Notes on Regularization and Robust Estimation

Psych 267 / CS 348D / EE 365 Prof. David J. Heeger

September 15, 1998

1 Regularization.

Regularization is a class of techniques that have been widely used to solve interpolation problems that frequently arise in image processing, computer vision, and computer graphics [1, 2, 3, 8, 10, 12, 11, 13]. The techniques are designed to find a "smooth", discrete signal v given a set of noisy and partial measurements u. For a one-dimensional signal, the relationship between v and u may be modeled by:

$$u[x] = \mathcal{A}(v[x]) + n[x], \quad x \in P \tag{1}$$

where $\mathcal{A}$ is a non-invertible operator, n is a random field (for example, a field of independent, zero mean, Gaussian random variables with common variance), and P is a subset of sample positions where the measurements u are available.

Denoising. To start with a simple example, we assume that $\mathcal{A}$ is the identity operator and that the noisy measurements u are available at every pixel. That is, we observe a noisy signal u and we wish to denoise it to recover v. Of course, this problem as stated is underconstrained; we need to know something about the original signal v and/or the noise. A typical regularization solution assumes that the original signal is smooth and chooses v to minimize the following set of constraints:

$$E(v) = \sum_{x=1}^{N} (v[x] - u[x])^2 + \lambda \sum_{x=1}^{N-1} (v[x+1] - v[x])^2, \tag{2}$$

where N is the number of samples in the signal. The first term is called the data constraint, specifying that the solution v should be close to the measurements u. The second term is called the smoothness constraint, specifying that neighboring sample values should be similar. The parameter λ determines the tradeoff between the two constraints; if λ is large then the solution will be smoother at the expense of being further from the measurements. It is worth noting that we have (rather arbitrarily) adopted a particular definition of what "smoothness" means, a point that we will return to later. It is also worth noting that we have (arbitrarily) used quadratic error functions, another point that we will return to later.

Taking the derivative of Eq. 2 with respect to the k^{th} sample value $v[k]$ gives:

$$\begin{aligned} \frac{\partial E(v)}{\partial v[k]} &= \frac{\partial}{\partial v[k]} \left[(v[k] - u[k])^2 + \lambda (v[k+1] - v[k])^2 + \lambda (v[k] - v[k-1])^2 \right] \\ &= 2 (v[k] - u[k]) + 2\lambda (-v[k-1] + 2v[k] - v[k+1]). \end{aligned}$$

Taking the derivatives for all sample values and setting them equal to zero gives a set of linear equations of the form:

$$v[x] + \lambda\left(-v[x-1] + 2v[x] - v[x+1]\right) = u[x]. \tag{3}$$

This full system of linear equations can be written using matrix notation:

$$\begin{pmatrix} 1+2\lambda & -\lambda & 0 & 0 & \dots & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 & \dots & 0 \\ 0 & -\lambda & 1+2\lambda & -\lambda & \dots & 0 \\ & & \ddots & \ddots & \ddots & \\ 0 & \dots & & 0 & -\lambda & 1+2\lambda \end{pmatrix} \begin{pmatrix} v[1] \\ v[2] \\ v[3] \\ \vdots \\ v[N] \end{pmatrix} = \begin{pmatrix} u[1] \\ u[2] \\ u[3] \\ \vdots \\ u[N] \end{pmatrix}, \tag{4}$$

and it could be solved in principle by inverting the matrix. But this is not practical when N is very large, especially for the case in which v and u are two dimensional images and N is the number of pixels.

An alternative is to solve for v iteratively:

$$v^{(i+1)}[x] = \tfrac{1}{1+2\lambda}\left(u[x] + \lambda v^{(i)}[x-1] + \lambda v^{(i)}[x+1]\right). \tag{5}$$

Note that Eq. 5 converges ($v^{(i+1)} = v^i$) when Eq. 3 is satisfied, i.e., when Eq. 2 is minimized.

