Learning optimal discretizations of the total variation

Thomas Pock

Institute of Computer Graphics and Vision
Graz University of Technology

Mathematical Image Analysis (MIA) 2021

Joint work with A. Chambolle / Ceremade - Université Paris-Dauphine
Edges

- Edges are among the most important features in images
- Image understanding relies on abstract discontinuity information
- Most successful image descriptors are based on intensity gradients
- First layers in deep convolutional networks represent edge detectors

In (b), meaningful edges are absent yet “visible”...
Treating images $u : \Omega \rightarrow \mathbb{R}$ as continuous (differentiable) functions, edges correspond to strong image gradients:

$$\nabla u = \left( \frac{\partial x_1 u}{\partial x_2 u} \right), \quad |\nabla u|_2 = \sqrt{(\partial x_1 u)^2 + (\partial x_2 u)^2}.$$

Treating images as discrete arrays $u \in X \simeq \mathbb{R}^{M \times N}$, image gradients are approximated using a finite differences operator $D$:

$$Du = ((Du)_1, (Du)_2) \approx \left( \frac{u_{i+1,j} - u_{i,j}}{u_{i,j+1} - u_{i,j}} \right), \quad |Du|_2 \approx \sqrt{(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2}.$$

Observe that the finite differences approximations can be computed by convolving the image with small filters kernels

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>
```

black=-1, white=+1

What is the statistics of edges in natural images?
Edge statistics of natural images

- Randomly extracted $15M$ image patches of size $2 \times 2$ from a natural image data set.
- Compute finite differences in horizontal and vertical direction.
- Yields a heavy tailed distribution $\sim$ most gradients are zero $\sim$ sparse gradients.
The total variation

- A natural prior for image reconstruction is the following regularization term, called the total variation:

$$TV(u) = \|Du\|_{2,1} = \sum_{i,j} \sqrt{(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2}.$$

- Has been introduced in [Rudin, Osher, Fatemi '92], [Chambolle, Lions '97].
- Provides a reasonable match to the true image statistics.
Continuous formulation

- Making use of standard duality, the total variation (in 2D) is defined for any function $u \in L^1(\Omega)$ as [Giusti '84]

$$TV(u) = \sup \left\{ \int_\Omega u \text{div } p \, dx : p \in C^\infty_c(\Omega, \mathbb{R}^2), \ |p(x)|_2 \leq 1 \ \forall x \in \Omega \right\},$$

where $p = (p^1, p^2)$ is the dual variable.

- Allows for discontinuities in $u$.
- It is a convex lower-semicontinuous function.
- It also has a nice geometric interpretation $\rightsquigarrow$ minimal surfaces.
- For sufficiently smooth functions $u \in W^{1,1}$ integration by parts gives

$$TV(u) = \int_\Omega |\nabla u| \, dx.$$
The ROF model

The ROF model [Rudin, Osher, Fatemi '92] is defined as the following minimization problem

$$\min_{u} \lambda \|Du\|_{2,1} + \frac{1}{2} \|u - g\|^2, \lambda > 0$$

Defines "the" prototypical variational model in mathematical image processing.
Gives a good tradeoff between simplicity of the model and denoising quality.
The total variation can identify and preserve the most important edges in images and hence remove noise and artifacts.
The ROF model is equal to the proximity operator of the total variation.
Has been extended in numerous ways [Chan, Eseduglu '04], [Osher, Sole, Vese '03], [Bredies, Kunisch, P. '10], ...
Can be efficiently minimized using duality based approaches [Chambolle '04], [Weiss, Aubert, Blanc-Féraud '08], [Beck, Teboulle '08] or primal-dual algorithms [Zhu, Chan '08], [Goldstein, Osher '09], [Esser, Zhang, Chan '10], [Chambolle, P. '09-'16], ...
