Low cost imaging and image processing with the Raspberry Pi

Jamie Magee

Supervised by: Dr Alexandre Kabla

Objectives

- Provide access to optical flow methods on cheap and readily available hardware
- Online and offline processing using the Raspberry Pi
- Capture data using the Raspberry Pi Camera
- Integrate with existing experiments and other OpenLabTools projects
- Provide a clear set of instructions aimed at undergraduate level

OpenCV & Raspberry Pi Camera

OpenCV is a free and open source library which provides easy access to many image processing and computer vision algorithms, including optical flow algorithms.

A precompiled binary does not exist for the Raspberry Pi, so it was necessary to compile it myself and write instructions on how to do so. The compilation time is around 10-12 hours depending on overclock.

The Raspberry Pi by default does not interface with the OS, so an extra driver (UV4l-raspi) was required as well.

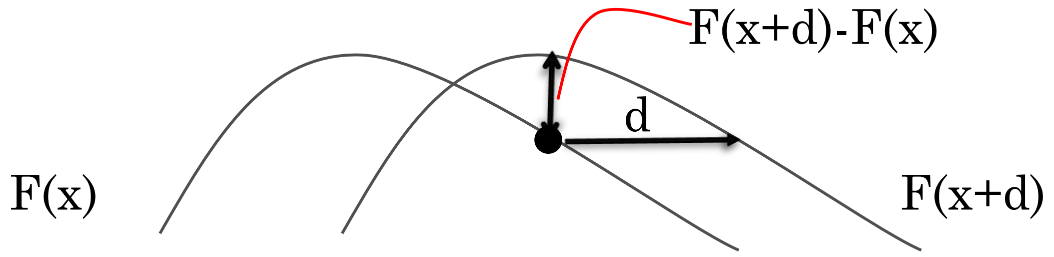
Optical Flow

I have considered three different algorithms:

- Lucas-Kanade [Lucas, B. and Kanade, T., 1981]
- Farnebäck [Farnebäck, G., 2003]
- SimpleFlow [Tao, M., Bai, J., Kohli, P. and Paris, S., 2012]

Lucas-Kanade

Assumes displacement is small and constant



$$F(x) \approx \frac{F(x+d) - F(x)}{d}$$
$$\therefore d \approx \frac{F(x+d) - F(x)}{F(x)}$$

Generalised to 2D:

$$I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1)$$
$$\vdots$$
$$I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2)$$
$$I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n)$$

These can be written in Matrix form $Av = b$ where

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, and\ b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

Lucas-Kanade

The system has more equations than unknowns, so it is usually over-determined

A solution can be found by least squares

$$A^T A v = A^T b$$
$$v = (A^T A)^{-1} A^T b$$

Therefore

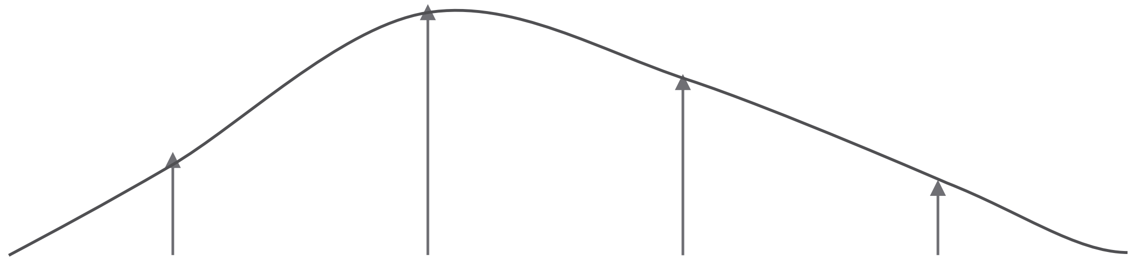
$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_y(q_i) I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix}$$

Farnebäck

Each neighbourhood of pixels is approximated by a quadratic polynomial of the form:

$$f(x) \sim \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^{\boldsymbol{T}}\boldsymbol{x} + c$$