Smoothness Constraints. The particular smoothness constraint used above is often called the (first-order) membrane model. It is a discrete approximation to the first derivative:

$$v[x+1] - v[x] \approx \tfrac{dv}{dx}.$$

This smoothness constraint is satisfied perfectly when v is a constant signal.

Another common choice for the smoothness constraint is called the (second-order) thin-plate model which is a discrete approximation to the second derivative:

$$v[x+1] - 2v[x] + v[x-1] \approx \tfrac{d^2v}{dx^2}.$$

This smoothness constraint is satisfied perfectly when v is a linear ramp signal.

But these smoothness constraints have been chosen somewhat arbitrarily. Choosing the "right" smoothness constraint, depends on having a prior model for the expected smoothness of the signal. The regularization solution may be derived from Bayesian estimation theory in which prior information about v is introduced using a Markov Random Field model [3, 8]. Unfortunately, real images are not particularly well-modeled by Markov Random Fields so smoothness constraints are typically ad hoc.

Linear Filter Interpretation. Equation 3 can be written with a convolution:

$$v = u - \lambda(g * v), \tag{6}$$

with $\vec{g} = [-1, 2, -1]$. Recall that Eq. 3 was derived from the the (first-order) membrane smoothness constraint. Starting with the (second-order) thin-plate smoothness constraint, we would get an equation of the same form but with a different filter: $\vec{g} = [1, -4, 6, -4, 1]$. Indeed, any high-pass filter could, in principle, be used instead.

Hence, regularization is a (spatially) shift-invariant, feedback, linear system, as illustrated in Fig. 1. Taking the Fourier transform (DTFT) of both sides of Eq. 3 gives:

$$V(w)[1 + \lambda(2 - e^{-jw} - e^{jw})] = U(w),$$

where V and U are the Fourier transforms of v and u, respectively. This simplifies to:

$$V(w)[1 + 2\lambda(1 - \cos(w))] = U(w),$$

where we have used the identity:

$$2\cos(jw) = e^{-jw} + e^{jw}.$$

One can now see that the regularization operation also corresponds to convolution with a *feedforward*, linear filter with frequency response:

$$H(w) = \frac{V(w)}{U(w)} = \frac{1}{1 + 2\lambda(1 - \cos(w))}, \tag{7}$$

In other words, the matrix equation (Eq. 4) and the gradient descent (Eq. 3) are equivalent to convolving the measurements u with an appropriate linear filter. If we adopt a different smoothness constraint, i.e., a different feedback filter $g[x]$, then from Eq. 6, the equivalent feedforward linear filter has frequency response:

$$H(w) = \frac{1}{1 + \lambda\, G(w)}, \tag{8}$$

where $G(w)$ is the Fourier transform of $g[x]$. Increasing the value of λ makes the output smoother, i.e., the feedforward filter becomes more lowpass (see Fig. 2).

So regularization is the same as convolution (with an appropriate filter). Note, however, that the iterative (gradient descent) algorithm still has a number of practical advantages. First, the feedforward convolution kernel $h[x]$ is typically very large, depending on the exact choice of $g[x]$ (see Fig. 1). Second, the iterative algorithm can be readily extended to deal with the general case (Eq. 1) in which the measurements are a non-invertible function of the original signal. Third, iterative gradient descent can also be used for *robust* error functions (see below).