Total variation minimization

\[ \sum_{i,j} \sqrt{(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2} \approx TV \approx 23501.84 \]
Total variation minimization

\[ (u_{i+1,j} - u_{i,j}) \]

\[ (u_{i,j+1} - u_{i,j}) \]

\[ \sqrt{(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2} \]

\[ \sum_{i,j} \rightarrow TV \approx 7916.47 \]
Total variation minimization

\[ TV \approx 2987.83 \]
Advanced applications

(a) Denoising  
(b) Deblurring  
(c) MRI

(d) Motion  
(e) Stereo  
(f) Segmentation
For most practical problems, the standard discrete total variation gives sufficiently good results.

However, on free discontinuity problems such as image inpainting, the standard discretization yields strong artifacts.

Inpainting of straight discontinuities
For most practical problems, the standard discrete total variation gives sufficiently good results.

However, on free discontinuity problems such as image inpainting, the standard discretization yields strong artifacts.

Inpainting of straight discontinuities
Advanced free discontinuities problems

Convexification of the Mumford-Shah functional [P., Cremers, Bischof, Chambolle '09]:

Convexification of Euler's elastica [Chambolle, P. '19]

(data from J. Weickert)

Here, the discretization can make a difference between “working” and “not working”.
Finding a good general discretization of the total variation is far from being trivial and hence many approaches have been proposed:

- Non-standard finite differences for anisotropic diffusion [Weickert, Welk, Wichert '13]
- Graph-based / MRFs / crystalline energies [Boykov, Kolmogorov '03], [Chambolle '05]
- Upwind discretization [Chambolle, Levine, Lucier '11]
- Shannon TV [Abergel, Moisan '17]
- Conforming P1 finite elements [Bartels '12]
- Non-conforming P1 (Crouzeix-Raviart) finite elements [Chambolle, P. 18]
- Duality based discretization using $H(\text{div})$-conforming Raviart-Thomas (RT0) vector fields [Herrmann, Herzog, Schmidt, Vidal, Wachsmuth '18], [Caillaud, Chambolle '20]
- Approximate Raviart-Thomas [Hintermüller, Rautenberg, Hahn '14], [Condat '17]
Related work

Finding a good general discretization of the total variation is far from being trivial and hence many approaches have been proposed:

- Non-standard finite differences for anisotropic diffusion [Weickert, Welk, Wichert '13]
- Graph-based / MRFs / crystalline energies [Boykov, Kolmogorov '03], [Chambolle '05]
- Upwind discretization [Chambolle, Levine, Lucier '11]
- Shannon TV [Abergel, Moisan '17]
- Conforming P1 finite elements [Bartels '12]
- Non-conforming P1 (Crouzeix-Raviart) finite elements [Chambolle, P. 18]
- Duality based discretization using $H(\text{div})$-conforming Raviart-Thomas (RT0) vector fields [Herrmann, Herzog, Schmidt, Vidal, Wachsmuth '18], [Caillaud, Chambolle '20]
- Approximate Raviart-Thomas [Hintermüller, Rautenberg, Hahn '14], [Condat '17]

- This talk: Optimized approximate Raviart-Thomas using learning.
General setting

- We introduce the finite differences operator $\mathbf{D}u = (D^1u, D^2u)$ with

$$
\begin{align*}
(D^1u)_{i+\frac{1}{2},j} &= u_{i+1,j} - u_{i,j} & i = 1, \ldots, M - 1, j = 1, \ldots, N, \\
(D^2u)_{i,j+\frac{1}{2}} &= u_{i,j+1} - u_{i,j} & i = 1, \ldots, M, j = 1, \ldots, N - 1.
\end{align*}
$$

- The total variation is defined as

$$
TV(u) := \sup \{ \langle \mathbf{p}, \mathbf{D}u \rangle_Y : \| F\mathbf{p} \|_{Z^*} \leq 1 \}
$$

where $\mathbf{p} = (p^1, p^2)$ are the dual variables and $F = (F^1, \ldots, F^L)$ are convolutional interpolation kernels defined as

$$
(F^l\mathbf{p})_{i,j} = \left( (F^{l,1}p^1)_{i,j}, (F^{l,2}p^2)_{i,j} \right) = \left( \sum_{m,n=-\nu}^{\nu} \xi^l_{m,n} p^1_{i+\frac{1}{2}-m,j-n}, \sum_{m,n=-\nu}^{\nu} \eta^l_{m,n} p^2_{i-m,j+\frac{1}{2}-n} \right)
$$

- The primal form has the structure of a sparse coding problem

$$
TV(u) = \min_{\mathbf{q} : F^*\mathbf{q} = \mathbf{D}u} \| \mathbf{q} \|_Z,
$$

where $F^*$ can be interpreted as a convolutional dictionary.
