Fast and Exact Solution of Total Variation Models on the GPU

Thomas Pock[1,2], Markus Unger[1]
[1]Institute for Computer Graphics and Vision, Graz University of Technology
{pock, unger, bischof}@icg.tugraz.at

Daniel Cremers[2] and Horst Bischof[1]
[2]Department of Computer Sience, University of Bonn
{pock, dcremers}@cs.uni-bonn.de

Abstract

This paper discusses fast and accurate methods to solve Total Variation (TV) models on the graphics processing unit (GPU). We review two prominent models incorporating TV regularization and present different algorithms to solve these models. We mainly concentrate on variational techniques, i.e. algorithms which aim at solving the Euler Lagrange equations associated with the variational model. We then show that particularly these algorithms can be effectively accelerated by implementing them on parallel architectures such as GPUs. For comparison we chose a state-of-the-art method based on discrete optimization techniques. We then present the results of a rigorous performance evaluation including 2D and 3D problems. As a main result we show that the our GPU based algorithms clearly outperform discrete optimization techniques in both speed and maximum problem size.

1. Introduction

Variational methods are among the most successful methods to solve a number of inverse problems in Computer Vision. Basically, variational methods aim to minimize an energy functional which is designed to appropriately describe the behavior of a certain Computer Vision task. The variational approach provides therefore a way to implement unsupervised processes by simply looking for the minimizer of the energy functional. Minimization is usually carried out by solving the Euler Lagrange (EL) differential equation associated with the energy functional.

In particular, variational models incorporating Total Variation regularization are of great interest for a large class of Computer Vision problems due to its discontinuity preserving property. Total Variation methods were introduced for non-linear image denoising [28] but in the last years they also showed great success for a much wider class of Computer Vision problems. Examples include real-time optical flow computation [34], medical image registration [27], 3D reconstruction [20], range image fusion [35] and image segmentation [30].

Variational optimization techniques are commonly used to compute the solution of Total Variation models. Due to their iterative nature they are often considered to be slow, but we show that especially these algorithms can be effectively accelerated on streaming processors such as GPUs. This leads to high performance algorithms actually outperforming state-of-the-art discrete optimization techniques.

The structure of the paper is as follows. In Section 2 we review two prominent Total Variation models. In Section 3 we discuss different algorithms to solve these models. In section 4 we give details to its numerical implementation on the GPU. Experimental details are presented in Section 5. In the last Section we give some conclusions.

2. Total Variation models

The history of L^1 estimation procedures goes back to Galileo (1632) and Laplace (1793) and has also received a lot of attention from the robust statistics community [19]. The first who introduced Total Variation methods to Computer Vision tasks were Rudin, Osher and Fatemi (ROF) in their paper on edge preserving image denoising [28]. The model is designed to remove noise and other unwanted fine scale details, while preserving sharp discontinuities (edges). The ROF model is defined as the following variational model:

$$\min_u \left\{ \int_\Omega |\nabla u| \, d\Omega + \frac{1}{2\lambda} \int_\Omega (u-f)^2 \, d\Omega \right\} , \qquad (1)$$

where Ω is the image domain, f is the observed image function which is assumed to be corrupted by Gaussian noise, and u is the sought solution. The free parameter λ is used to control the amount of smoothing in u. The aim of the

ROF model to minimize the Total Variation of u:

$$\int_\Omega |\nabla u| \, d\Omega = \int_\Omega \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2} \, d\Omega \, . \quad (2)$$

Its main property is that it allows for sharp discontinuities in the solution while still being a convex in u [28].

Similar to the ROF model, the TV-L^1 model [2], [23], [11] is defined as the variational problem

$$\min_u \left\{ \int_\Omega |\nabla u| \, d\Omega + \lambda \int_\Omega |u - f| \, d\Omega \right\} \, . \quad (3)$$

The difference compared to the ROF model is that the squared L^2 data fidelity term has been replaced by the L^1 norm. Moreover, while the ROF model in its unconstrained formulation (1) poses a strictly convex minimization problem, the TV-L^1 model is not strictly convex. This means that in general, there is no unique global minimizer.

The TV-L^1 model also offers some desirable improvements. First, it turns out that the TV-L^1 model is more effective than the ROF model in removing impulse noise (e.g. salt and pepper noise) [23]. Second, the TV-L^1 model is contrast invariant. This means that, if u is a solution of (3) for a certain input image f, then cu is also a solution for cf for $c \in \mathbb{R}^+$. Therefore the TV-L^1 model has a strong geometrical meaning which makes it useful for scale-driven feature selection [13] and denoising of shapes [24].

3. Computing the Solution of Total Variation models

Algorithms
Explicit time marching [28], [21]
Linearization of the EL equation [32], [31], [8]
Nonlinear primal-dual method [12]
Duality based methods [12], [6], [5], [18], [7], [22]
Non-linear multigrid methods [15], [4], [29], [10], [9]
First order schemes from convex optimization [33]
Second-order cone programming [16]
Graph cut methods [14], [7], [17]

Table 1. A selected list of numerical algorithms to solve Total Variation models.

