Efficient Numerical Estimation of Optical Flow

Semester Project, TMA4205

Thorbjørn Djupvik, Rudolfs Sietinsons & Ivan Nygaard

1 Introduction

When an observer looks at a three-dimensional scene through a camera lens or an eye, the motion in the scene and the relative motions of the lens produce a displacement vector field for each pixel between successive video frames[2][4]. This apparent displacement field is known as optical flow. Optical flow enables the understanding of motion and is therefore a hot area of research because it plays an important role in technologies such as computer vision and video prediction[1][2]. In this project, we attempt to model the optical flow following Horn and Schunck's approach by implementing efficient numerical solutions of the resulting equations by preconditioned conjugate gradient algorithm, using a multigrid V-cycle as preconditioner.

2 Theory

The target problem is to recover the optical flow $\boldsymbol{w}(x,y) := [u(x,y), v(x,y)]$, at some fixed time $t_0 \in \mathbb{R}$ given two consecutive frames in a video $I : \Omega \times \mathbb{R} \to \mathbb{R}$. u and v describe the motion of each pixel in the image plane $\Omega := [0, L_x] \times [0, L_y]$. The first assumption we impose on the problem is that the brightness of a point moving through a scene will remain constant. This approximation is colloquially referred to as brightness consistency. To that end, we may for a sufficiently small timestep write

$$I(x(t_0 + \Delta t), y(t_0 + \Delta t), t_0 + \Delta t) = I(x(t_0), y(t_0), t_0). \tag{1}$$

The left hand side may be linearized by employing a multivariable Taylor expansion about the point $(x(t_0), y(t_0))$

$$\begin{aligned} I(x(t_0 + \Delta t), y(t_0 + \Delta t), t_0 + \Delta t) = I(&x(t_0), y(t_0), t_0) \\ &+ \partial_x I(x(t_0), y(t_0), t_0) \\ &+ \partial_x I[x(t_0 + \Delta t) - x(t_0)] \\ &+ \partial_y I[y(t_0 + \Delta t) - y(t_0)] \\ &+ \partial_t I \Delta t. \end{aligned} \tag{2}$$

Here, the shorthand notation

$$\partial_x I := \left.\frac{\partial I}{\partial x}\right|_{x=x(t_0),\, y=y(t_0),\, t=t_0}, \quad \partial_y I := \left.\frac{\partial I}{\partial y}\right|_{x=x(t_0),\, y=y(t_0),\, t=t_0}, \quad \partial_t I := \left.\frac{\partial I}{\partial t}\right|_{x=x(t_0),\, y=y(t_0),\, t=t_0},$$

has been introduced. Equating (1) and (2), dividing through by Δt and taking the limit

$$\lim_{\Delta t \to 0} \left(\partial_x I \frac{\Delta x(t)}{\Delta t} + \partial_y I \frac{\Delta y(t)}{\Delta t} + \partial_t I \right) = 0, \tag{3}$$

one ends up with

$$u \partial_x I + v \partial_y I = -\partial_t I. \tag{4}$$

Where we have defined

$$u(x,y) := \frac{dx}{dt}(x,y),$$
$$v(x,y) := \frac{dy}{dt}(x,y).$$

Dirichlet-type boundary conditions will be used with $u = v = 0$ on $\partial\Omega$. Equation (4) has two unknowns, an additional constraint must therefore be imposed to obtain a unique solution. This fact is referred to as the aperture problem in optical flow[4]. One possible remedy is to assume that the optical flow is smooth. This assumption entails that u and v change slowly in space and may be imposed by requiring u and v to be the minimizes of

$$\frac{1}{2}\|u\,\partial_x I + v\,\partial_y I + \partial_t I\|^2 + \frac{\lambda}{2}\left(\|\nabla u\|^2 + \|\nabla v\|^2\right) \tag{5}$$

where $\lambda > 0$, for which the optimality condition is known to be coupled the partial differential equations

$$\left.\begin{aligned}(u\partial_x I + v\partial_y I)\partial_x I - \lambda\Delta u &= -\partial_x I\,\partial_t I\\ (u\partial_x I + v\partial_y I)\partial_y I - \lambda\Delta v &= -\partial_y I\,\partial_t I\end{aligned}\right\} \tag{6}$$

on Ω. The spatial derivatives are discretized by forward differences (7), except at the endpoints where a backward difference (8) is used. For $k = 0, 1$

$$\partial_x I_k(x_i,y_j) := \begin{cases} I_k(x_{i+1},y_j) - I_k(x_i,y_j), & \text{if } i < m,\\ I_k(x_m,y_j) - I_k(x_{m-1},y_j), & \text{if } i = m.\end{cases} \tag{7}$$

$$\partial_x I_k(x_i,y_j) := \begin{cases} I_k(x_i,y_{j+1}) - I_k(x_i,y_j), & \text{if } j < n,\\ I_k(x_i,y_n) - I_k(x_i,y_{n-1}), & \text{if } j = n.\end{cases} \tag{8}$$

