Practical Course GPU Programming Project Assignment Superresolution winter semester 2012/13

Martin Oswald, Frank Steinbruecker

March 15, 2013

1 Assignment

Your assignment in this project is to implement a variational motion estimation / optical flow method as well as a superresolution method using the NVIDIA CUDA Framework.

Comment your code well and hand it in at the end of your project.

Prepare a 30min presentation about your work explaining the benefit of parallel computation for the methods you have implemented and display the runtime results of your method. The presentation should include a short live demonstration of your program as well.

2 Theoretical Details

2.1 Optical Flow

2.1.1 Original Quadratic Approach

The general idea of optical flow is that given two images $I_1, I_2 : \Omega \to \mathbb{R}$, one computes a vector field $u : \Omega \to \mathbb{R}^2$, that matches the image intensities:

$$I_2(\mathbf{x} + u(\mathbf{x})) = I_1(\mathbf{x})$$

Naturally, for an intensity value in a pixel in I_1 there may be many pixels in I_2 that have the same value. Also, there might be none at all. Therefore, it is common to formulate the problem as a variational approach with a data term penalizing deviations from the intensity values and a regularity term penalizing spatial variations of the flow field:

$$E(u) = \int_\Omega (I_2(\mathbf{x} + u(\mathbf{x})) - I_1(\mathbf{x}))^2 + \lambda |\nabla u|^2 \quad \mathrm{d}\mathbf{x}$$

For the two-dimensional flow field $u(\mathbf{x}) = [u_1(\mathbf{x}), u_2(\mathbf{x})]^\top$, the gradient magnitude $|\nabla u|$ is defined as

$$|\nabla u| = \sqrt{{u_1}_x^2 + {u_1}_y^2 + {u_2}_x^2 + {u_2}_y^2}$$

where the subindices $_x$ and $_y$ indicate derivatives in x and y direction.
The minimum of this energy is a flow field that is smooth and matches the image intensities reasonably well.
The equation above however is highly non-convex, since the desired argument u is an argument of the image intensities I. Therefore, a typical approach is to approximate $I_2(\mathbf{x} + u(\mathbf{x}))$ by a first order Taylor expansion. For this, I_1 is regarded as an image at time t, and I_2 is regarded as an image at time $t + 1$.:

$$I_2(\mathbf{x} + u(\mathbf{x})) = I(\mathbf{x} + u(\mathbf{x}), t + 1) \approx I(\mathbf{x}, t) + \nabla I^\top u + \underbrace{I_t}_{\frac{\mathrm{d}I}{\mathrm{d}t}} \tag{1}$$

Concerning the implementation, the temporal derivative I_t is simply the difference $I_2 - I_1$ at each pixel, while the gradient ∇I is usually averaged over both images for each pixel, since it is arbitrary whether one chooses to linearize the spatial displacement at time t or at time $t + 1$. Plugging this into the equation, we get

$$\begin{aligned} E(u) &= \int_\Omega (I(\mathbf{x}, t) + \nabla I^\top u + I_t - I(\mathbf{x}, t))^2 + \lambda |\nabla u|^2 \quad \mathrm{d}\mathbf{x} \\ &= \int_\Omega (\nabla I^\top u + I_t)^2 + \lambda |\nabla u|^2 \quad \mathrm{d}\mathbf{x} \end{aligned} \tag{2}$$

This is the method of Horn, B. and Schunck, B. 1981. “Determining optical flow.” in ”Artificial Intelligence“, 17:185-203.(You do not have to read this)