Example: Forward differences

- Interpolation kernels (Nearest neighbor interpolation):
  \[(Fp)i,j = \begin{pmatrix} p^1_{i+\frac{1}{2},j} \\ p^2_{i,j+\frac{1}{2}} \end{pmatrix}\].

- The $Z$-norm is given by
  \[\|z\|_Z = \sum_{i,j} \sqrt{(z^1_{i+\frac{1}{2},j})^2 + (z^2_{i,j+\frac{1}{2}})^2}\].

Interpolation kernels $F$
Example: Raviart-Thomas

- Interpolation kernels (Nearest neighbor interpolation):

\[
(F^1 p)_{i-\frac{1}{2},j-\frac{1}{2}} = \begin{pmatrix} p^1_{i-\frac{1}{2},j} \\ p^2_{i,j-\frac{1}{2}} \end{pmatrix}, \quad (F^2 p)_{i-\frac{1}{2},j+\frac{1}{2}} = \begin{pmatrix} p^1_{i-\frac{1}{2},j} \\ p^2_{i,j+\frac{1}{2}} \end{pmatrix},
\]

\[
(F^3 p)_{i+\frac{1}{2},j-\frac{1}{2}} = \begin{pmatrix} p^1_{i+\frac{1}{2},j} \\ p^2_{i,j-\frac{1}{2}} \end{pmatrix}, \quad (F^4 p)_{i+\frac{1}{2},j+\frac{1}{2}} = \begin{pmatrix} p^1_{i+\frac{1}{2},j} \\ p^2_{i,j+\frac{1}{2}} \end{pmatrix}.
\]

- The $Z$-norm is given by

\[
\|(z^1, z^2, z^3, z^4)\|_Z := \sum_{i,j} |z^1_{i-\frac{1}{2},j-\frac{1}{2}}|^2 + |z^2_{i-\frac{1}{2},j+\frac{1}{2}}|^2 + |z^3_{i+\frac{1}{2},j-\frac{1}{2}}|^2 + |z^4_{i+\frac{1}{2},j+\frac{1}{2}}|^2
\]
Example: Condat’s discretization

- Interpolation kernels (bilinear interpolation):

\[
(F^1 p)_{i,j} = \left( \frac{p^1_{i-\frac{1}{2},j} + p^1_{i+\frac{1}{2},j}}{2}, \frac{p^2_{i,j-\frac{1}{2}} + p^2_{i,j+\frac{1}{2}}}{2} \right),
\]

\[
(F^2 p)_{i+\frac{1}{2},j} = \left( \frac{p^1_{i+\frac{1}{2},j}}{2}, \frac{p^2_{i,j-\frac{1}{2}} + p^2_{i,j+\frac{1}{2}} + p^2_{i+1,j-\frac{1}{2}} + p^2_{i+1,j+\frac{1}{2}}}{4} \right), \quad (F^3 p)_{i,j+\frac{1}{2}} = \left( \frac{p^1_{i-\frac{1}{2},j} + p^1_{i+\frac{1}{2},j} + p^1_{i-\frac{1}{2},j+1} + p^1_{i+\frac{1}{2},j+1}}{4}, \frac{p^2_{i,j+\frac{1}{2}}}{2} \right).