The temporal derivative is approximated by frame-differencing, i.e.

$$\partial_t I(x_i,y_j) := I_1(x_i,y_j) - I_0(x_i,y_j) \tag{9}$$

Note that we have taken $\Delta t = 1$, $\Delta x = \Delta y =: h = 1$. The discretization of (6) results in

$$\left.\begin{aligned}(\partial_x I)_{ij}^2 u_{ij} + (\partial_y I)_{ij}(\partial_x I)_{ij} v_{ij} - \lambda(\Delta u)_{ij} &= -(\partial_t I)_{ij}(\partial_x I)_{ij},\\ (\partial_y I)_{ij}^2 v_{ij} + (\partial_y I)_{ij}(\partial_x I)_{ij} u_{ij} - \lambda(\Delta v)_{ij} &= -(\partial_t I)_{ij}(\partial_y I)_{ij},\end{aligned}\right\} \tag{10}$$

for $1 \le i \le m, 1 \le j \le n$. Where the Laplacian, Δ, is approximated by the five-point stencil

$$(\Delta u)_{ij} := \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij}}{h^2}. \tag{11}$$

Inserting (11) into (10) and reordering terms, one finds

$$\left.\begin{aligned}\Big[(\partial_x I)_{ij}^2 + \frac{4\lambda}{h^2}\Big]u_{ij} - \frac{\lambda}{h^2}\Big[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}\Big] + (\partial_y I)_{ij}(\partial_x I)_{ij} v_{ij} &= -(\partial_t I)_{ij}(\partial_x I)_{ij},\\ \Big[(\partial_y I)_{ij}^2 + \frac{4\lambda}{h^2}\Big]v_{ij} - \frac{\lambda}{h^2}\Big[v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1}\Big] + (\partial_y I)_{ij}(\partial_x I)_{ij} u_{ij} &= -(\partial_t I)_{ij}(\partial_y I)_{ij}.\end{aligned}\right\} \tag{12}$$

Which may be written on the form

$$\underbrace{\begin{bmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22}\end{bmatrix}}_{\boldsymbol{A}}\underbrace{\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{v}\end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix}\boldsymbol{b}_1\\ \boldsymbol{b}_2\end{bmatrix}}_{\boldsymbol{b}}, \tag{13}$$

with $\boldsymbol{A}_{11}$ and $\boldsymbol{A}_{22}$ have the same structure as the two-dimensional Laplacian, but respectively with $((\partial_x I)_{ij}^2 + \frac{4\lambda}{h^2})$ and $((\partial_y I)_{ij}^2 + \frac{4\lambda}{h^2})$ as diagonal elements and $-\frac{\lambda}{h^2}$ on the off diagonals. $\boldsymbol{A}_{12}$ and $\boldsymbol{A}_{21}$ are diagonal matrices

with $(\partial_x I)_{ij}(\partial_y I)_{ij}$ as the diagonal elements. For the indexing of $\boldsymbol{u}$ and $\boldsymbol{v}$ a consecutive row mapping is used, i.e. $(i,j) \mapsto (i-1)n + (j-1)$. We quickly note that (12) implies that $\boldsymbol{A}$ is not diagonally dominant unless $(\partial_x I)_{ij} = (\partial_y I)_{ij}$. In this case it is also irreducibly diagonally dominant. $\boldsymbol{A}$ can however be shown to be symmetric positive definite, which is of great consequence for the convergence of Gauß–Seidel (GS) and the conjugate gradient (CG).

Theorem 2.1. *Let $\mathcal{V}$ be an innerproduct space. If $A : \mathcal{V} \to \mathcal{V}$ is symmetric positive definite operator and $B : \mathcal{V} \to \mathcal{V}$ is a symmetric positive semi-definite operator. Then the sum $A + B : \mathcal{V} \to \mathcal{V}$ is a symmetric positive definite operator.*

Proof. First we prove that the operator is symmetric. Let $v, w \in \mathcal{V}$. Then

$$\langle (A+B)v, w\rangle = \langle Av, w\rangle + \langle Bv, w\rangle = \langle v, Aw\rangle + \langle v, Bw\rangle = \langle v, (A+B)w\rangle$$

Then we prove that it is positive definite.

$$\langle (A+B)v, v\rangle = \langle Av, v\rangle + \langle Bv, v\rangle$$

where $\langle Av, v\rangle > 0$, and $\langle Av, v\rangle \geq 0$. This again implies the sum above is strictly larger than zero and our claim follows. □