Computing the solution of Total Variation models is a challenging task. The main reason lies in the non-differentiability of the L^1 norm at zero. It is therefore not surprising that one can find many items about this topic in the literature. Tab. 1 gives a selected overview of numerical methods to solve Total Variation models. Describing all these approaches in detail is clearly beyond the scope of this paper. We rather proceed by restricting our investigations to the variational approach. We do this mainly for three reasons. First, variational methods are very general and can easily be adapted to different applications. Second, variational algorithms provide a continuous solution to the underlying optimization problem. Third, variational methods are well suited to be computed on highly parallel computer architectures such as graphics processing units (GPUs).

3.1. Computing the Solution of the ROF Model

The aim of the variational approach is to minimize an energy functional by solving its associated Euler-Lagrange (EL) differential equation. For the unconstrained ROF model the EL equation is given by

$$-\nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right) + \frac{1}{\lambda}(u - f) = 0 \, , \quad (4)$$

When looking at this equation, one can make two observations: First, due to the $\frac{1}{|\nabla u|}$ term, the equation is highly non-linear. Second, the equation is not defined for $\nabla u = \mathbf{0}$. To overcome the second limitation, a simple and commonly used approach is to replace $|\nabla u|$ by a regularized version $|\nabla u|_\varepsilon = \sqrt{|\nabla u|^2 + \varepsilon}$. However, for small ε the equation is still nearly degenerated and for larger ε the ability of the ROF model to preserve sharp discontinuities is lost.

In [32], Vogel and Oman proposed a fixed point algorithm to solve (4). The basic idea is to linearize (4) by taking the non-linear terms $\frac{1}{|\nabla u|_\varepsilon}$ from the previous iteration. Therefore, at each iteration n, their method requires to solve a sparse system of linear equations

$$-\nabla \left(\frac{\nabla u^{n+1}}{|\nabla u^n|_\varepsilon}\right) + \frac{1}{\lambda}\left(u^{n+1} - f\right) = 0 \, . \quad (5)$$

This can be done with any sparse solver (e.g. Jacobi, Gauss-Seidel, SOR). In practice, the system of linear equations needs not to be solved exactly during each iteration. A few iterations of a Jacobi or Gauss-Seidel algorithm are sufficient to achieve a reasonable convergence of the entire algorithm. One serious limitation of this method is still the choice of the regularization parameter ε. Again, for small ε the algorithm becomes slow in flat regions and for large ε edges get blurred. We will refer to this algorithm in the following as *ROF-primal*.

Chan et al. in [12], Carter et al. in [5] and Chambolle in [6] studied the dual formulation of the ROF model. All three approaches exploit the dual formulation of the TV norm:

$$|\nabla u| = \max_{\mathbf{p}} \{\mathbf{p} \cdot \nabla u \, : \, \|\mathbf{p}\| \leq 1\} \, . \quad (6)$$

By substituting this expression into the ROF model (1) one arrives at the so-called primal-dual formulation of the ROF model

$$\min_u \max_{\|\mathbf{p}\| \leq 1} \left\{ \int_\Omega \mathbf{p} \cdot \nabla u \, d\Omega + \frac{1}{2\lambda} \int_\Omega (u - f)^2 \, d\Omega \right\} \, . \quad (7)$$

Since this expression is convex, we can interchange the min and the max. Furthermore, the optimality condition with respect to u is readily given by

$$u = f + \lambda \nabla \cdot \mathbf{p} \; . \tag{8}$$

Using this relation, the primal variable u can be eliminated and one arrives at the dual ROF model

$$\min_{\|\mathbf{p}\| \leq 1} \left\{ - \int_\Omega \mathbf{p} \cdot \nabla f \, d\Omega + \frac{\lambda}{2} \int_\Omega \left(\nabla \cdot \mathbf{p} \right)^2 d\Omega \right\} \; . \tag{9}$$

Note that in order to be consistent with the other approaches, we have turned the original maximization problem with respect to the dual variable (see (7)) into a minimization problem. We do this by flipping the sign of the entire functional. The Euler-Lagrange equation of (9) is given by

$$-\nabla \left(f + \lambda \nabla \cdot \mathbf{p} \right) = 0 \; , \quad \|\mathbf{p}\| \leq 1 \; . \tag{10}$$

The major advantage of the dual ROF model is that it is continuously differentiable and therefore does not suffer from the problem of the primal model which gets degenerated if $\nabla u = \mathbf{0}$. On the other hand, the dual problem has the constraint that $\|\mathbf{p}\| \leq 1$, which requires sophisticated optimization techniques. In [5], Carter studied several algorithms (interior-point primal-dual method with three relaxation methods: dual, hybrid, and barrier) to solve the dual ROF model. However, the algorithms are not very useful for practical problems due to a heavy runtimes and dependence on additional parameters.