For the two-dimensional flow field $u(\mathbf{x}) = [u_1(\mathbf{x}), u_2(\mathbf{x})]^\top$, one gets two Euler-Lagrange Equations:

$$\begin{aligned} 0 &= L_{u_1} - \frac{\partial}{\partial x} L_{{u_1}_x} - \frac{\partial}{\partial y} L_{{u_1}_y} \\ 0 &= L_{u_2} - \frac{\partial}{\partial x} L_{{u_2}_x} - \frac{\partial}{\partial y} L_{{u_2}_y} \end{aligned}$$

where the subindices stand for the partial derivative of the Lagrangian integrand with respect to the partial derivative of u. In the case of the Horn and Schunck equation the terms read the following way:

$$\begin{aligned} L_{u_1} &= 2(I_x^2 u_1 + I_x I_y u_2 + I_x I_t) \\ L_{u_2} &= 2(I_x I_y u_1 + I_y^2 u_2 + I_y I_t) \\ L_{u_{1_x}} &= 2\lambda u_{1x} \\ L_{u_{2_x}} &= 2\lambda u_{2x} \\ L_{u_{1_y}} &= 2\lambda u_{1y} \\ L_{u_{2_y}} &= 2\lambda u_{2y} \end{aligned}$$

The Euler-Lagrange equations then read

$$\begin{aligned} 0 &= I_x^2 u_1 + I_x I_y u_2 + I_x I_t - \lambda \Delta u_1 \\ 0 &= I_x I_y u_1 + I_y^2 u_2 + I_y I_t - \lambda \Delta u_2 \end{aligned}$$

Since 1981, there have been developed a large amount of improvements of this method, some of which you are going to implement as well. As a reference, read Papenberg et al. "Highly accurate optical flow computation with theoretically justified warping.". (**http://www.mia.uni-saarland.de/Publications/papenberg-ijcv06.pdf**) Except for advanced constancy assumptions and temporal smoothness of the flow field, your task is to implement the method presented in this paper, with the features listed below.

2.1.2 Advanced Penalty Functions

Implement the method with a robust regularity penalty function as you did previously for image regularization $\Phi(s^2) = \sqrt{\epsilon + s^2}$:

$$E(u) = \int_\Omega \Phi_D((\nabla I^\top u + I_t)^2) + \lambda \Phi_R(|\nabla u|^2) \quad \mathrm{d}\mathbf{x}$$

Different to the image regularization method you implemented in the course, here also the data term has a penalty function Φ_D that yields better robustness against noisy outliers.

2.1.3 Warping

Implement the method with the course-to-fine warping approach as described in the paper above.
Warping is somewhat similar to the fixed point iteration scheme you use for the last subsection. If you have a nonlinear diffusivity, you compute an approximate solution of an equation system with a fixed diffusivity, and then update the diffusivity with the

computed solution. Warping means having an approximate solution u^k, and splitting the desired u into the known part u^k and the unknown increment du^k:

$$\begin{aligned}
E(u) &= \int_\Omega \Phi_D((I_2(\mathbf{x}+u(\mathbf{x}))-I_1(\mathbf{x}))^2)+\lambda\Phi_R(|\nabla u|)^2 \quad \mathrm{d}\mathbf{x} \qquad \Rightarrow \\
E(du^k) &= \int_\Omega \Phi_D((I_2(\mathbf{x}+u^k(\mathbf{x})+du^k(\mathbf{x}))-I_1(\mathbf{x}))^2)+\lambda\Phi_R(|\nabla u^k+\nabla du^k|^2) \quad \mathrm{d}\mathbf{x} \\
&\approx \int_\Omega \Phi_D((\nabla {I^k}^T du^k+\underbrace{I_2(\mathbf{x}+u^k(\mathbf{x}))-I_1(\mathbf{x})}_{I_t^k})^2)+\lambda\Phi_R(|\nabla u^k+\nabla du^k|^2) \quad \mathrm{d}\mathbf{x} \\
&= \int_\Omega \Phi_D((\nabla {I^k}^T du^k+I_t^k)^2)+\lambda\Phi_R(|\nabla u^k+\nabla du^k|^2) \quad \mathrm{d}\mathbf{x}
\end{aligned}$$