Theorem 2.2. *The matrix $\boldsymbol{A}$ is symmetric positive definite.*

Proof. We assume it to be known that the discrete negative laplacian operator is symmetric positive definite for Dirichlet boundary conditions. From (2.1) we only need to prove that the remaining part is symmetric positive semi-definite. In this proof we will look at the remainder of $\boldsymbol{A}$ as an operator $\boldsymbol{D} : \mathbb{R}^{n\times m} \times \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m} \times \mathbb{R}^{n\times m}$ instead of as a matrix. To obtain an inner product that is consistent with the matrix formulation, we define it as

$$\langle w, p\rangle = \langle (w^{(u)}, w^{(v)}), (p^{(u)}, p^{(v)})\rangle = \sum_{i,j} w_{ij}^{(u)} p_{ij}^{(u)} + w_{ij}^{(v)} p_{ij}^{(v)} \tag{14}$$

First we show that it is symmetric.

$$\begin{aligned}
&\langle Dw, p\rangle \\
&= \sum_{i,j} \left[(\partial_x I)_{i,j}^2 w_{ij}^{(u)} + (\partial_y I)_{i,j}(\partial_x I)_{i,j} w_{i,j}^{(v)}\right] p_{ij}^{(u)} + \left[(\partial_y I)_{i,j}^2 w_{ij}^{(v)} + (\partial_y I)_{i,j}(\partial_x I)_{i,j} w_{i,j}^{(u)}\right] p_{ij}^{(v)} \\
&= \sum_{i,j} \left[(\partial_x I)_{i,j}^2 p_{ij}^{(u)} + (\partial_y I)_{i,j}(\partial_x I)_{i,j} p_{i,j}^{(v)}\right] w_{ij}^{(u)} + \left[(\partial_y I)_{i,j}^2 p_{ij}^{(v)} + (\partial_y I)_{i,j}(\partial_x I)_{i,j} p_{i,j}^{(u)}\right] w_{ij}^{(v)} \\
&= \langle w, Dp\rangle
\end{aligned}$$

Now we show positive semi-definiteness.

$$\begin{aligned}
&\langle Dw, w\rangle \\
&= \sum_{i,j} \left[(\partial_x I)_{i,j}^2 w_{ij}^{(u)} + (\partial_y I)_{i,j}(\partial_x I)_{i,j} w_{i,j}^{(v)}\right] w_{ij}^{(u)} + \left[(\partial_y I)_{i,j}^2 w_{ij}^{(v)} + (\partial_y I)_{i,j}(\partial_x I)_{i,j} w_{i,j}^{(u)}\right] w_{i,j}^{(v)} \\
&= \sum_{i,j} ((\partial_x I)_{i,j}^2 (w_{ij}^{(u)})^2 + 2(\partial_y I)_{i,j}(\partial_x I)_{i,j} w_{i,j}^{(v)} w_{ij}^{(u)} + (\partial_y I)_{i,j}^2 (w_{ij}^{(v)})^2 \\
&= \sum_{i,j} \left[(\partial_x I)_{i,j} w_{ij}^{(u)} + (\partial_y I)_{i,j} w_{ij}^{(v)}\right]^2 \geq 0.
\end{aligned}$$

Which proves our claim. □

Further, this implies that convergence for the Gauß-Seidel algorithm is guaranteed, as Gauß-Seidel converges on symmetric positive definite matrices. However, later on we want to use the multigrid method as a preconditioner in the preconditioned conjugate gradient method. We assume it to be known that preconditioners for PCG need to be symmetric positive definite as well, which is achieved if we modify our Gauß-Seidel routine to be a symmetric Gauß-Seidel routine instead.

3 Implementation

The overall workflow is illustrated in figure 1. The function `preprocessing_image` reads two images, smooths them using the gaussian filter function from scipy.ndimage, and scales them. It returns the spatial derivatives and the temporal derivative that are computed based on the 7,8 and 9 applied to the discretized grid where each gridpoint is centered in a pixel of the two preprocessed frames, the right hand sides of (12), and the grid spacing in x and y-dimensions. The function `preprocessing_test_images` constructs synthetic test images of two Gaussian's circling each other that have size $2^k \times 2^k$, for $k = 6, 7, 8, 9$ computes the spatial and temporal derivatives in the same manner as `preprocessing_image`. The precomputed derivatives are used as input values to the different solvers.

Algorithm 1 CG

Compute $r_0 := b - Ax_0,\ p_0 := r_0$
for $j = 0, 1, \ldots,$ until convergence **do**
    $\alpha_j \leftarrow \frac{(r_j, r_j)}{(Ap_j, p_j)}$
    $x_{j+1} \leftarrow x_j + \alpha_j p_j$
    $r_{j+1} \leftarrow r_j - \alpha_j A p_j$
    $\beta_j \leftarrow \frac{(r_{j+1}, r_{j+1})}{(r_j, r_j)}$
    $p_{j+1} \leftarrow r_{j+1} + \beta_j p_j$
end for

With regards to the `CG`-method, two implementations have been written. Both implementing the `CG`-method as given on p. 190 in [3]. One constructs the linear system using sparse matrices, `OF_cg`, and the other works directly on the grid, `cg`. From algorithm 1 wee see that its complexity is on the order of $\mathcal{O}(n^2)$ per iteration. `V_cycle` implements a multigrid V-cycle, and calls on one of the `CG`-methods at the coarsest level to solve the error equation. It is implemented recursively and calls on the sub-routines `smoothing`, `residual`, `restriction` and `prolongation`. Here `prolongation` interpolates the correction computed on a coarser grid into a finer grid, and `restriction` takes the residual error to the coarser grid. `smoothing` reduces the high frequency error components by applying iterations of the Red-Black Gauß-Seidel method. `PCG` accelerates the convergence of the `CG`-method by employing a multigrid V-cycle as a preconditioner, as shown below.