It is somehow astonishing that there exists a very basic algorithm which has not been considered by Carter in [5]. In fact, a simple but efficient algorithm is obtained by a straightforward gradient descent and subsequent reprojection of (10).

$$\mathbf{p}^{n+1} = \frac{\mathbf{p}^n + \frac{\tau}{\lambda} \left(\nabla \left(f + \lambda \nabla \cdot \mathbf{p}^n \right) \right)}{\max(1, \left| \mathbf{p}^n + \frac{\tau}{\lambda} \left(\nabla \left(f + \lambda \nabla \cdot \mathbf{p}^n \right) \right) \right|}. \tag{11}$$

This algorithm has been proposed by Chambolle in [7] as a variant of a more comprehensive algorithm [6]. In practice, convergence is achieved as long as $\tau \leq 1/4$. The primal variable can be recovered via $u = f + \lambda \nabla \cdot \mathbf{p}$. Besides its simplicity, a further promise of this algorithm is its robustness an a fast convergence. We will refer to this algorithm in the following as *ROF-dual*.

Very recently, Aujol [1] established connections between the projected gradient descend algorithm [7], and a class of more general algorithms proposed almost 30 years ago in [3].

3.2. Computing the Solution of the TV-L^1 Model

Unfortunately, the TV-L^1 model (3) makes use of two L^1 norms, one for the TV term and one for data term. Therefore the TV-L^1 model is not strictly convex meaning that many solutions may exist. This gives us raise to the assumption that the TV-L^1 model is even more difficult to solve than the ROF model. Let us take a look at the Euler-Lagrange equation of the TV-L^1 model:

$$-\nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) + \lambda \frac{(u - f)}{|u - f|} = 0 \; . \tag{12}$$

We can easily see that this equation is degenerated either if $\nabla u = \mathbf{0}$ or $u - f = 0$. Therefore a first idea is to apply the same trick we used to regularize the Euler-Lagrange equation of the ROF model. Doing so, we simply replace $|\nabla u|$ by $|\nabla u|_\varepsilon = \sqrt{|\nabla u| + \varepsilon}$ and $|u - f|$ by $|u - f|_\delta = \sqrt{|u - f|^2 + \delta}$.

To find a convergent algorithm to solve the regularized Euler-Lagrange equation, we follow the approach of Vogel and Oman. For each iteration $n + 1$ we have to solve the following sparse system of linear equations:

$$-\nabla \cdot \left(\frac{\nabla u^{n+1}}{|\nabla u^n|_\varepsilon} \right) + \lambda \frac{(u^{n+1} - f)}{|u^n - f|_\delta} = 0 \; , \tag{13}$$

where the non-linear terms have been taken from the previous iteration n. We note that the performance of this algorithm is very sensitive with respect to the particular choice of the parameters ε and δ. Basically, small values of ε and δ slow down the algorithm a lot and large values of ε and δ induce large errors with respect to the original TV-L^1 model. We will refer to this algorithm in the following as *TVL^1-primal*.

A natural question is, whether we can solve the TV-L^1 model exactly? More precisely, can we make use of the same duality principles, we used to solve the ROF model exactly? Unfortunately, it turns out that we cannot use the duality principles directly. The reason is that the TV-L^1 model is not strictly convex.

In order to make the TV-L^1 model strictly convex, Aujol et al. proposed the following convex approximation [2]:

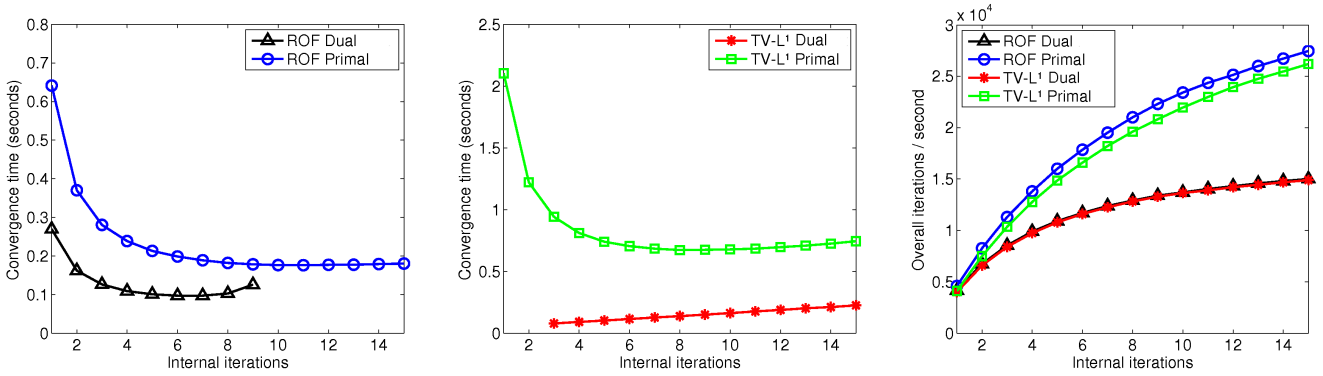
$$\min_{u,v} \left\{ \int_\Omega |\nabla u| d\Omega + \frac{1}{2\theta} \int_\Omega (u - v)^2 d\Omega + \lambda \int_\Omega |v - f| d\Omega \right\} \; . \tag{14}$$

For $\theta > 0$ (14) is a convex approximation of the TV-L^1 model and as $\theta \to 0$ (14) approaches the original TV-L^1 model (3).

