J Lellmann

Advanced Color Image Processing – Variational Models

Examples Sessions

1. In this example we will explore several ways of numerically solving variational formulations of the denoising problem for a one-dimensional signal. The `task_tv_1d.m` file contains a MATLAB skeleton code to get you started. You will also have to have CVX set up with the MOSEK solver, see `www.cvxr.com`. The file is split into sections, which can be run individually by pressing Ctrl+Return.

   (a) Run section 1) to load and visualize the input data.

   (b) Section 2) shows how to use CVX for solving the Tikhonov-regularized problem

   $$\min_u \frac{1}{2}\int_\Omega (u-I)^2 dx + \lambda \frac{1}{2}\int_\Omega \|Du\|_2^2 dx, \tag{0.1}$$

   where $I : \Omega \to \mathbb{R}$ is the noisy input data and $\Omega = [0, 5]$. Familiarize yourself with how CVX is used. Try different parameter choices for λ and observe the changes. Try to find a good setting that removes the noise but preserves the jumps. What do you notice?

   (c) From the lectures we know that the Tikhonov problem can also be solved directly by solving a linear equation system. Implement this technique in section 3) and verify that it gives the same result as in 1b).

   (d) The TV-L^1 model

   $$\min_u \frac{1}{2}\int_\Omega |u-I|\, dx + \lambda \int_\Omega |Du| \tag{0.2}$$

   is known to be work particularly well if the data that should be reconstructed is piecewise constant. Implement the model in CVX in section 4). Try different settings for λ and find one that removes the noise but preserves the edges.

   (e) Many solvers require to reformulate the problem into a standard form. So far we have been using CVX to do this transformation, but it is not always available or fast enough. In this section we will see how a problem can be manually reformulated into the standard form.

   We will use the following standard form for Linear Programs (LP):

   $$\min_{x\in\mathbb{R}^n} c^\top x \quad \text{s.t.} \quad Ax \geqslant b. \tag{0.3}$$

   Transform the TV-L^1 problem (0.2) into this standard form. You will most likely need to introduce some extra variables and constraints. Implement the reformulated problem in Section 5) using CVX and verify that it gives the same result as in 1d).

   (f) Solve the Linear Program in 1e) directly using MOSEK's `msklpopt` command. Section 6) shows how to call `msklpopt` to solve problems of the form (0.3). Verify that it gives the same result as the other methods.

2. When working with color images, the choice of the norm can make a subtle but important difference. In this example we will look at the simple Frobenius norm and compare it to the more advanced nuclear norm. The source code for this example can be found in `task_colornorms.m`.

   (a) Run section 1) and complete the missing parts in section 2) to solve the Rudin-Osher-Fatemi model

   $$\min_u \frac{1}{2}\int_\Omega \|u-I\|_2\, dx + \lambda \int_\Omega \|Du\|_F, \tag{0.4}$$

   with the Frobenius norm $\|A\|_F = (\sum_{i,j} A_{ij}^2)^{1/2}$. The norms can be computed using the `norms` command (see `help norms`).

(b) We now would like to solve the model with the nuclear norm instead of the Frobenius norm:

$$\min_u \frac{1}{2} \int_\Omega \|u - I\|_2 \, dx + \lambda \int_\Omega \|Du\|_*. \tag{0.5}$$

From the lectures we know that the nuclear norm can be rewritten as a semidefinite program. Use this fact to implement the missing parts in section 3) to solve model (0.5) using CVX. Semidefinite constraints can be enforced in CVX by using the constraint notation

```
A == semidefinite([k k n]),
```

which enforces that each matrix `A(:,:,i)` (for `i=1:end`) be positive semidefinite. It might be useful to know that MATLAB allows to concatenate multi-dimensional matrices as follows: If `A` is an array of `n` matrices of size `[k k]` each, i.e., `size(A)=[k k n]`, and `size(B) = [k m n]`, then `size([A B]) = [k k+m n]`, and similar for vertical concatenation `[A;B]`. Another useful command might be `permute(A,[2 1 3])`, which transposes the first two dimensions of `A`.

Compare the results to the results obtained using the Frobenius norm. What are the differences?

3. In the lectures we have seen that combinatorial problems such as image segmentation can be solved by solving a convex relaxed problem instead. A simple approach for solving the segmentation problem with `m` labels is to use the following constrained variational model:

$$\min_{u:\Omega\to\mathbb{R}^m} \quad \sum_{i=1}^m \int_\Omega s(x,i)u_i(x)dx + \lambda \int_\Omega \|Du\|_F. \tag{0.6}$$

$$\text{s.t.} \quad u \geqslant 0, \quad \sum_{i=1}^m u_i(x) = 1, \, \forall x \in \Omega. \tag{0.7}$$

A skeleton for solving this problem is provided in `task_segmentation.m`.

(a) Run section 1) to load and visualize the input image.

(b) The goal is to segment this image into three classes: sky (class 1), flowers (class 2), and leaves (class 3). For each of the classes, a typical color is provided in the vector `c(i,:)`. Complete section 2) for computing the costs `s` so that the cost for assigning class label i to a point x is $\|I(x) - c(i,:)\|_2$, where $I(x)$ is the color of the input image at point x. Run the section to visualize what the segmentation looks like if only the local features are used.

(c) Implement the model (0.7) in section 3) and solve it using CVX. Find a good setting for the parameter λ and compare the results to the pointwise segmentation.