Algorithm 2 PCG

Compute $r_0 := b - Ax_0,\ z_0 := M^{-1}r_0,\ p_0 := z_0$
for $j = 0, 1, \ldots,$ until convergence **do**
    $\alpha_j \leftarrow \frac{(r_j, z_j)}{(Ap_j, p_j)}$
    $x_{j+1} \leftarrow x_j + \alpha_j p_j$
    $r_{j+1} \leftarrow r_j - \alpha_j A p_j$
    $z_{j+1} \leftarrow M^{-1} r_{j+1}$
    $\beta_j \leftarrow \frac{(r_{j+1}, z_{j+1})}{(r_j, z_j)}$
    $p_{j+1} \leftarrow z_{j+1} + \beta_j p_j$
end for

On the note of Red-Black Gauß-Seidel. The idea is to create a Gauß-Seidel method which can be parallelized by updating entries which are independent of each other at the same time in parallel. In the equation we are solving, we have a 5-point stencil describing the dependence of u, v on other grid-points of u, v respectively. In addition we have that u_{ij}, v_{ij} are interdependent as well. In other words, if we do a `Red-update` on u, it will be independent of a `Black-update` on v, but not independent of `Red-Update` of v. This results in the following `Red-Black` updating scheme.

Algorithm 3 Red-Black

1: `Red-Update`(u)
2: `Black-Update`(v)
3: `Red-Update`(v)
4: `Black-Update`(u)

The flow of information is depicted as in figure 1 when using the `PCG`-algorithm to solve the problem.

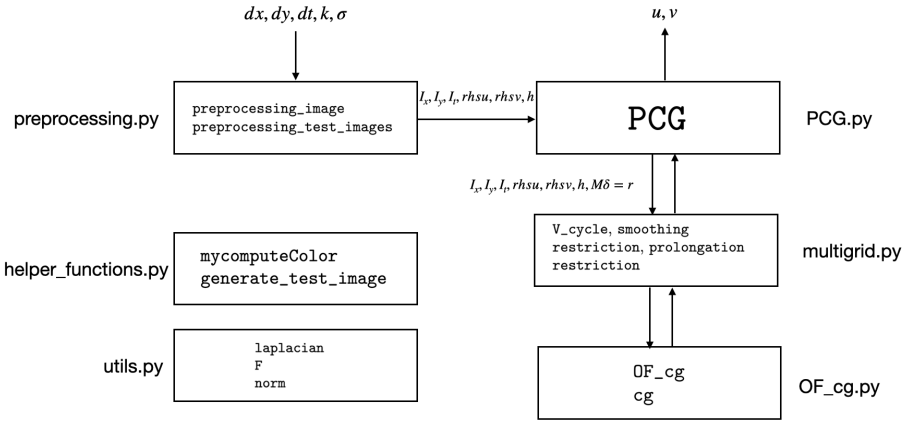
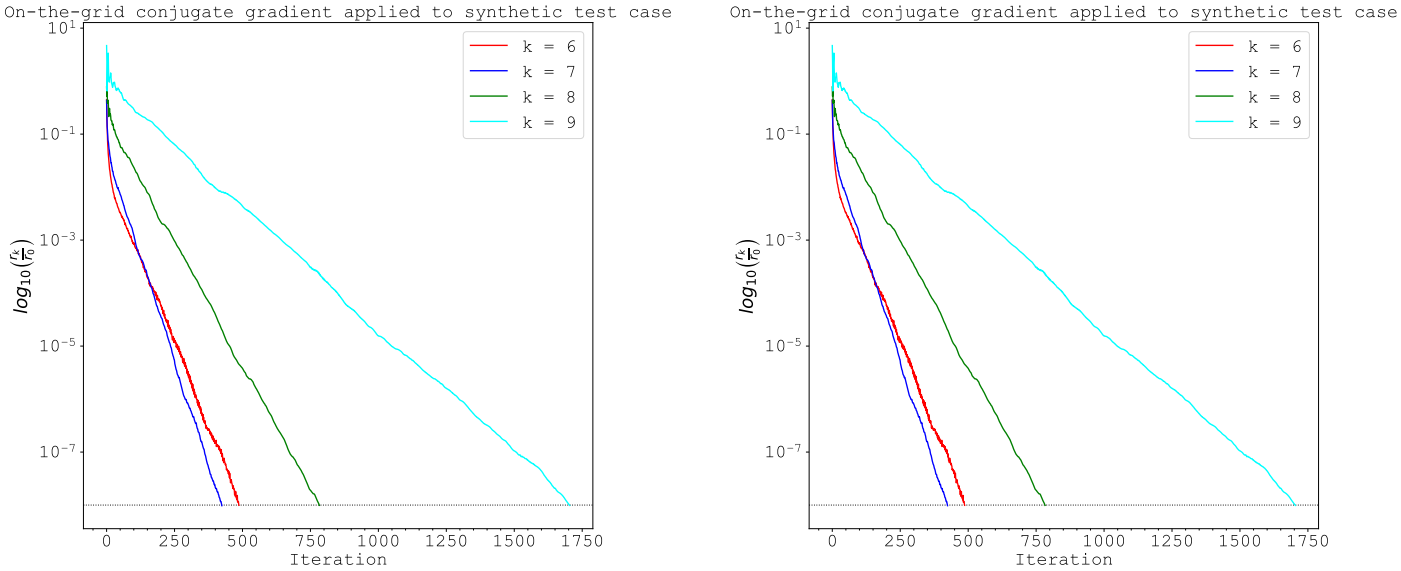