Unlike the original TV-L^1 model (14) is now a optimization problem in two variables, u and v. Therefore we have to perform an alternating minimization with respect to u and v. The outline of the alternating minimization procedure is as follows:

1. For fixed v, solve (14) for u.

$$\min_u \left\{ \int_\Omega |\nabla u| \, d\Omega + \frac{1}{2\theta} \int_\Omega (u - v)^2 d\Omega \right\} \; . \tag{15}$$



(a) (b) (c)

Figure 1. (a, b) Convergence time for the ROF and TV-L^1 algorithms in dependence of number of internal iterations. (c) Overall iterations per second in depenence of number of internal iterations.

This optimization problem is exactly the ROF model, with θ being the regularization parameter. We can use the projected gradient descend algorithm to solve this sub-problem.

2. For fixed u, solve (14) for v.

$$\min_v \left\{ \frac{1}{2\theta} \int_\Omega (u-v)^2 d\Omega + \lambda \int_\Omega |v-f| d\Omega \right\} . \quad (16)$$

(16) is a point-wise convex minimization problem which can be solved via the following soft-thresholding scheme:

$$v = \begin{cases} u - \lambda\theta & \text{if} \quad u - f > \lambda\theta \\ u + \lambda\theta & \text{if} \quad u - f < -\lambda\theta \\ f & \text{if} \quad |u-f| \leq \lambda\theta \end{cases} \quad (17)$$

3. Goto 1. until convergence.

We note that the algorithm is very robust with respect to the choice of the approximation parameter θ. We found that a good choice is to set θ such that the soft-threshold $\lambda\theta$ is $1-5\%$ of the maximum gray-level interval of f. We will refer to this algorithm in the following as *TVL^1-dual*.

4. Implementation on the GPU using CUDA

We implemented the following variational variational algorithms: *ROF-primal*, *ROF-dual*, *TVL^1-primal* and *TVL^1-dual* on the graphics card. The implementation is basically a straight-forward implementation of the presented schemes. We used Jacobi's method to solve the systems of linear equations in the primal methods. Our implementations can handle 2D and 3D problems.

With the introduction of the 8-series [25], NVidia also introduced the CUDA (Compute Unified Device Architecture) framework [26]. CUDA provides a standard C language interface for programming on the GPU. It can handle a massive number of parallel threads that are scheduled to the processor. CUDA also provides the user with a programming interface that handles scheduling and execution on the GPU.

The processing units of the GPU are arranged into groups of so-called multiprocessors. One multiprocessor, can execute several independent threads having access to the same shared memory. While reading data from the global GPU memory is still fast (about $50-100$ GB/s), reading from the shared memory is even 75 times faster. We exploited this feature by loading a local image patch into the shared memory and ran the algorithms for several iterations before writing the results back into global memory. High speedups can be gained using the shared memory. On the other hand, the number of such internal iterations should also be limited. The information at block borders cannot be exchanged during computation leading to a slower convergence of the entire algorithm. The effects of internal iterations versus the convergence behavior the algorithms are illustrated in Fig. 1 shows the performance of our algorithms depending on the number of internal iterations. We found that using 5 internal iterations gives the best overall performance.

5. Experimental Results

For evaluation we used a standard personal computer equipped with a 2.13 GHz Core2-Duo CPU, 2 GB of main memory and a NVidia 8800GTX graphics card. The computer runs a 32-bit Linux operating system. Basically, with our GPU based implementation we achieved a speedup factor of approximately 1000 compared to an optimized Matlab implementation. However, in this paper we do not compare our GPU-based algorithms to CPU-based variants, because this question is of minor interest. The more interesting question is weather GPU-accelerated Total Variation

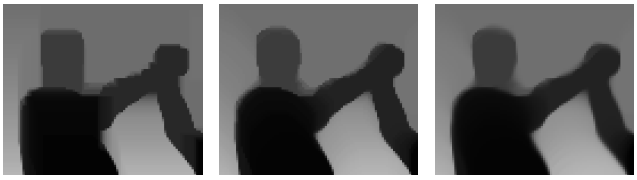
(a) (b) (c) (d) (e)

Figure 2. Test images: (a) *Summit* image: 256×256. (b) *Basecamp* image: 512×512. (c) *Sunset* image: 1024×1024. (d) *Brain* data set: $256 \times 320 \times 256$. (e) *Liver* data set: $512 \times 512 \times 128$

methods can compete with discrete optimization techniques such as graph cuts. Note that TV models and graph cuts can solve problems of equal complexity. So far, it is not clear which method will be the clear winner. Maybe, the work presented in this paper can shed further light on this question. We therefore decided to compare our algorithms to a recently published method based on parametric max-flow algorithms [17]. The algorithm of [17] was downloaded from Wotao Yin's homepage. It was executed on the same machine as the GPU-based algorithms using Matlab 7.0.3. Note that the core of this algorithm is based on a high performance C/C++ implementation.