In 1D



Consider the polynomial

$$f_1(x) = \boldsymbol{x}^T \boldsymbol{A_1}\boldsymbol{x} + \boldsymbol{b_1^T}\boldsymbol{x} + c_1$$

Shifted by a global displacement **d**

$$\begin{aligned} f_2(x) &= f_1(\boldsymbol{x} - \boldsymbol{d}) \\ &= (\boldsymbol{x} - \boldsymbol{d})^T \boldsymbol{A_1}(\boldsymbol{x} - \boldsymbol{d}) + \boldsymbol{b_1^T}(\boldsymbol{x} - \boldsymbol{d}) + c_1 \\ &= \boldsymbol{x^T}\boldsymbol{A}\boldsymbol{x} + (\boldsymbol{b_1} - 2\boldsymbol{A_1}\boldsymbol{d})^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{d^T}\boldsymbol{A_1}\boldsymbol{d} - \boldsymbol{b_1^T}\boldsymbol{d} + c_1 \\ &= \boldsymbol{x^T}\boldsymbol{A_2}\boldsymbol{x} + \boldsymbol{b_2^T}\boldsymbol{x} + c_2 \end{aligned}$$

Farnebäck

Which yields

$$\boldsymbol{A_2} = \boldsymbol{A_1}$$
$$\boldsymbol{b_2} = \boldsymbol{b_1} - 2\boldsymbol{A_1}\boldsymbol{d}$$
$$c_2 = \boldsymbol{d^T}\boldsymbol{A_1}\boldsymbol{d} - \boldsymbol{b_1^T}\boldsymbol{d} + c_1$$

Therefore we can solve for the translation $\mathbf{d}$, if $\mathbf{A}_1$ is non-singular

$$2\boldsymbol{A_1}\boldsymbol{d} = -(\boldsymbol{b_2} - \boldsymbol{b_1})$$
$$\boldsymbol{d} = -\frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{A_1^{-1}}(\boldsymbol{b_2} - \boldsymbol{b_1})$$