Figure 1: Schematic of the workflow.

4 Results and Discussion

In the following, results for a numerical experiment is presented. The study was conducted on a MacBook Air with an Apple M1 chip and 8 GB of RAM. The two implementations of the Conjugate Gradient algorithm were first tested against each other without using multigrid for convergence comparison on a synthetic test case of two Gaussian's circling each other, the image sizes were set to $2^k \times 2^k$, for $k = 6, 7, 8, 9$. The convergence results are presented in figure 2 while the test images and the reconstructed optical flow field is presented in figures 6 and 7 in appendix A.



(a) Convergence plot of sparse implementation.

(b) Convergence plot of on-the-grid implementation.

Figure 2: Convergence plots of sparse vs on-the-grid implementations of the Conjugate Gradient algorithm, without multigrid, tested on two synthetic image frames depicting two Gaussian's circling each other. The image sizes were $2^k \times 2^k$, for $k = 6, 7, 8, 9$.

k	t_{SCG} [s]	t_{GCG} [s]	t_{SCG}/t_{GCG}
6	0.047	0.086	0.546
7	0.131	0.118	1.110
8	0.988	0.815	1.212
9	9.022	9.281	0.972

Table 1: Timing comparison for sparse (t_{SCG}) vs. on-the-grid (t_{GCG}) implementations of the Conjugate Gradient algorithm without multigrid, tested on two synthetic image frames depicting two Gaussians circling each other. Image sizes are $2^k \times 2^k$ for $k = 6, 7, 8, 9$.

From figure 2 and table 1 we see that the two implementations perform approximately equally well, with the sparse implementation having a small edge for the smallest and largest k-values for the given numerical experiment. As expected, both implementations converge faster for smaller image (grid) - sizes. In addition to this, we see that for that the number of iterations needed for $k = 6$ and $k = 7$ is roughly equal. For $k > 7$ this is no longer true, and the number of iterations starts to grow. Proceeding with the multigrid-method, only one implementation is needed, and we chose to continue with the grid-based implementation for the V-cycle. In any case, running the sparse and on-the-grid algorithms side by side allowed us to debug inconsistencies and confirm robustness. After this, the V-cycle was tested on the same synthetic test case of Gaussian's circling each other. The reconstructed optical flow fields were identical to those presented in figure 7 given in appendix A.