Practically, this means you create a warped image $I^k_{2\text{warped}}$, by looking it up in I_2 with the computed flow u^k:

$$I^k_{2\text{warped}}(\mathbf{x}) := I_2(\mathbf{x}+u^k(\mathbf{x}))$$

From this image, you create the new image gradient ∇I^k and temporal derivative I_t^k. If the warped second image matches the first image perfectly, the resulting incremental flow du^k will be zero, since the already computed flow u^k is perfect and $I_t^k \equiv 0$.
You proceed just as in equation (2), with the difference of having an additional right-hand-side expression $\lambda\Delta u^k$ in the Euler-Lagrange-Equations:

$$\begin{aligned}
0 &= \Phi'_D\cdot({I_x^k}^2 du_1^k + I_x^k I_y^k du_2^k + I_x^k I_t^k) - \lambda\mathrm{div}(\Phi'_R\cdot\nabla u_1^k) - \lambda\mathrm{div}(\Phi'_R\cdot\nabla du_1^k) \\
0 &= \Phi'_D\cdot(I_x^k I_y^k du_1^k + {I_y^k}^2 du_2^k + I_y^k I_t^k) - \lambda\mathrm{div}(\Phi'_R\cdot\nabla u_2^k) - \lambda\mathrm{div}(\Phi'_R\cdot\nabla du_2^k)
\end{aligned}$$

These are solved for du^k, and u^k is regarded constant. After each warping iteration, the increment is added to the known flow:

$$u^{k+1} := u^k + du^k$$

Although the warping strategy might improve the flow results on one image resolution alone, the really interesting fact is, that you can start with very small versions of the image, and compute the coarse, far-reaching flow on those scales, and then upsample the flow, and compute the incremental flow on the next finer scale. This way, you are able to compute the flow between two images that spans over several pixels and naturally violates the assumption of a small motion required for linearization as in (1).

2.2 Superresolution

2.2.1 General Variational Superresolution Model

Consider an Image I, that has been degraded by a linear degrading operator F

$$I_D = F \cdot I$$

F is singular and eliminates certain parts of the image.
If we have several observations of degraded images, that have been transformed by other linear operators T^i, $i \in \{1, ..., n\}$, we get a large system of linear equations consisting of systems of linear equations for every i

$$\begin{aligned} I_F^1 &= FT^1 I \\ I_F^2 &= FT^2 I \\ &\vdots \\ I_F^n &= FT^n I \end{aligned}$$

For every i, the system of linear equations is under-determined, because F is singular. The combined linear system of equations is in general overdetermined (due to noise, for example) and it can still be under-determined. Therefore, similar to the optical flow approach above, we penalize the deviations of a perfect solution to get rid of the overdetermined part of the system and introduce a regularity term for the under-determined part of the system.

$$E(I) = \sum_{i=1}^{n} \mu \left\| FT^i I - I_F^i \right\|_1 + \lambda \left\| \nabla I \right\|_1 \tag{3}$$

Up to now, we have expressed the L_1-Norm of a function by means of sublinear penalty functions:

$$\|I\|_1 = \int_\Omega |I(x)| \, \mathrm{d}x = \int_\Omega \Phi((I(x))^2) \mathrm{d}x \qquad \Phi(s) = \sqrt{s}$$

Now, we express it by its convex biconjugate (`http://en.wikipedia.org/wiki/Convex_conjugate`)

$$\|I\|_1 = \int_\Omega |I(x)| \, \mathrm{d}x = \sup_{\xi, \|\xi\|_\infty \leq 1} \int_\Omega \langle I(x), \xi(x) \rangle \, \mathrm{d}x$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. For scalar-valued images, this is a simple multiplication. The reason for this replacement is to remove the discontinuity of the L_1-Norm which makes the optimization more difficult. This also introduces a new variable ξ which is called a *dual variable*. In the following, we write $\langle I, \xi \rangle$ for $\int_\Omega \langle I(x), \xi(x) \rangle \, \mathrm{d}x$.