\]

- The \( Z \)-norm is given by

\[
\| (z^1, z^2, z^3) \|_Z := \sum_{i,j} |z^1_{i,j}|_2 + |z^2_{i+\frac{1}{2},j}|_2 + |z^3_{i,j+\frac{1}{2}}|_2
\]
Comparison

Input
Comparison

Forward differences

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>80.366</td>
<td>26.937</td>
<td>86.226</td>
<td>27.024</td>
<td>80.967</td>
<td>23.615</td>
<td>22.899</td>
<td>23.299</td>
<td>81.726</td>
</tr>
</tbody>
</table>
Comparison

Raviart-Thomas
Comparison

Condat
Consistency result

- We define a family of discrete total variations for pixels of size $\varepsilon \times \varepsilon$:

$$TV_\varepsilon(u) = \min \{ \varepsilon^2 \|q\|_{Z_\varepsilon} : F^*_\varepsilon q = D_\varepsilon u \} = \sup \{ \varepsilon^2 \langle p, D_\varepsilon u \rangle_{Y_\varepsilon} : \|F_\varepsilon p\|^*_{Z} \leq 1 \}$$

Theorem

Assume the supports and the weights of the convolutions defining $F_\varepsilon$ are uniformly bounded that is

$$\sum_{m,n} \xi^l_{m,n} = \sum_{m,n} \eta^l_{m,n} = 1 \iff F^{l,1}, F^{l,2} \in C_{\Sigma=1}.$$  

Then $TV_\varepsilon$ $\Gamma$-converges to

$$TV(u) := \begin{cases} |Du|(\Omega) & \text{if } u \in BV(\Omega), \\ +\infty & \text{else}. \end{cases}$$

As long as the filter coefficients sum up to one and are uniformly bounded, we are having a consistent discretization of the total variation $\rightsquigarrow$ learning.
We consider the following class of total variation minimization problems

\[
\min_{Du=F^*q} \lambda \|q\|_Z + G(u, g),
\]

with a saddle-point formulation

\[
\min_{u,q} \max_p \langle Du - F^*q, p \rangle + \lambda \|q\|_Z + G(u, g)
\]

Can be applied to a large class of inverse problems in imaging such as denoising, inpainting, segmentation, ....

We need access to the proximal maps of \( \|\cdot\|_Z \) and \( G(\cdot, g) \).
Primal-dual algorithm

Preconditioned primal-dual algorithm

- **Initialization**: \( u^0 \in X, \ q^0 \in Z, \ p^0 \in Y \).
- **Step sizes**: Choose the block-wise step sizes \( \tau_u, \tau_q, \sigma_p \) such that

\[
\left\| \text{diag}(\sigma_p)^{\frac{1}{2}} (D, F^*) \text{diag}(\tau_u, \tau_q)^{\frac{1}{2}} \right\| \leq 1,
\]

and set \( \theta = 1 \).

- **Iterations**: For \( k = 0, \ldots, K - 1 \) let

\[
\begin{align*}
  p^{k+1} &= p^k + \sigma_p (D u^k - F^* q^k) \\
  \bar{p}^{k+1} &= p^{k+1} + \theta (p^{k+1} - p^k) \\
  u^{k+1} &= \text{prox}_{\tau_u G}(u^k - \tau_u D^* \bar{p}^{k+1}) \\
  q^{k+1} &= \text{shrink}_{\tau_q \lambda}(q^k - \tau_q F \bar{p}^{k+1})
\end{align*}
\]

- **Output**: Approximate saddle point \( (u^K, q^K, p^K) \)
Supervised learning

- We make use of a loss function

\[ \mathcal{L}(F) = \frac{1}{MNS} \sum_{s=1}^{S} \ell(u_s^*(F), t_s), \]

that measures the error between the targets \( t_s \) and the solutions \( u_s^* \).

- Here, the loss function is given by \( \ell(u, t) = \frac{1}{2} \|u - t\|_2^2 \).

- For learning we consider the following bilevel optimization problem:

\[
\min_{F} \mathcal{L}(F) + \mathcal{R}(F), \\
\quad u_s^* \in \arg\min_{u, q} \max_p \langle Du - F^*q, p \rangle + \lambda \|q\|_Z + G(u, g_s), \quad s = 1, \ldots, S
\]

where \( \mathcal{R}(F) \) can be used to impose the constraints on the filters \( F \).

\[
\mathcal{R}(F) = \delta_{C_{\Sigma=1}^{L,2}}(F) = \sum_{l=1}^{L} \delta_{C_{\Sigma=1}^{l,1}}(F^{l,1}) + \delta_{C_{\Sigma=1}^{l,2}}(F^{l,2})
\]
Computing derivatives

▶ For learning, we need the gradient of the loss function with respect to the filter coefficients.
