi}u + f_{yi}v + f_{ti}) f_{xi} = 0$$

$$\sum (f_{xi}u + f_{yi}v + f_{ti}) f_{yi} = 0$$

$$\sum f_{xi}^2 u + \sum f_{xi} f_{yi} v = -\sum f_{xi} f_{ti}$$

$$\sum f_{xi} f_{yi} u + \sum f_{yi}^2 v = -\sum f_{yi} f_{ti}$$

$$\begin{bmatrix} \sum f_{xi}^2 & \sum f_{xi} f_{yi} \\ \sum f_{xi} f_{yi} & \sum f_{xi}^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum f_{xi} f_{ti} \\ -\sum f_{yi} f_{ti} \end{bmatrix}$$



Gunnar Farneback method

Dense optical flow

A optimal method to track all the pixels in an image

Farneback method uses Polynomial Expansion to approximate the neighbors of a pixel[7].

Demo with Dense techniques

- Extract the moving object from the static background[8]
  - Gaussion filter
  - Get background with threshold

- Track the moving object
  - findContours() in opencv
  - Draw the rectangle edge of the moving object

Speed up with numba!

Thanks

Reference

[1] Nixon M. Feature extraction & image processing[M]. Academic Press, 2008.

[2] Giachetti, A., Matching Techniques to Compute Image Motion, Image Vision Comput., 18(3), pp. 247–260, 2000

[3] Correlation based similarity measures-Summary, https://siddhantahuja.wordpress.com/tag/normalized-cross-correlation/

[4] Zero Mean Normalized Cross-Correlatio, https://martin-thoma.com/zero-mean-normalized-cross-correlation/#tocAnchor-1-2

[5] O'Donovan, Peter. "Optical flow: Techniques and applications." The University of Saskatchewan (2005).

[6] Coarse-to-fine Optical Flow, http://eric-yuan.me/coarse-to-fine-optical-flow/

[7] The Gunnar-Farneback optical flow, https://www.safaribooksonline.com/library/view/opencv-essentials/9781783984244/ch07s04.html

[8] SuganyaDevi, K., N. Malmurugan, and R. Sivakumar. "Efficient foreground extraction based on optical flow and SMED for road traffic analysis." *International Journal of Cyber-Security and Digital Forensics (IJCSDF)* 1.3 (2012): 177-182.