Equation (3) now reads as

$$E(I) = \sum_{i=1}^{n} \sup_{q^i, \|q^i\|_\infty \leq 1} \mu \left\langle FT^i I - I_F^i, q^i \right\rangle + \sup_{\xi, \|\xi\|_\infty \leq 1} \lambda \left\langle \nabla I, \xi \right\rangle \tag{4}$$

Note that we can push the scalar factors μ and λ into the definition of the solution set for the dual variables:

$$E(I) = \sum_{i=1}^{n} \sup_{q^i, \|q^i\|_\infty \leq \mu} \left\langle FT^i I - I_F^i, q^i \right\rangle + \sup_{\xi, \|\xi\|_\infty \leq \lambda} \langle \nabla I, \xi \rangle \tag{5}$$

For arbitrary operators such as F however, one can also draw μ in the operator. In the CPU sample solution, we implemented a mixture of both versions, allowing you to draw a portion inside the operator, and a portion in the limit of the dual variables. In the rest of these instructions, we will stick to the later one.

2.2.2 Huber Norm

In addition, we use the Huber Norm

$$\begin{aligned} \|I\|_\epsilon &= \int_\Omega |I(x)|_\epsilon \, \mathrm{d}x \\ |I(x)|_\epsilon &= \begin{cases} \frac{|I(x)|^2}{2\epsilon} & \text{if } |I(x)| \leq \epsilon \\ |I(x)| - \frac{\epsilon}{2} & \text{if } |I(x)| > \epsilon \end{cases} \end{aligned}$$

Note that as the Huber-Parameter ϵ approaches 0, the Huber-norm approaches the L_1-norm. We mention this here and include this in the code to stay consistent with the paper, but in the CPU sample solution, ϵ is actually set to 0. The energy now reads as

$$E(I) = \sum_{i=1}^{n} \mu \left\| FT^i I - I_F^i \right\|_\epsilon + \lambda \|\nabla I\|_\epsilon \tag{6}$$

and the biconjugate formulation (you can check that, if you want)

$$E(I) = \sum_{i=1}^{n} \sup_{q^i, \|q^i\|_\infty \leq \mu} \left\langle FT^i I - I_F^i, q^i \right\rangle - \frac{\epsilon}{2\mu} \left\| q^i \right\|^2 + \sup_{\xi, \|\xi\|_\infty \leq \lambda} \langle \nabla I, \xi \rangle - \frac{\epsilon}{2\lambda} \|\xi\|^2 \tag{7}$$

2.2.3 Optimization

This problem is now a *saddle point problem.* The problem is convex in the *primal variable* I and concave in the dual variables q and ξ. It can be optimized by doing an alternating form of gradient descent in I and an ascent in q and ξ.
For this, note that we need the *adjoint* operators of F, T and ∇. The adjoint operator of the gradient ∇ is the negative divergence $-\mathrm{div}$. The update steps look the following way:

Ascent update of the dual variables for the norm of the difference to the degraded images:

$$\begin{aligned}
{q^i}^{k+1/3} &= {q^i}^k + \tau_d(FT^i\bar{I}^k - I_D^i) \\
{q^i}^{k+2/3} &= \underset{p}{\operatorname{argmin}} \left\{ \frac{\left\| p - {q^i}^{k+1/3} \right\|^2}{2\tau_d} + \frac{\epsilon}{2\mu} |p|^2 \right\} \\
&= \frac{{q^i}^k + \tau_d(FT^i\bar{I}^k - {I_D^i}^k)}{1 + \frac{\epsilon\tau_d}{\mu}} \qquad \text{(feel free to check this)} \\
{q^i}^{k+1} &= \frac{{q^i}^{k+2/3}}{\max\left\{1, \frac{\left|{q^i}^{k+2/3}\right|}{\mu}\right\}} \qquad \text{(clipping to the feasible set)}
\end{aligned} \tag{8}$$

Ascent update of the dual variables for the norm for the regularity term:

$$\begin{aligned}
\xi^{k+1/3} &= \xi^k + \tau_d \nabla \bar{I}^k \\
\xi^{k+2/3} &= \underset{p}{\operatorname{argmin}} \left\{ \frac{\left\| p - \xi^{k+1/3} \right\|^2}{2\tau_d} + \frac{\epsilon}{2\lambda} |p|^2 \right\} \\
&= \frac{\xi^k + \tau_d \nabla \bar{I}^k}{1 + \frac{\epsilon\tau_d}{\lambda}} \\
\xi^{k+1} &= \frac{\xi^{k+2/3}}{\max\left\{1, \frac{\left|\xi^{k+2/3}\right|}{\lambda}\right\}} \qquad \text{(clipping to the feasible set)}
\end{aligned} \tag{9}$$