▶ For hard problems such as inpainting, unrolling of primal-dual iterations is out of reach due to memory limitations.
▶ We need to compute the adjoint state

\[
\min_{U} \max_{Q, P} \langle DU - F^* Q, P \rangle + \frac{1}{2} \langle \nabla^2 G(u^*) U, U \rangle - \frac{1}{2} \langle \nabla^2 \|q^*\| Z Q, Q \rangle + \langle \nabla \ell(u^*, t), U \rangle.
\]

▶ In turn the gradient of the loss function with respect to the filters \( F \) is given by

\[
\nabla \mathcal{L}(F) = \frac{1}{MNS} \sum_{s=1}^{S} \nabla_{F} \ell(u^K_s(F), t_s).
\]

with

\[
\langle \nabla_{F} \ell(u^K, t), F \rangle = -\langle Q^K, F p^K \rangle - \langle q^K, F P^K \rangle \iff \nabla_{F} \ell(u^K, t) = -(Q^K \otimes p^K + q^K \otimes P^K).
\]

▶ We propose a “Piggyback” [Griewank, Faure '03] primal-dual algorithm.
▶ Jointly computes the solution of the lower level problem and its adjoint state.
Piggyback primal-dual algorithm

Initialization: \( u^0, U^0 \in X, \; q^0, Q^0 \in Z, \; p^0, P^0 \in Y \).

Step sizes: Choose the block-wise step sizes \( \tau_u, \tau_q, \sigma_p \) such that

\[
\left\| \text{diag}(\sigma_p)^{\frac{1}{2}} (D, F^*) \text{diag}(\tau_u, \tau_q)^{\frac{1}{2}} \right\| \leq 1,
\]

and set \( \theta = 1 \).

Iterations: For each \( k = 0, \ldots, K - 1 \) let

\[
\begin{cases}
  p^{k+1} = p^k + \sigma_p (D u^k - F^* q^k), & \quad P^{k+1} = P^k + \sigma_p (D U^k - F^* Q^k) \\
  \tilde{p}^{k+1} = p^{k+1} + \theta (p^{k+1} - p^k), & \quad \tilde{P}^{k+1} = P^{k+1} + \theta (P^{k+1} - P^k) \\
  \tilde{u}^{k+1} = u^k - \tau_u D^* \tilde{p}^{k+1}, & \quad \tilde{U}^{k+1} = U^k - \tau_u (D^* \tilde{P}^{k+1} + \nabla \ell(u^k, t)) \\
  u^{k+1} = \text{prox}_{\tau_u G} (\tilde{u}^{k+1}), & \quad U^{k+1} = \nabla \text{prox}_{\tau_u G} (\tilde{u}^{k+1}) \cdot \tilde{U}^{k+1} \\
  \tilde{q}^{k+1} = q^k - \tau_q F \tilde{p}^{k+1}, & \quad \tilde{Q}^{k+1} = Q^k - \tau_q F \tilde{P}^{k+1} \\
  q^{k+1} = \text{shrink}_{\tau_q \lambda} (\tilde{q}^{k+1}), & \quad Q^{k+1} = \nabla \text{shrink}_{\tau_q \lambda} (\tilde{q}^{k+1}) \cdot \tilde{Q}^{k+1}
\end{cases}
\]

Output: Approximate saddle point \((u^K, q^K, p^K)\) and corresponding adjoint state \((U^K, Q^K, P^K)\)
Convergence and implementation

- The piggyback algorithm has a linear convergence rate (with a strictly slower convergence rate for the adjoint state) in case the lower level problem is $C^2$ and strongly convex.
- Can be relaxed to strongly convex and locally $C^{2,\alpha}$ (ongoing work).
- In practice works well even for very non-smooth problems (why?).
- Some operations such as the derivatives of the proximal maps are implemented using automatic differentiation (Pytorch).
- For learning we make use of a nonconvex inertial proximal method.

Proximal gradient method for learning

- Initialization: $F^{-1} = F^0 \in (C_{\Sigma=1})^{L,2}$.