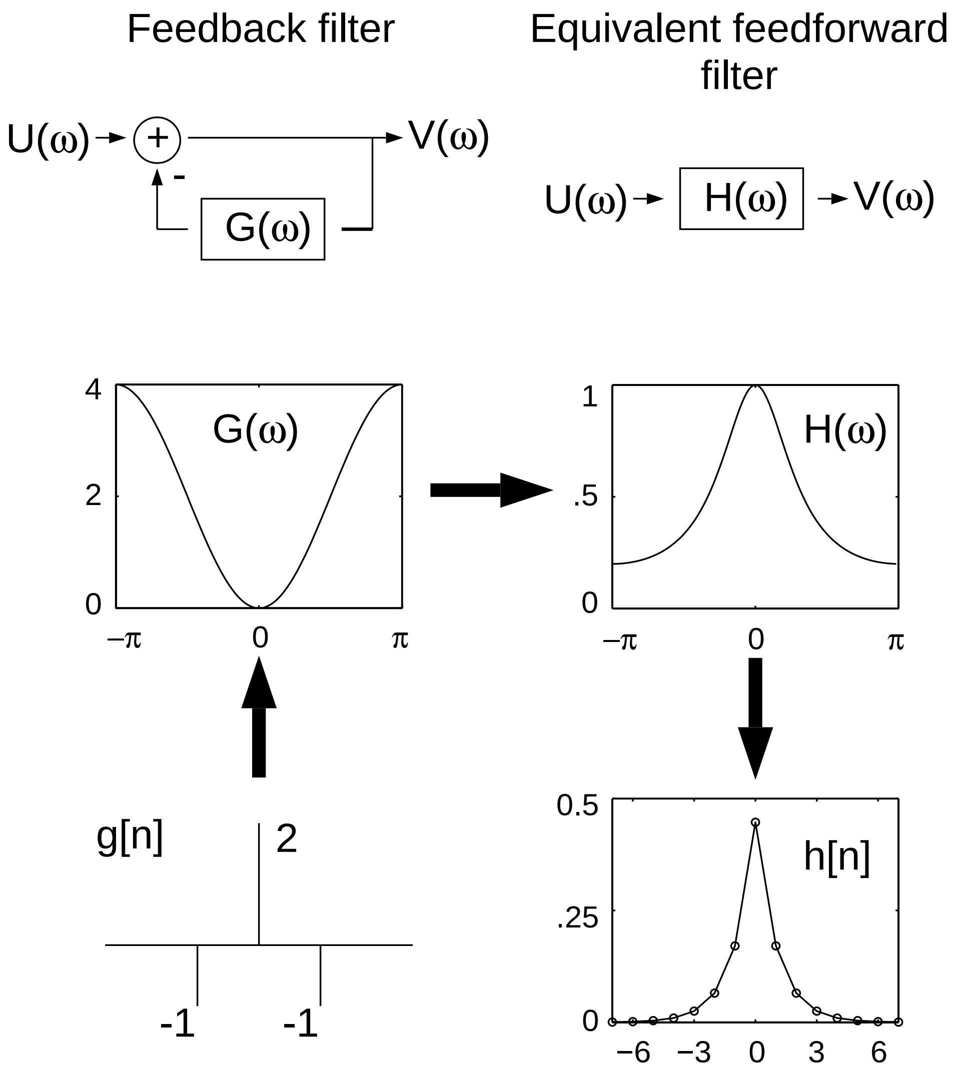

Figure 1: Regularization as a shift-invariant, feedback, linear system: $G(w)$ is the frequency response of the feedback filter and $H(w)$ is the frequency response of the equivalent feed-forward filter for $\lambda = 1$