Unfortunately, there exist no sharp L^2, or even better, L^∞ error bound for variational methods. A common method is to compute the residuum norm of the Euler-Lagrange equations and to stop the iterations when the residuum norm is below a certain threshold. On the other hand, the residual norm does not give any information about the error of the solution with respect to the true solution. Note that graph-cut based algorithms come along with such an error bound [7], [17]. This is can be seen as a clear advantage of graph cut methods over variational methods. We applied the following procedure to estimate the L^2 error bound. We first run our algorithms for a long time to produce a ground truth. We can then measure the time the algorithms need to fall below a predefined L^2 error bound.

On the other hand, graph cut methods suffer from the so-called metrication error. In contrast to variational methods, one cannot use the Euclidean vector norm to approximate the TV norm. Instead, one has to rely on a weighted L^1 vector norm based on connected graph nodes. Fig. 3 points out the metrication error of graph cut methods. When using a 4-connected graph the results become very blocky. When using a 16-connected graph, the results are very close to the results of the variational methods. We therefore decided to compare our algorithms to the max-flow algorithm based on a 16-connected graph. Note that both the required memory and the computing time increase for a higher connectivity in the graph-structure.



(a) (b) (c)

Figure 3. Metrication error of the max-flow algorithm compared to the variational methods. (a) Blocky structures when using a 4-connected graph. (b) 16-connected graph. (c) Our algorithms.

5.1. Test Data

Fig. 2 depict our 2D and 3D test images. The 2D images are of 8-bit and are rescaled in order to have pixel intensities lying in the interval $[0, 1]$. We used a L^2-error threshold of $1e^{-03}$ to determine the runtimes. The max-flow algorithms were computed using 8-bit accuracy. Note that our L^2-error is equivalent to a 10-bit accuracy. The *Brain* data set is also of 8-bit accuracy, so that we used a L^2-error threshold of $1e^{-03}$. The *Liver* data set is of 16-bit accuracy, so we adjusted the L^2-error threshold to be $1e^{-04}$.

5.2. Numerical Results for the 2D ROF model

The parameters for the evaluation of the algorithms to solve the ROF model were as follows: For *ROF-primal*, the ε parameter was set to $1e^{-04}$. Clearly, larger values of ε would speed up the convergence of *ROF-primal*. But for $\varepsilon > 1e^{-04}$ *ROF-primal* did not converge to a solution within the L^2 error bound with respect to the exact solution of the duality based methods. *ROF-dual* does not depend on any additional free parameters.

Tab. 2 shows the runtimes needed to compute the solution of the ROF model for different values of λ, and for different image sizes. In general, as λ increases, all algorithms become slower since the regularization becomes stronger and makes the problem more difficult to solve. For small values of λ one can see that the variational methods are up to 1000 times faster than *ROF-max-flow*. For larges value of λ the relative performance decreases but the speedup is still

Image	*Basecamp*						*Summit*	*Basecamp*	*Sunset*
λ	0.01	0.05	0.10	0.20	0.50	1.00	0.20		
ROF-primal (GPU)	0.0011	0.0859	0.3161	0.8661	2.9414	8.3621	0.2825	0.8661	2.4000
ROF-dual (GPU)	0.0013	0.0054	0.0213	0.0596	0.3312	1.4667	0.0175	0.0596	0.5041
ROF-max-flow (CPU)	1.4665	2.5433	3.6647	5.1440	8.3739	12.449	1.0169	5.1440	25.072

Table 2. Runtimes (in seconds) to solve the ROF model for different values of λ, and for different sizes.

more than a factor of 10. The evaluation on different image sizes was done with fixed $\lambda = 0.2$. We can see that *ROF-primal* and *ROF-dual* have superior performance compared to *ROF-max-flow*. The runtime of *ROF-max-flow* increases slightly faster than linear, which is also reported in [17]. In contrast, the runtimes of *ROF-primal* and *ROF-dual* increase slightly slower than linear. Again, *ROF-dual* is the fastest algorithm.