Descent update of the primal variable:

$$I^{k+1} = I^k - \tau_p \left(-\operatorname{div} + \sum_{i=1}^{n} {T^i}^\top F^\top {q^i}^{k+1} \right) \tag{10}$$

Over-relaxation of the primal variable for the next update of the dual variables:

$$\bar{I}^{k+1} = \omega I^{k+1} + (1-\omega) I^k, \qquad \omega \in [1, 2] \tag{11}$$

This optimization scheme has the free parameters τ_p, the time-step size of the primal update, τ_d, the time-step size of the dual update, and ω, the over-relaxation parameter. Use the ones provided in the CPU version. Pay attention to the fact that the dual updates only rely on the over-relaxed version $\bar{I}$ of I.

Though this scheme might seem complicated at first glance, it is quite straight-forward if you have a CPU reference implementation.
The remaining question is, of course, what the degrading operator F and the transformation operators T^i look like, and how we implement their adjoint (transposed) versions for equation (10).

2.2.4 The Superresolution Model you are supposed to implement on the GPU

We want you to implement the superresolution model of Unger et al. (the paper is in the repository) on the GPU. In this case, the Transformation operators T^i simply describe the backward warping performed by a given optical flow field. This means, if the flow u describes the flow from image I^i to image I_j, i.e.

$$I_j(\mathbf{x} + u(\mathbf{x})) = I_i(\mathbf{x})$$

this yields a warping operator W_{ij}:

$$(W_{ij} I_j) = I_i$$

The adjoint warping operator is a bit tricky. It is basically the bilinear backward warping reversed (in the paper, the authors use a bicubic interpolation, but we use bilinear interpolation). In the backward warping, at a pixel position $\mathbf{x}$ you add the flow at that position and get $(\mathbf{x}+u(\mathbf{x}))$, bilinearly interpolate the image value $I_2(\mathbf{x}+u(\mathbf{x}))$ and write it back to I_w at $\mathbf{x}$. For the the adjoint warping operator $W_{ij}^\top$ you perform a forward warping. This means, you take the image value at $I_1(\mathbf{x})$ and distribute that value in $I_2(\mathbf{x} + u(\mathbf{x}))$ to the four adjacent pixels belonging to the subpixel position $(\mathbf{x} + u(\mathbf{x}))$ with their corresponding coefficients according to bilinear interpolation. You accumulate all those values in the warped image. On the GPU, you have to figure out how to avoid race conditions in this somewhat random incrementation of image values.
The degrading operator F consists in our iteration of first a Gaussian blur operator B and a subsequent downsampling operator D.
Concerning the blur operator, we want a different handling of the image boundaries in this case. We want to mirror the values outside the image boundary to values inside the image. To give an example, if we have to access the image value $I(-2, y)$ somewhere in the convolution, we want to map it to the position $I(1, y)$, and on the other side, $I(nx+1, y)$ maps to $I(nx-2, y)$ on the other side. This way, the convolution preserves the average gray value of the image. A simple clamping of the image values to the boundaries does not preserve the average gray value. Mirroring the boundaries also makes the blur operator self-adjoint, i.e. $B^\top = B$.
The downsampling operator D is not really a downsampling, because you do not take samples of a function, but rather do a weighted average of the image. For the specifies we refer to the paper.
The energy now reads as

$$E(I^i) = \sum_{j \in N_i} \sup_{q^j, \|q^j\|_\infty \leq \mu} \left\langle DBW_{ji} I^i - I_F^j, q^j \right\rangle - \frac{\epsilon}{2\mu} \left\| q^j \right\|^2 + \sup_{\xi, \|\xi\|_\infty \leq \lambda} \left\langle \nabla I^i, \xi \right\rangle - \frac{\epsilon}{2\lambda} \|\xi\|^2 \tag{12}$$