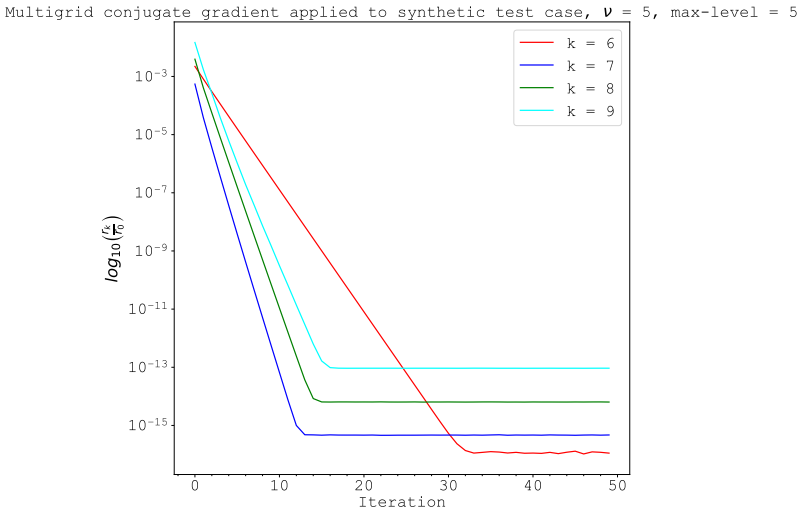
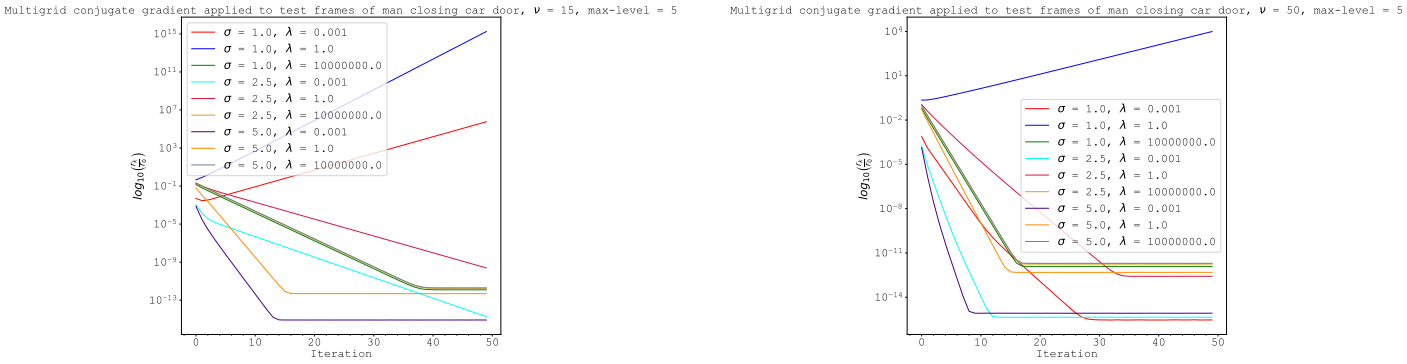


Figure 3: Convergence plot for multigrid conjugate gradient applied to the synthetic test case of two Guassian's circling each other for image sizes of $2^k \times 2^k$ for $k = 6, 7, 8, 9$.

However, as can be seen from figure 3, the convergence rate was faster, but the same trends observed in figure 2 with respect to the dependence of the convergence rates on grid-size is no more, and the convergence rate seems rather arbitrary. Following this, the V-cycle was tested on two provided test frames of a man closing his car door, see figure 8 in appendix A, for different combinations of standard deviations, σ, used in the smoothing in the preprocessing, and for different values of λ. The resulting optical flow fields are presented in figure 9 in appendix A. It appears that λ acts as a diffusion constant, where larger values result in a more dispersed optical flow field. The diffusive effect λ has appears to be most significant for $0.001 \leq \lambda \leq 1.0$. σ also disperses the field, which is expected, but in figure 9(g) it appears as if a higher σ also results in greater contrast. With respect to the runtime, it is approximately equal for the different combinations of λ and σ (when $\nu = 15$) as can be observed in table 2. Especially for $nu = 15$ a larger σ seems to decrease the runtime by a small amount.



(a) Convergence plot for multigrid V-cycle for different values of σ and λ, $\nu = 50$, max-lvl $= 5$.

(b) Convergence plot for multigrid V-cycle for different values of σ and λ, $\nu = 15$, max-lvl $= 5$.

Figure 4: Convergence plots for for multigrid conjugate gradient for various σ and λ values.

σ	λ	$t_{\nu=15,\,\text{lvl}=5}$	$t_{\nu=50,\,\text{lvl}=5}$	$t_{\nu=15,\,\text{lvl}=2}$
1.0	10^{-3}	0.4135	1.2325	0.5367
1.0	10^{0}	0.4008	1.2472	0.5177
1.0	10^{7}	0.3988	1.3043	0.5087
2.5	10^{-3}	0.3934	1.2449	0.4848
2.5	10^{0}	0.3951	1.2290	0.5195
2.5	10^{7}	0.3991	1.2549	0.5073
5.0	10^{-3}	0.3891	1.2478	0.4470
5.0	10^{0}	0.3826	1.3599	0.5340
5.0	10^{7}	0.3982	1.2861	0.5144

Table 2: Runtimes for multigrid V-cycle for various (σ, λ) and smoothing settings.

It is an unfortunate fact of life, that our implementation of the V-cycle diverges for small combinations of λ and σ values, see figure 4. Specifically, divergence was observed during the numerical experiments for $\sigma, \lambda = 1.0, 0.001$ and $\sigma, \lambda = 1.0, 0.001$ when the number of iterations for the smoother ν, was set to 15. The number of iterations for the smoother was then increased from $\nu = 15$ to $\nu = 50$. The results are presented in figure 4 and table 2. Increasing the number of smoothing iterations increased the run time, but did not completely fix the bad behavior. For $\nu = 50$ a clearer trend may be observed with respect to the runtime and value of λ, where the runtime increases for larger λ. We also tried to change the max-level, which increased the runtime by a small amount, see table $2(a)$, as this means more work for the conjugate gradient. Finally, a preconditioned conjugate gradient algorithm, using the multigrid as preconditioner, was tested on the provided frames of a man closing a car door with $\sigma = 1.0$ and $\lambda = 5.0$. The convergence plot 5 shows how the preconditioner drastically increases the convergence rate. Comparing times, the non-preconditioned conjugate gradient used 12.210s while the preconditioned used 4.325s for the given experiment.

