Linear Programming Project (Classical Track)
Recovering a Blurred Image via ℓ_1 Deblurring

M1 Optimization (teams of 2; beginners in LP welcome)

September 19, 2025

Abstract

Your camera has failed: instead of returning each pixel intensity, it reports a weighted average of neighboring pixels, and a small set of sensors are faulty (sparse corruption). You receive a blurred and partially corrupted image $\tilde{x}$ and want to approximately reconstruct the original image $\bar{x} \in [0,1]^n$. We will cast this as a *linear program* (LP) and solve it with standard LP solvers.

1 Problem description

Blurring model. Let $\bar{x} \in [0,1]^n$ be the (unknown) vector of original pixel intensities. We observe a blurred image $\tilde{x} \in [0,1]^n$:

$$\tilde{x} \;=\; A\,\bar{x},$$

where $A \in \mathbb{R}^{n\times n}$ is the *blurring matrix*. In simple box-average blurs, each entry of $\tilde{x}$ is the average of neighboring entries of $\bar{x}$. In general, A is not invertible, so $Ax = \tilde{x}$ may be infeasible (in the presence of noise) or underdetermined.

A common surrogate is to seek a low-energy solution:

$$\min_{0\le x\le 1} \quad \|x\|_1 \quad \text{s.t.} \quad Ax = \tilde{x},$$

where $\|x\|_1 = \sum_i |x_i|$.

Sparse (impulsive) noise. In practice the observation is corrupted:

$$\tilde{x} \;=\; A\,\bar{x} \;+\; b,$$

where b is *sparse* (only a small number of pixels are corrupted). For sparse noise, an ℓ_1 data term is more robust than ℓ_2. We will estimate $\bar{x}$ by solving

$$\min_{0\le x\le 1} \quad \|Ax - \tilde{x}\|_1 \;+\; \lambda\,\|x\|_1\,, \tag{1}$$

with a positive parameter λ (tuned to noise level). The goal of this project is to study (1) and reconstruct $\bar{x}$ from A and $\tilde{x}$.

Remark (blur kernel). In real applications, the blurring matrix A is often only approximately known; there is a large literature on estimating blur kernels. In this project, *assume A is known* (kernel estimation itself is an optimization problem).

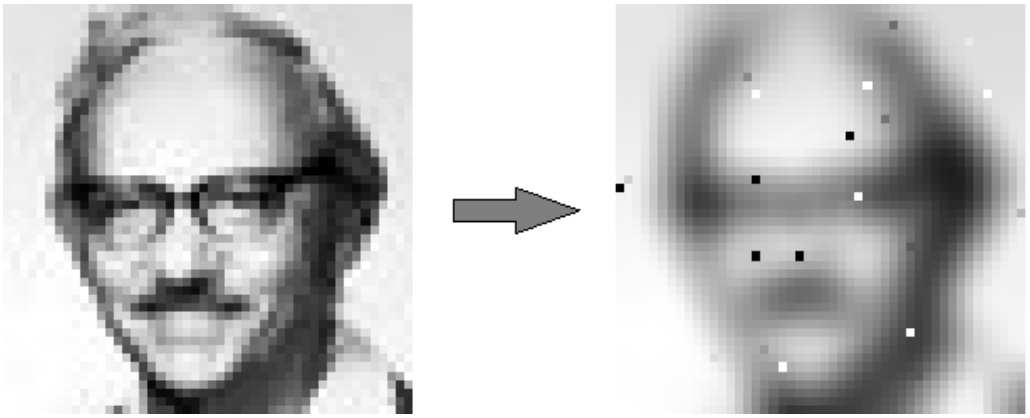

Figure 1: Left: original image. Right: blurred and sparsely corrupted observation.

2 What you will do

1. **Model as an LP.** Show how problem (1) can be written as a linear program. Explain your reasoning (introduce auxiliary variables as needed; keep it clear and compact).
2. **Write the LP in standard form.** Provide the canonical LP standard form

$$\min c^\top x \quad \text{s.t.} \quad Ax = b, \quad x \geq 0,$$

for your formulation, specifying the stacked decision variable, the constraint blocks, and the bounds.

3. **Solve on the provided images.** Use any LP solver you like:
   - *GNU Octave/GLPK* (`glpk`) or MATLAB (`linprog`);
   - *Python SciPy* (`scipy.optimize.linprog` with `method="highs"`).

   Deblur the files `ExampleX.mat` for $X \in \{0, 1, 2\}$. Show the recovered images and basic metrics.
4. **Vertex test.** Is the solution you obtained a *vertex* (extreme point) of the feasible polyhedron of your LP? Briefly justify.
5. **Sensitivity to λ.** For `Example0.mat`, study reconstruction error as a function of λ (the original image is provided). Which λ values perform best for each example? Use logarithmic scales for clarity. Although the true image for `Example1.mat` is not provided, run the analysis for multiple values of λ, and report your observations. Which value seems to provide the best recovered image ?

3 Practical guidance

Input format. Each `ExampleX.mat` contains:

- the blurred observation $\tilde{x}$ (as a vector or reshaped image),
- the blurring matrix A (or an operator you can apply as a matrix–vector product),
- and, for evaluation, the ground-truth image $\bar{x}$ for selected examples.

Scaling and types. Work with pixel values in $[0, 1]$. If you reconstruct a vector $x^\star$, reshape it to the original image size for visualization.

Solvers. Any standard LP solver is acceptable (Octave/GLPK, MATLAB `linprog`, Python `linprog`). If you use Python, prefer the HiGHS backend (`method="highs"`). Keep your model sparse for speed and memory.

Plots to include.

- side-by-side images: original (when available), blurred input, reconstruction;
- ℓ_2 reconstruction error $\|x^\star - \bar{x}\|_2$ vs. λ (log scale on axes);
- optionally, the residual image $|Ax^\star - \tilde{x}|$ (to see which pixels were treated as outliers).

4 Deliverables and logistics

Teams. Work in teams of **2** students. Students with prior LP background—or students from Innopolis—should take the *research track* project on TSDP.

Milestone 1 (midterm). *Standard-form model.* Produce c, A, b and the stacked variable x exactly as in the LP standard form. Provide a short table explaining each block of A.

Final deliverables - What to submit.

- **Report** (max 10 pages, excluding code appendix and cover page): concise modeling, standard form, solver setup, experiments, sensitivity study, and key figures.
- **Code appendix**: all scripts used to run your experiments (Octave/MATLAB/Python).

Submission. Submit on Moodle by the deadline (see course page). Late policy per course rules.

Notes and references

- ℓ_1 data fidelity is robust for sparse (impulsive) noise; ℓ_2 is appropriate for Gaussian noise.
- Octave (GLPK), MATLAB (`linprog`), or Python (SciPy `linprog`) are all fine for this project.