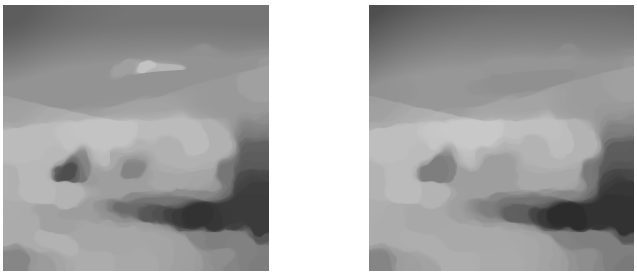
5.3. Numerical Results for the 2D TV-L^1 model

We used the following parameters for the evaluation of the TV-L^1 algorithms. For TVL^1*-primal*, the ε and δ parameters were both set to $1e^{-03}$. We also tried smaller values but it took extremely long to fall beyond the L^2-error threshold of $1e^{-03}$. For TVL^1*-dual* the θ parameter was dynamically adjusted such that the soft threshold $\lambda\theta$ remains constantly $\lambda\theta = 0.01$ for all values of λ (see also (17)).

Tab. 3 shows the runtimes needed to compute the solution of the TV-L^1 model for different values of λ and different image sizes. TVL^1*-dual* performs extremely well compared to TVL^1*-primal* and TVL^1*-max-flow*. On the other hand, the advance of TVL^1*-primal* over TVL^1*-max-flow* is not very high since the primal Euler-Lagrange equation of the TV-L^1 model is very difficult to solve. One can increase ε and δ parameters but this leads to wrong results. As TVL^1*-primal*, TVL^1*-dual* is also a convexification of the original TV-L^1 model, but in a very different way. It does only depend upon one additional parameter which can be automatically chosen.

Also note that the relative performance of TVL^1*-max-flow* over the variational TV-L^1 methods is much better than the relative performance of *ROF-max-flow* over the variational algorithms to solve the ROF model. A possible explanation could be that since the TV-L^1 problem is purely geometric, it it more suitable to be computed on graphs than the ROF model. For the comparison of different image sizes we used $\lambda = 0.5$. One can see that TVL^1*-dual* scales very well with increasing image size, whereas TVL^1*-primal* gets much slower. The runtime of TVL^1*-max-flow* also heavily increases for larger images.

Fig. 4 shows a comparison of the ROF model to the TV-L^1 model. One can clearly observe the contrast invariance of the TV-L^1 model, that is to remove structures of a certain scale. For example the snow covered mountain has a large contrast to the background but is of a small scale.



(a) ROF model:$\lambda = 1.0$ (b) TV-L^1 model:$\lambda = 0.1$

Figure 4. This figure shows an comparison of the ROF model to the TV-L^1 model. Due to the contrast invariance of the TV-L^1 model, structures of a certain scale are removed.

5.4. Numerical Results for the 3D ROF model

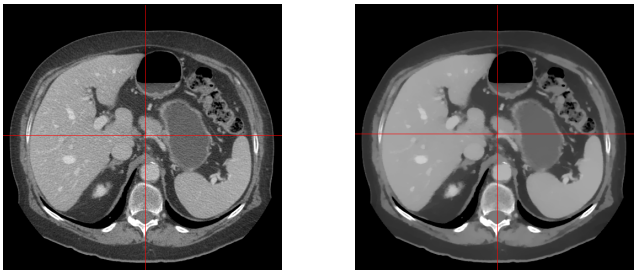
We used the following parameters for the *ROF-primal* algorithm. In case of the *Brain* data set we used $\varepsilon = 1e-04$. In case of the*Liver* data set we used $\varepsilon = 1e-05$ since the resolution of the intensity values is about 10 times higher. For the *ROF-dual* algorithm we did not have to set any additional algorithms. *ROF-max-flow* was computed using a 6-connected graph and 8-bit accuracy.

Tab. 4 shows the results of the ROF model applied to the *Brain* data set and the *Liver* data set for different values of λ. Unfortunately, *ROF-max-flow* cannot be executed for the *Liver* data set, due to a heavy memory requirement. We even tried it on a 64-bit machine equipped with 16GB of main memory, but the algorithm failed. On the other hand, using *ROF-dual*, the solution was obtained after approximately 20 seconds. This enables our method to be applicable for clinical practice even for large data sets. Fig. 5 gives an example of the denoising capability of the ROF model applied to the *Liver* data set.

One can see that *ROF-dual* has always a superior performance compared to *ROF-primal* and *ROF-max-flow*. One can also see that the relative performance of the variational methods of the graph based method decrease in case of larger values of λ. This also reflects the results we obtained from our 2D experiments. In contrast to the 2D experiments the relative performance of *ROF-primal* compared to *ROF-dual* is considerably better. In case of the primal formulation of the ROF model, the size of the optimization problem is exactly the size of the input image. In case of the dual

Image	*Basecamp*						*Summit*	*Basecamp*	*Sunset*
λ	0.10	0.20	0.30	0.50	0.70	1.00	0.50		
TVL^1-*primal* (GPU)	2.9564	1.0967	0.3588	0.3553	0.2093	0.1848	0.3401	0.3588	0.9113
TVL^1-*dual* (GPU)	0.1567	0.0636	0.0372	0.0456	0.0437	0.1610	0.0684	0.0456	0.0513
TVL^1-*max-flow* (CPU)	3.9400	2.5600	2.0000	1.5600	1.2900	1.0600	0.3200	1.5600	7.1900

Table 3. Runtimes (in seconds) to solve the TV-L^1 model for different values of λ and different image sizes.