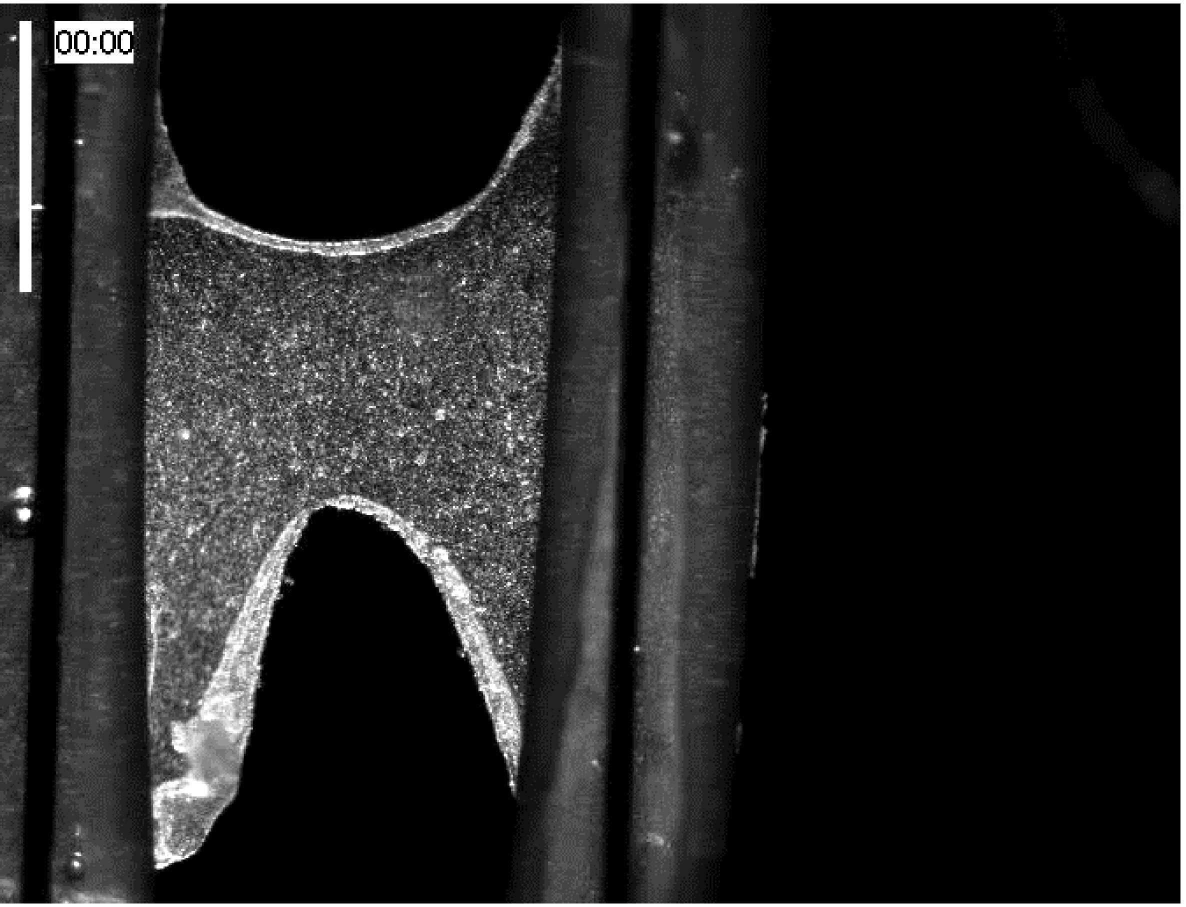
SimpleFlow

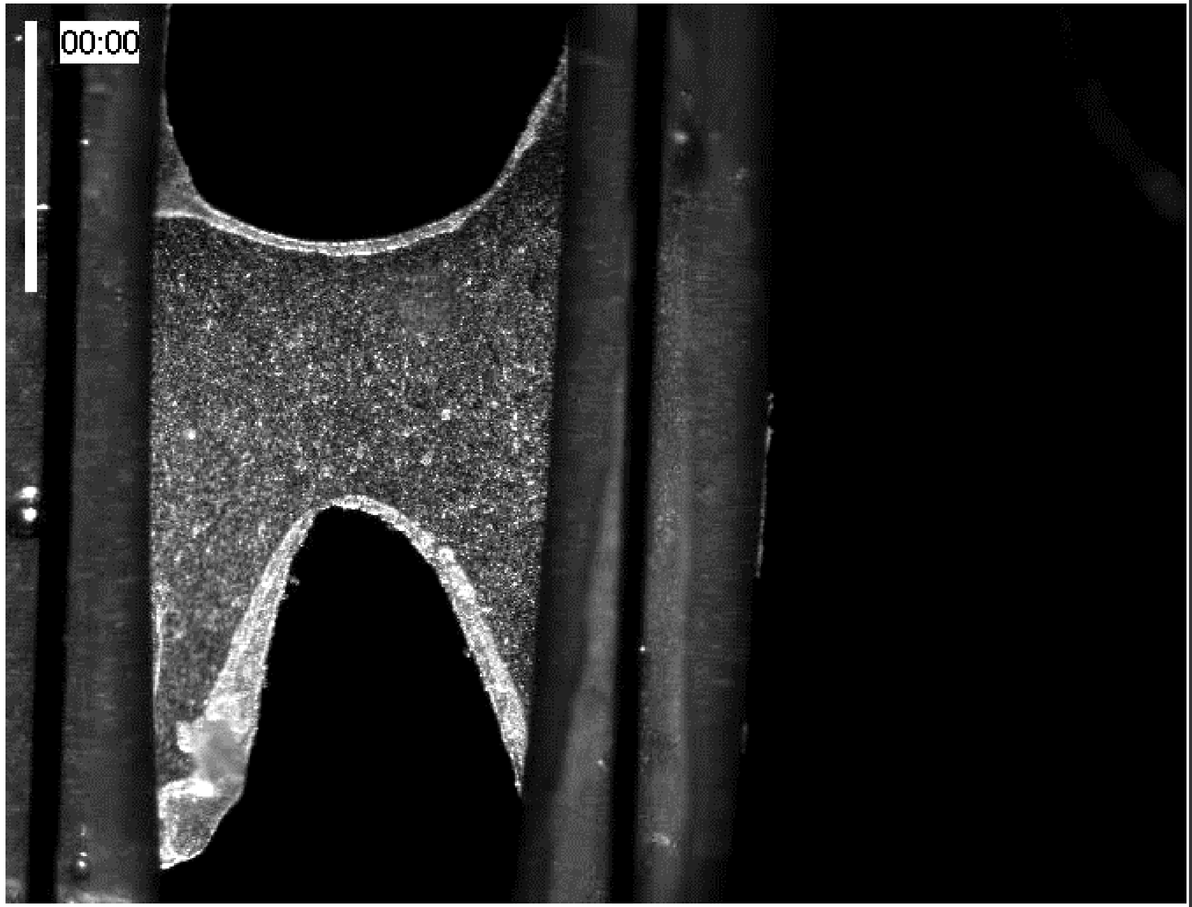
A likelihood model is used, that is, we seek flow vectors (u,v) such that $F_t(x,y)$ and $F_{t+1}(x+u,y+v)$ are similar. This is modelled by the energy term:

$$e(x,y,u,v) = \|F_t(x,y) - F_{t+1(}x+u,y+v)\|^2$$

The algorithm is a *semi-dense* optical flow method, in that it only runs a full flow estimation at a few key pixels, and linearly interpolates the rest of the flow.

Output

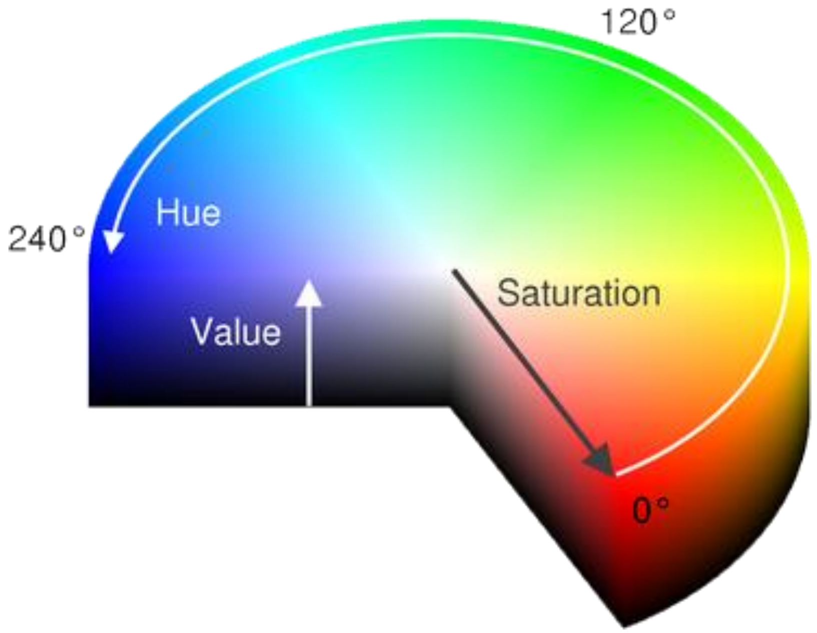


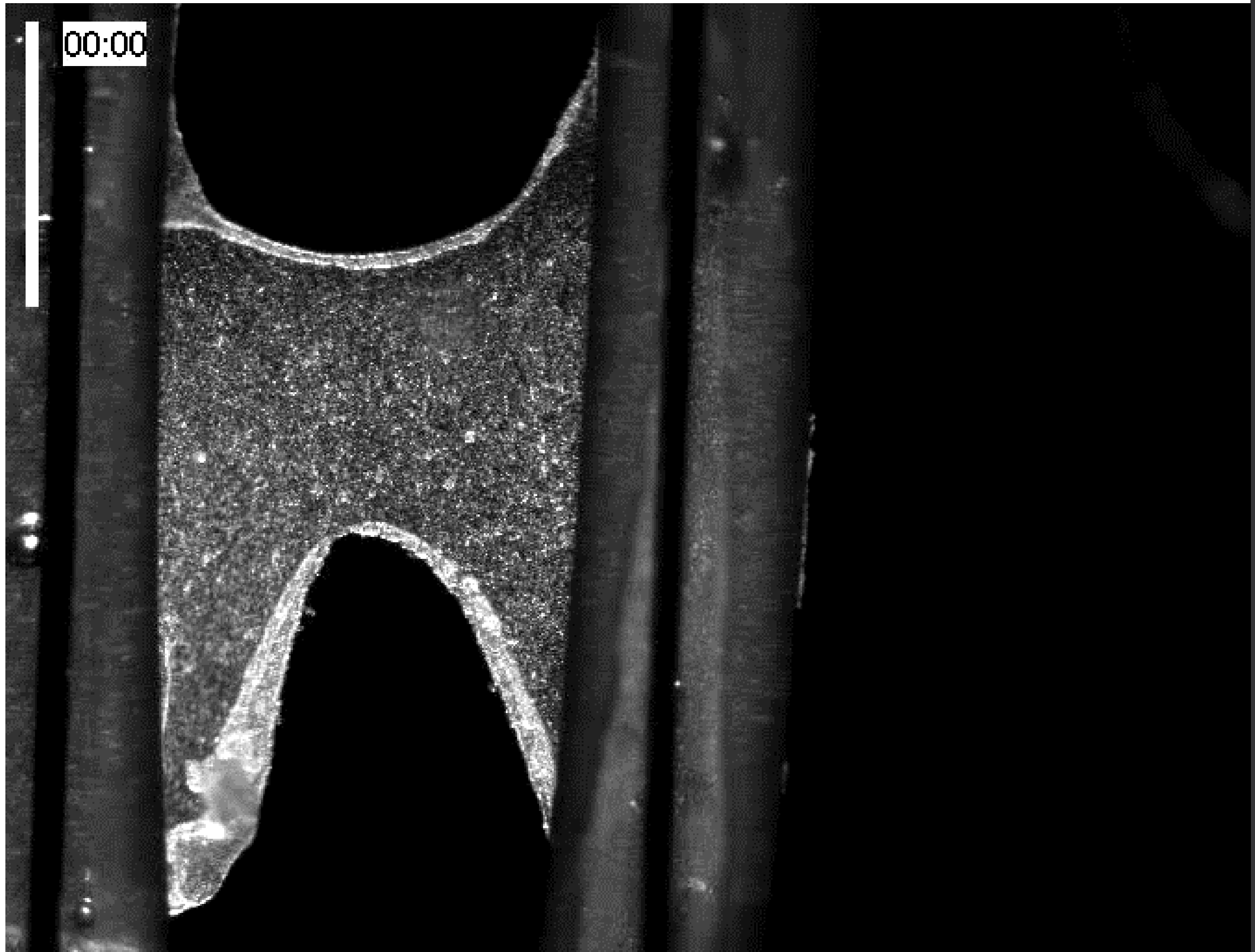


Output

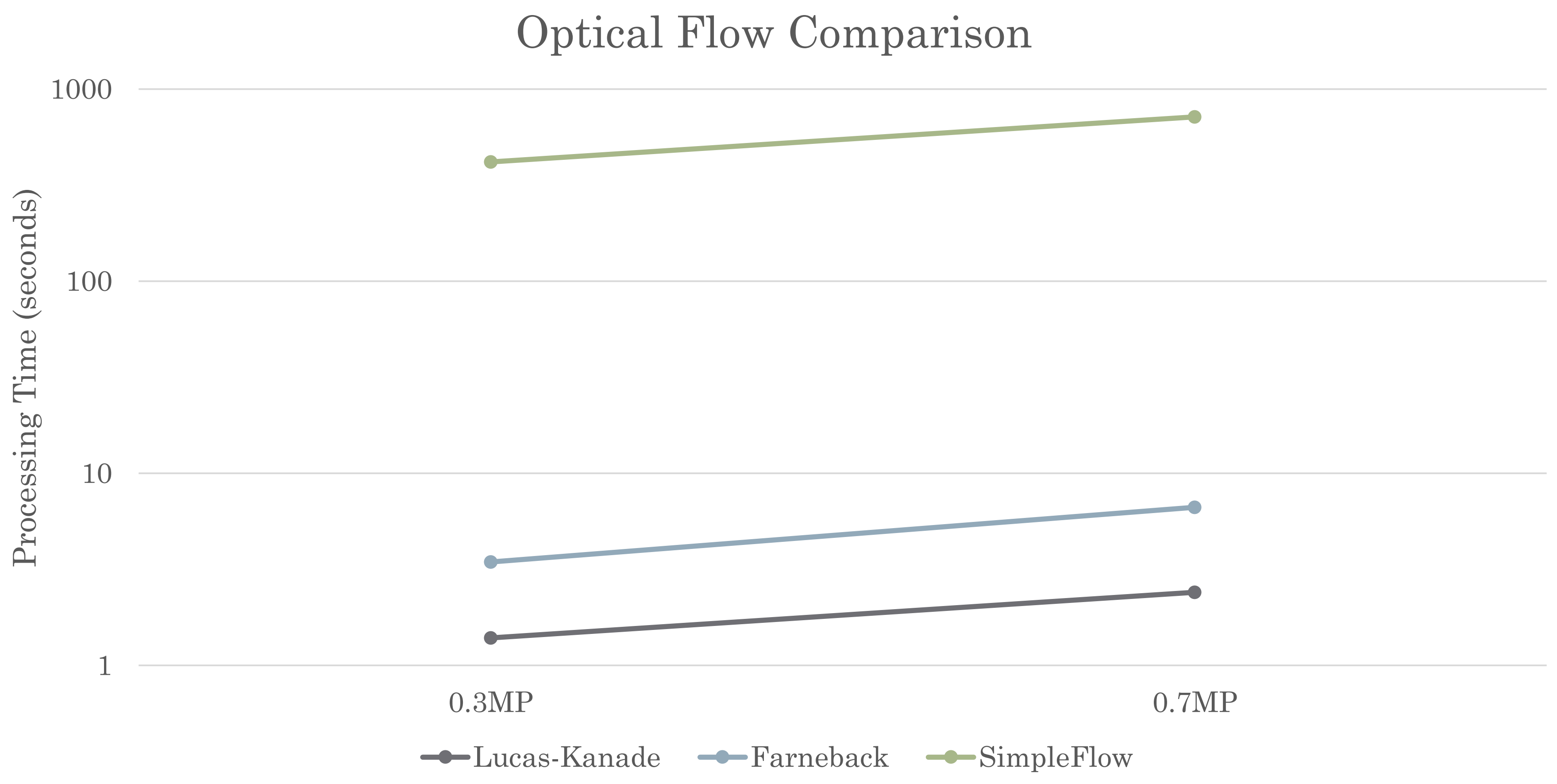
Vectors are converted into polar coordinates and mapped to the HSV colour space as follows:

Hue = Angle
Saturation = 255
Value = Magnitude





Comparison


Optical Flow Comparison
Processing Time (seconds)
1000
100
10
1
0.3MP
0.7MP
Lucas-Kanade
Farneback
SimpleFlow

Improvements

- Find the curl of the vector field
- Find the divergence of the vector field
- Optimise implementation for online processing with Raspberry Pi camera
- Integrate with materials testing (1st year experiment)
- Integrate with other OpenLabTools projects (Microscope)
- Provide a GUI for immediate user feedback

Questions?