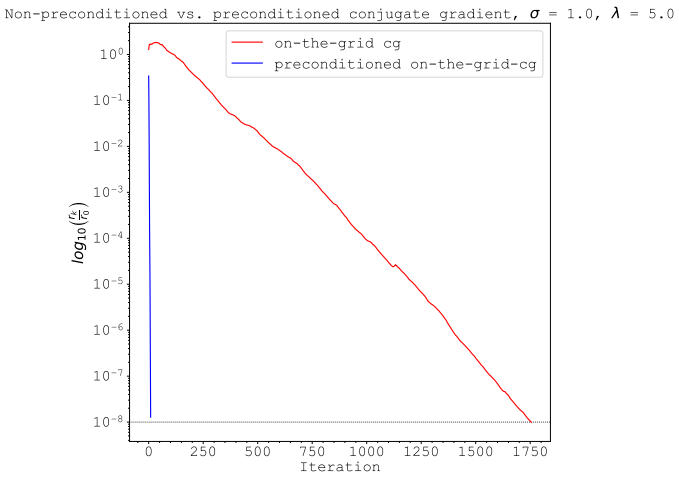
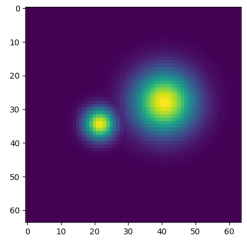


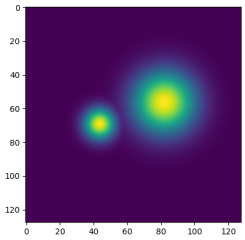
Figure 5: Convergence plot comparing the non-preconditioned on-the-grid implementation of the Conjugate Gradient algorithm vs. the preconditioned version for the test frames of a man closing a car door.

In summary, the main observations were that the on-the-grid and sparse implementations perform approximately equally well. By introducing the multigrid V-cycle, the convergence rate increases in general. However, for synthetic tests, the dependence on the grid size becomes irregular. For real images, larger λ produces more dispersed flow fields, and the convergence rate slows somewhat as σ and λ increase (for $\nu = 50$). Using the V-cycle as a preconditioner significantly improves performance, reducing the CG time drastically from 12.210s to 4.325s. The best runtime we got was 0.3826s (see table 2) for $\lambda = 1$, $\sigma = 2.5$ with $\nu = 15$ and the max-level set to 5.

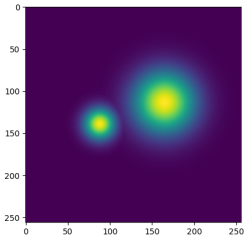
A Appendix A - Optical Flow Field Plots



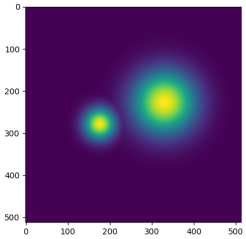
(a) k = 6



(b) k = 7

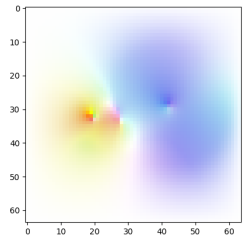


(c) k = 8

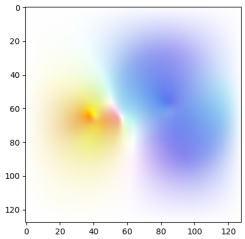


(d) k = 9

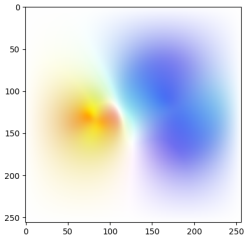
Figure 6: Second frame from synthetic test images of Gaussian's circling each other for image sizes of $2^k \times 2^k$ for $k = 6, 7, 8, 9$.