This means the following:
You have been given a sequence of degraded images $I_F^i, i \in \{1, ..., n\}$, and for every image I^i you also have the flows from other images to that image on a higher resolution level (taken from trivially upsampled images) in a certain neighborhood of that image,

lets say, 8 images before and after image No i in the sequence. You assume, that the degraded image I_F^j has been produced by backward warping a higher resolution image I^i with the flow from j to i and afterwards blurring and downsampling that image ($I_F^j = DBW_{ji}I^i$).
The regularity term is usually weighted very weakly weighted (since what you actually want to recover are the high-frequency irregularities of the image), and its main purpose is to make the problem well-posed.

3 Practical Details

3.1 Framework

We provide you with a framework with a complete CPU version of both the optical flow computation and the superresolution computation. We documented the interface but not the computation code. In our sample solution, the CPU and GPU versions produce the same results. You might deviate from this a bit, for example, if you actually use the linear interpolation hardware from GPU textures, the floating point accuracy might differ from the CPU. However, as you can see, we also perform Red-Black-SOR on the CPU, which is somewhat counterintuitive, but ensures results equal to the ones on the GPU.

You have to implement the GPU functions corresponding to the linear operators in the superresolution formulation, the superresolution updates, and the optical flow updates. You also need the resampling and blurring operator for optical flow computation, so we suggest you first focus on those. Second, focus on the optical flow computation, since you need the results of the optical flow method as input for the superresolution method. We provided a main file for testing the optical flow and another one for testing the superresolution against the CPU versions.

3.2 Discretization

3.2.1 Spatial Derivatives

In the optical flow part, we use the SOR method with central differences, while for the superresolution we use forward differences for the image gradients and backward differences for the divergence of the dual variable ξ. To produce correct adjoint operators (∇ and $-$div) with a vanishing image gradient at the boundary, the dual variable xi has to be 0 outside the image. You can easily verify this for yourself in the one-dimensional case, i.e.

$$\langle \nabla I, \xi \rangle = \int_\Omega \frac{\partial}{\partial x} I(x) \cdot \xi(x) \mathrm{d}x \approx \sum_{x=1}^{N} (I(x+1) - I(x)) \cdot \xi(x) = \sum_{x=1}^{N} I(x) \cdot -(\xi(x) - \xi(x-1))$$

with $\xi(0) = 0$ and $I(N+1) = I(N)$. This is also closely related to integration by parts theorem.

3.2.2 Temporal Derivatives

For temporal derivatives in the optical flow part, use forward differences (which is somewhat intuitive, given the fact we are only computing the flow from one image to the next). This means

$$I_t(x, y) \approx I_2(x, y) - I_1(x, y)$$

4 Organizational Details

You can work in groups of up to 3 people. The workload and the general understanding of the project should be equally distributed within the team.
If you have questions, ask us. If you can formulate your questions in an email, please do so. From our experience, this allows a more precise question formulation and it also allows us to answer them in a more flexible manner.

4.1 Final Presentation

Please prepare a 30min presentation of your work. The presentation should cover the basic method, where you emphasize on how various GPU programming techniques aided you in your implementation. Each one of the team should talk approximately the same amount of time.

In short, your representation should:

- explain the main ideas of optical flow and super-resolution estimation.
- emphasize on how various GPU programming techniques have been applied in the implementation and why.
- show and explain experimental results.
- (optional:) you may also explain difficulties you experienced during the implementation
- a live demo of your project (optical flow and super-resolution)

After your presentation, we will ask you some questions about your work.

You should be familiar with the CUDA hierarchical memory model and thread organization. You should have a good understanding of the both energies and able to explain the meaning of each term. We will not ask about the theoretical part of the optimization or any derivations. However, we might ask you about the practical part or its implementation.

4.2 Assessment

- 30% basic implementations: We will thoroughly evaluate your programming assignments of the first week.
- 50% project: That will be based on how well your program works.
- 20% project presentation/questions: We assess the style of your presentation, slides and what of your work you chose to present. Further, we assess your answers on questions about the project and CUDA related topics.