- Step sizes: Choose $\alpha^k > 0$, $\beta^k \in [0,1)$.
- Iterations: For $k = 0, \ldots, K - 1$ let

$$\begin{align*}
\bar{F}^k &= F^k + \beta^k \left( F^k - F^{k-1} \right) \\
F^{k+1} &= \text{proj}_{(C_{\Sigma=1})^{L,2}} \left( \bar{F}^k - \alpha^k \nabla L(\bar{F}^k) \right)
\end{align*}$$

- Output: Learned interpolation kernels $F^K$
Learning for inpainting

- We train on 64 images of size $64 \times 64$ with directions uniformly sampled between $[0, 2\pi]$ and we include random subpixel shifts.
- We train on a training set and evaluate on a test set.
- We experiment with different numbers of filters and different symmetry constraints for the filters.

(a) Input images $g_s$

(b) Target images $t_s$
Results

<table>
<thead>
<tr>
<th>Data</th>
<th>FD</th>
<th>RT</th>
<th>CD</th>
<th>L = 2</th>
<th>L = 2 (s)</th>
<th>L = 3</th>
<th>L = 3 (s)</th>
<th>L = 4 (s)</th>
<th>L = 8 (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>135</td>
<td>195</td>
<td>6.69</td>
<td>1.26</td>
<td>1.22</td>
<td>1.19</td>
<td>1.27</td>
<td>0.85</td>
<td>0.77</td>
</tr>
<tr>
<td>Test</td>
<td>134</td>
<td>194</td>
<td>6.33</td>
<td>1.63</td>
<td>1.45</td>
<td>1.29</td>
<td>1.29</td>
<td>0.87</td>
<td>0.82</td>
</tr>
</tbody>
</table>

Table: $10^5 \times$ the mean squared error (MSE) of handcrafted and learned filters evaluated on both the training and test data.

**Note that transpose symmetry is almost automatically learned!**
Filter: $L = 8(s)$

<table>
<thead>
<tr>
<th>Filter 1</th>
<th>Filter 2</th>
<th>Filter 3</th>
<th>Filter 4</th>
<th>Filter 5</th>
<th>Filter 6</th>
<th>Filter 7</th>
<th>Filter 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>98.52</td>
<td>45.31</td>
<td>46.26</td>
<td>47.71</td>
<td>50.09</td>
<td>49.82</td>
<td>52.75</td>
<td>50.33</td>
</tr>
<tr>
<td>49.95</td>
<td>50.71</td>
<td>47.36</td>
<td>47.14</td>
<td>48.97</td>
<td>46.14</td>
<td>47.34</td>
<td>45.31</td>
</tr>
<tr>
<td>97.63</td>
<td>46.36</td>
<td>47.67</td>
<td>47.50</td>
<td>49.33</td>
<td>49.96</td>
<td>52.11</td>
<td>50.76</td>
</tr>
<tr>
<td>50.62</td>
<td>50.05</td>
<td>46.53</td>
<td>44.75</td>
<td>47.83</td>
<td>47.66</td>
<td>46.17</td>
<td>46.56</td>
</tr>
<tr>
<td>98.38</td>
<td>45.21</td>
<td>47.33</td>
<td>47.65</td>
<td>49.48</td>
<td>50.22</td>
<td>52.70</td>
<td>51.18</td>
</tr>
<tr>
<td>51.15</td>
<td>50.22</td>
<td>46.90</td>
<td>47.22</td>
<td>48.71</td>
<td>47.59</td>
<td>47.23</td>
<td>45.51</td>
</tr>
<tr>
<td>98.44</td>
<td>46.58</td>
<td>47.62</td>
<td>48.73</td>
<td>49.46</td>
<td>49.83</td>
<td>51.10</td>
<td>51.28</td>
</tr>
<tr>
<td>48.87</td>
<td>50.20</td>
<td>46.51</td>
<td>45.19</td>
<td>48.36</td>
<td>47.30</td>
<td>45.38</td>
<td>45.07</td>
</tr>
</tbody>
</table>
Comparison

\[
L = 2, n = 3 \text{ (s)}
\]

\[
L = 3, n = 3 \text{ (s)}
\]

\[
L = 4, n = 3 \text{ (s)}
\]

\[
L = 8, n = 3 \text{ (s)}
\]
Learning for disk regularization

- We train on 64 images with binary disks of various radii and subpixel shifted centers.