(a) Original image (b) Denoised image

Figure 5. Denoising capability of the ROF model for the *Liver* data set using $\lambda = 0.01$.

formulation the size of the optimization problem is three times larger (two times in 2D). This leads to a higher memory bandwidth on the GPU and hence to a higher processing time. However, note that the dual algorithm is still faster.

	Brain		*Liver*
λ	0.05	0.1	0.01
ROF-primal (GPU)	13.00	60.00	51.71
ROF-dual (GPU)	1.68	3.59	18.16
ROF-max-flow (CPU)	106.00	124.00	*failed*

Table 4. Runtimes (in seconds) to solve the ROF model for the *Brain* data set and the *Liver* data set.

5.5. Numerical Results for the 3D TV-L^1 model

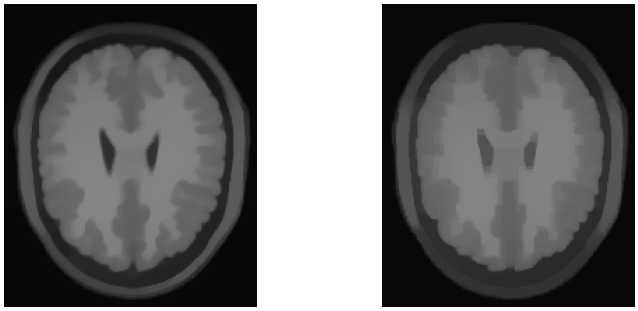
We used the following parameter setting for the TVL^1-*dual* algorithm: For the *Brain* data set we used $\lambda\theta = 0.01$ and for the *Liver* data set we used $\lambda\theta = 0.005$. For TVL^1-*primal* we used $\epsilon = \delta = 1e-03$ in case of the *Brain* data set and $\epsilon = \delta = 1e-04$ in case of the *Liver* data set. *ROF-max-flow* was computed using a 6-connected graph and 8-bit accuracy.

Tab. 5 shows the results of the TV-L^1 model applied to the *Brain* data set and the *Liver* data set for different values of λ. One can observe similar results to the 2D experiments. In any case, TVL^1-*dual* is the best performer. Note that TVL^1-*dual* is $4-9$ times faster than TVL^1-*max-flow*. As for the ROF model, TVL^1-*max-flow* could not be executed on the large *Liver* data set.

Fig. 6 shows a comparison between the results of TVL^1-*dual* and TVL^1-*max-flow* applied to the *Brain* data set. One can clearly observe the metrication error of the discrete method leading to the development of unnatural blocky structures.

	Brain		*Liver*
λ	0.5	0.2	1.0
TVL^1-*primal* (GPU)	37.10	137.00	113.41
TVL^1-*dual* (GPU)	14.40	16.60	97.63
TVL^1-*max-flow* (CPU)	50.10	142.00	*failed*

Table 5. Runtimes (in seconds) to solve the TV-L^1 model for the *Brain* data set and the *Liver* data set.



(a) TVL^1-*dual* (b) TVL^1-*max-flow*

Figure 6. Results of the TV-L^1 model applied to the *Brain* data set. The figure shows a comparison between TVL^1-*dual* and TVL^1-*max-flow* for $\lambda = 0.5$. One can clearly see the metrication error of the discrete method.

6. Conclusion

In this paper we have evaluated the different variational algorithms to solve the ROF model and the TV-L^1 model. We have compared our GPU-based implementation to a recently published method based on discrete optimization techniques. Based on our observations we can give the following conclusions: Variational methods are easy to implement and can be effectively computed on GPUs. Graph-based methods are difficult to implement and their parallelization is still an open problem. Variational methods do not suffer from metrication errors. Variational methods do not provide an explicit error-estimate, which makes it hard to define a stopping critereon. In contrast discrete optimization techniques come along with such an estimate. Finding such an estimate would be a great benefit for the variational methods.

References

[1] J.-F. Aujol. Some algorithms for total variation based image restoration. Technical report, CMLA, ENS CACHAN, CNRS, UNIVERSUD, 2008.

[2] J.-F. Aujol, G. Gilboa, T. Chan, and S. Osher. Structure-texture image decomposition–modeling, algorithms, and parameter selection. *Int. J. Comp. Vis.*, 67(1):111–136, 2006.

[3] A. Bermudez and C. Moreno. Duality methods for solving variational inequalities. *Comp. and Math. with Appls.*, 7:43–58, 1981.

[4] A. Bruhn and J. Weickert. Towards ultimate motion estimation: Combining highest accuracy with real-time performance. In *Proc. 11th Int. Conf. Comp. Vis.*, pages 749–755, 2005.