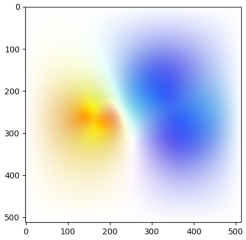
(a) k = 6



(b) k = 7



(c) k = 8

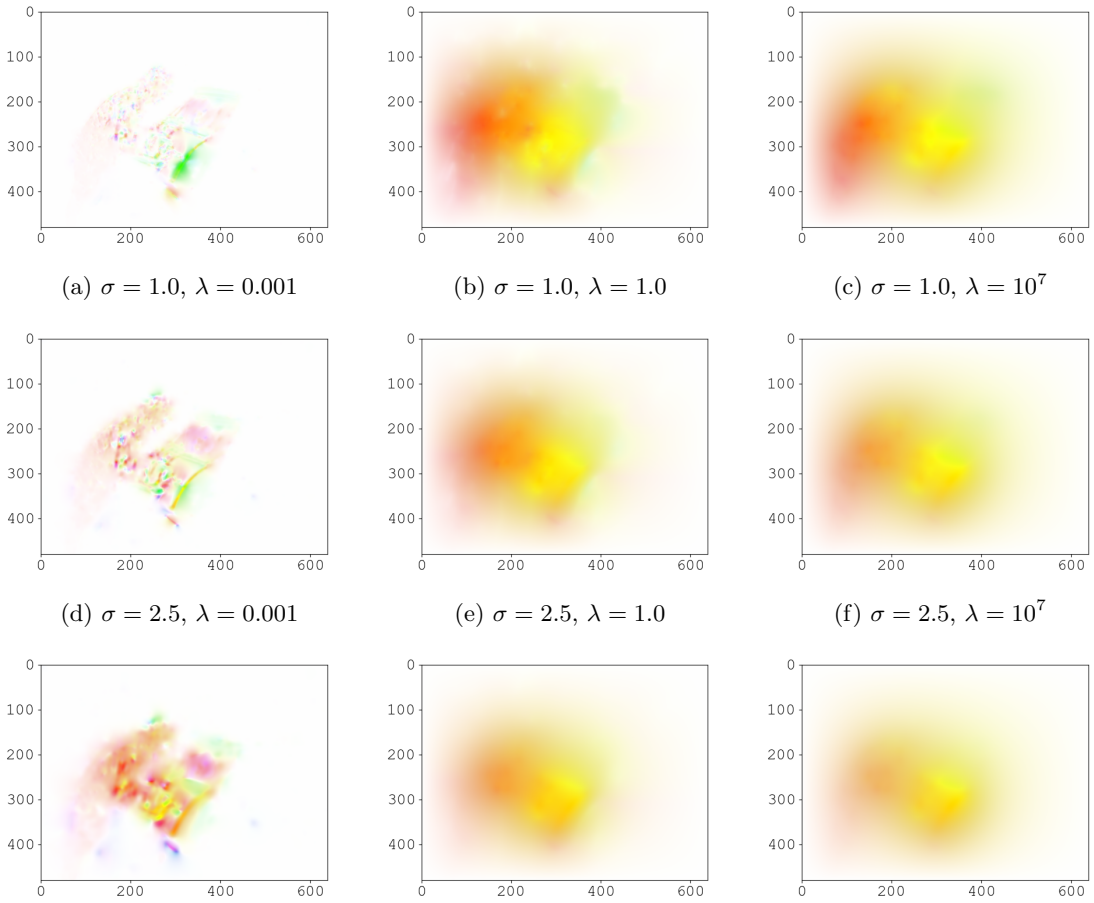


(d) k = 9

Figure 7: Resulting optical flow fields for the test case of two Gassians circling each other for image sizes of $2^k \times 2^k$ for $k = 6, 7, 8, 9$ using the sparse implementation of the conjugate gradient algorithm (on-the-grid implementation yields identical results).

(a) First frame used for testing. (b) Second frame used for testing.

Figure 8: The two frames of a man closing his car door which were used to test the implementation.



(a) $\sigma = 1.0$, $\lambda = 0.001$ (b) $\sigma = 1.0$, $\lambda = 1.0$ (c) $\sigma = 1.0$, $\lambda = 10^7$

(d) $\sigma = 2.5$, $\lambda = 0.001$ (e) $\sigma = 2.5$, $\lambda = 1.0$ (f) $\sigma = 2.5$, $\lambda = 10^7$

(g) $\sigma = 5.0$, $\lambda = 0.001$ (h) $\sigma = 5.0$, $\lambda = 1.0$ (i) $\sigma = 5.0$, $\lambda = 10^7$

Figure 9: Resulting optical flow field for the provided test frames of a man closing his car door using the multigrid conjugate gradient (on-the-grid, not sparse) algorithm. The number of iterations for the smoother was set to $\nu = 15$ and the max-level for the V-cycle was set to 5.

A Appendix B - Code

Code can be found at https://github.com/IvanNygaard/Efficient-Optical-Flow.

References

[1] Andrea Alfarano, Luca Maiano, Lorenzo Papa, and Irene Amerini. Estimating optical flow: A comprehensive review of the state of the art. *Computer Vision and Image Understanding*, 2024. DOI: https://doi.org/10.1016/j.cviu.2024.104160.

[2] Qiaole Dong and Yanwei Fu. Memflow: Optical flow estimation and prediction with memory. *arXiv preprint arXiv:2404.04808*, 2024. DOI: https://doi.org/10.48550/arXiv.2404.04808.

[3] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2nd edition, 2003.

[4] Tianfan Xue, Hossein Mobahi, Frédo Durand, and William T. Freeman. The aperture problem for refractive motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9. IEEE, 2015.