- The ground truth solutions can be computed with an explicit formula.
- We train on a training set and evaluate on a test set.
- We experiment with different numbers of filters and different symmetry constraints for the filters.

(a) Input images $g_s$

(b) Target images $t_s$
Table: $10^5$ times the mean squared error (MSE) of handcrafted and learned filters for the disk denoising problem.

Observe that $L = 4(s)$ is very close to RT on cubic meshes!
**Comparison**

![Graph showing PSNR vs. rs with different L values (L = 2, 3, 4, 8) for Target, FD, RT, and CD.]

- **PSNR vs. rs**
  - Blue line: FD
  - Red line: RT
  - Green line: CD
  - Black line: L = 2 (s)
  - Pink line: L = 3 (s)
  - Orange line: L = 4 (s)
  - Cyan line: L = 8 (s)

- **Images showing Target and different L values:**
  - Target
  - FD
  - RT
  - CD
  - L = 2 (s)
  - L = 3 (s)
  - L = 4 (s)
  - L = 8 (s)
Natural image denoising

- We extract 64 patches of size $64 \times 64$ from a natural image database.
- The input images $g_s$ contain 5% Gaussian noise.
- We learn both the filter weights and the regularization parameter $\lambda$ by projecting on the set of filters with sum equals $\lambda > 0$.

(a) Input images $g_s$

(b) Target images $t_s$
Results

\[
L = 8 (s), \ 2 \times 2
\]

\[
L = 40 (s), \ 6 \times 6
\]

\[
L = 40, \ 6 \times 6
\]

<table>
<thead>
<tr>
<th>Data</th>
<th>FD</th>
<th>RT</th>
<th>CD</th>
<th>(L = 8 (s))</th>
<th>(L = 40 (s))</th>
<th>(L = 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>5.05</td>
<td>5.33</td>
<td>4.87</td>
<td>4.58</td>
<td>4.31</td>
<td>4.22</td>
</tr>
<tr>
<td>Test</td>
<td>4.72</td>
<td>5.05</td>
<td>4.51</td>
<td>4.28</td>
<td>4.10</td>
<td>4.13</td>
</tr>
</tbody>
</table>

Table: \(10^4\times\) the mean squared error (MSE) of handcrafted and learned filters for natural image denoising.
Comparison

(a) Training set

(b) Test set

(c) Example from the test set
Crossover experiments

- How well do the learned filters generalize to other tasks?
- We compare the filters $L = 8(s)$ which gave good results on all tasks.

<table>
<thead>
<tr>
<th>Evaluation task</th>
<th>Learning task</th>
<th>Handcrafted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Line</td>
<td>Disk</td>
</tr>
<tr>
<td>Line</td>
<td>0.82</td>
<td>243.55</td>
</tr>
<tr>
<td>Disk</td>
<td>1.88</td>
<td>0.47</td>
</tr>
<tr>
<td>Natural</td>
<td>48.68</td>
<td>49.65</td>
</tr>
</tbody>
</table>

- The filters learned for inpainting generalize best, but there is no universal best discretization.
Conclusion

- We proposed learning optimized finite differences discretizations of the total variation.
- The learning is constrained to a class of consistent discretizations which $\Gamma$-converge to the continuous total variation.
- We proposed a piggy-back primal-dual algorithm for computing derivatives.
- Symmetry constraints on the filters give better generalizations.
- The learned discretizations give significant improvements when optimized for certain applications but no best universal discretization could be learned.
Conclusion

- We proposed learning optimized finite differences discretizations of the total variation.
- The learning is constrained to a class of consistent discretizations which $\Gamma$-converge to the continuous total variation.
- We proposed a piggy-back primal-dual algorithm for computing derivatives.
- Symmetry constraints on the filters give better generalizations.
- The learned discretizations give significant improvements when optimized for certain applications but no best universal discretization could be learned.

Thank you for your attention!