[5] J. Carter. *Dual Methods for Total Variation-based Image Restoration*. PhD thesis, UCLA, Los Angeles, CA, 2001.

[6] A. Chambolle. An algorithm for total variation minimizations and applications. *J. Math. Imaging Vis.*, 2004.

[7] A. Chambolle. Total variation minimization and a class of binary MRF models. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 136–152, 2005.

[8] A. Chambolle and P.-L. Lions. Image recovery via total variation minimization and related problems. *Numer. Math.*, 76:167–188, 1997.

[9] T. Chan and K. Chen. On a nonlinear multigrid algorithm with primal relaxation for the image total variation minimisation. *Numerical Algorithms*, 41:387–411, 2006.

[10] T. Chan, K. Chen, and J. Carter. Iterative methods for solving the dual formulation arising from image restoration. *Electronic Transactions on Numerical Analysis*, 26:299–311, 2007.

[11] T. Chan and S. Esedoglu. Aspects of total variation regularized L^1 function approximation. *SIAM J. Appl. Math.*, 65(5):1817–1837, 2004.

[12] T. Chan, G. Golub, and P. Mulet. A nonlinear primal-dual method for total variation-based image restoration. *SIAM J. Sci. Comp.*, 20(6):1964–1977, 1999.

[13] T. Chen, W. Yin, X. Zhou, D. Comaniciu, and T. Huang. Total variation models for variable lighting face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(9):1519–1524, 2006.

[14] J. Darbon and M. Sigelle. Image restoration with discrete constrained total variation, part i: fast and exact optimization. *J. Math. Imaging Vis.*, 26(3):261–276, 2006.

[15] C. Frohn-Schauf, S. Henn, and K. Witsch. Nonlinear multigrid methods for total variation image denoising. *Comput. Visual Sci.*, pages 199–206, 2004.

[16] D. Goldfarb and W. Yin. Second-order cone programming methods for total variation-based image restoration. *SIAM Journal on Scientific Computing*, 27(2):622–645, 2005.

[17] D. Goldfarb and W. Yin. Parametric maximum flow algorithms for fast total variation minimization. Technical report, Rice University, 2007.

[18] M. Hintermüller and K. Kunisch. Total bounded variation regularization as bilaterally constrained optimization problems. *SIAP*, 64(4):1311–1333, 2004.

[19] P. Huber. *Robust Statistics*. Wiley, New York, 1981.

[20] K. Kolev, M. Klodt, T. Brox, S. Esedoglu, and D. Cremers. Continuous global optimization in multiview 3D reconstruction. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 441–452, China, 2007.

[21] A. Marquina and S. Osher. Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal. *SIAM J. Sci. Comput.*, 22:387–405, 2000.

[22] M. K. Ng, L. Qi, Y.-F. Yang, and Y.-M. Huang. On semismooth Newton's method for total variation minimization. *J. Math. Imaging Vis.*, 27:265–276, 2007.

[23] M. Nikolova. A variational approach to remove outliers and impulse noise. *J. Math. Imaging Vis.*, 20(1-2):99–120, 2004.

[24] M. Nikolova, S. Esedoglu, and T. Chan. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal of Applied Mathematics*, 66(5):1632–1648, 2006.

[25] NVidia. NVidia GeForce 8800 GPU architecture overview. Technical report, NVidia, 2006.

[26] NVidia. Nvidia CUDA Compute Unified Device Architecture programming guide 1.1. Technical report, NVidia, 2007.

[27] T. Pock, M. Urschler, C. Zach, R. Beichel, and H. Bischof. A duality based algorithm for TV-L^1-optical-flow image registration. In *10th International Conference on Medical Image Computing and Computer Assisted Intervention*, pages 511–518, Brisbane, Australia, 2007.

[28] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[29] J. Savage and K. Chen. An improved and accelerated nonlinear multigrid method for total-variation denoising. *J. Math. Imaging Vis.*, 82(8):1001–1015, 2005.

[30] M. Unger, T. Pock, and H. Bischof. Continuous globally optimal image segmentation with local constraints. In *Computer Vision Winter Workshop 2008*, page accepted, 2008.

[31] C. Vogel. A multigrid method for total variation-based image denoising. *Progress in Systems and Control Theory*, 1995.

[32] C. Vogel and M. Oman. Iteration methods for total variation denoising. *SIAM J. Sci. Comp.*, 17:227–238, 1996.

[33] P. Weiss, L. Blanc-Féraud, and G. Aubert. Efficient schemes for total variation minimization under constraints in image processing. Technical report, INRIA, 2007.

[34] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L^1 optical flow. In *29th DAGM Symposium on Pattern Recognition*, pages 214–223, Heidelberg, Germany, 2007.

[35] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust TV-L^1 range image integration. In *Proc. 13th Int. Conf. Comp. Vis.*, Rio de Janeiro, Brazil, 2007.