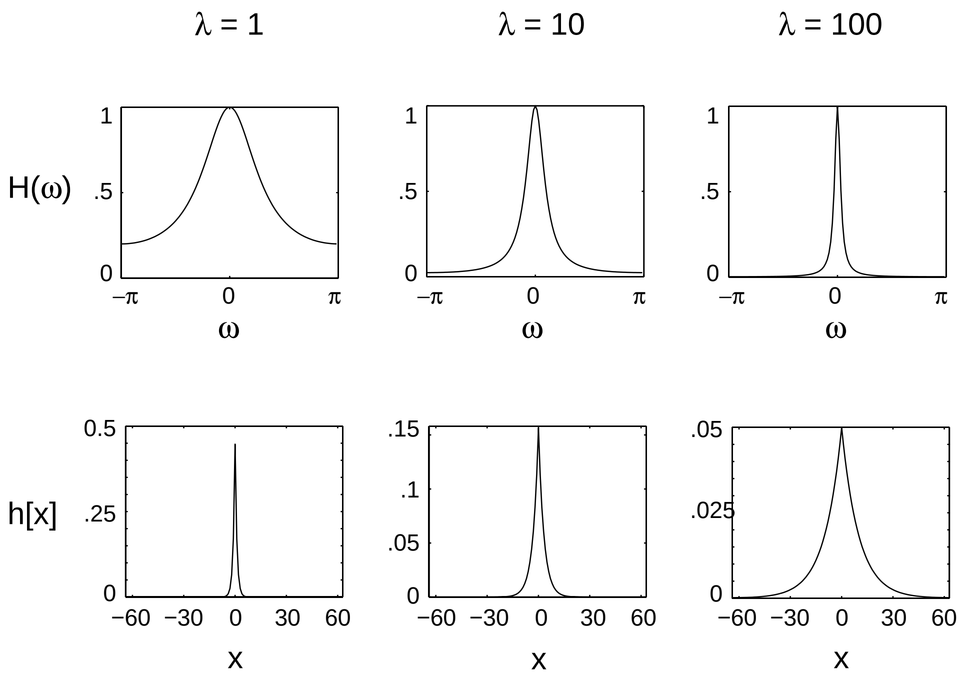

Figure 2: Frequency response $H(w)$ and impulse response $h[x]$ of regularization filter, for several values of λ.

Interpolating Missing Data. The regularization framework can be used to interpolate, as well as smooth, the input. Consider the problem of estimating a smooth signal $v[x]$ from a set of noisy measurements u where the measurements exist at only a subset of the sample positions. Equation 2 may now be rewritten as follows:

$$E(v) = \sum_{x \in P} (v[x] - u[x])^2 + \lambda \sum_{\text{all}\, x} (v[x+1] - v[x])^2, \tag{9}$$

where P is a subset of sample positions where the measurements u are available. For samples where the measurements exist, the gradient descent is the same as before (Eq. 3). For a sample position k where no measurement exists, taking the derivative of Eq. 9 with respect to $v[k]$ and setting it equal to zero gives:

$$-v[k-1] + 2\, v[k] - v[k+1] = 0.$$

The full system of linear equations can still be written as a matrix equation, i.e., it is still a linear system and it could again be solved by inverting a *big* matrix. The matrix in Eq. 4 gets replaced by a new matrix in which some of the rows look the same, but some of the rows (those corresponding to missing data) look like: $(0 \quad \ldots \quad -1 \quad 2 \quad -1 \quad \ldots \quad 0)$, with zeros substituted for the corresponding $u[k]$ on the right-hand side. Since these rows are different from the other rows, the system is no longer shift-invariant, and there is no longer a convolution kernel that could be used to compute v.

An iterative, gradient descent algorithm can again be used to solve for v. For the

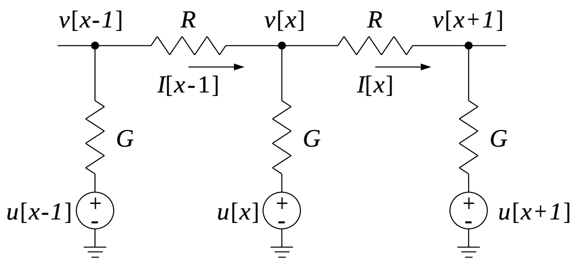

Figure 3: Regularization implemented in hardware as a resistive grid.

(first-order) membrane model smoothness constraint, the iterative equation is:

$$v^{(i+1)}[x] = \begin{cases} \frac{1}{1+2\lambda}\left(u[x] + \lambda v^{(i)}[x-1] + \lambda v^{(i)}[x+1]\right) & \text{for } x \in P \\ \\ \frac{1}{2}\left(v^{(i)}[x-1] + v^{(i)}[x+1]\right) & \text{otherwise} \end{cases} . \tag{10}$$

Harware Implementation. Fig. 3 shows a "resistive grid" that performs regularization. The inputs $u[x]$ are voltage sources. The outputs $v[x]$ are the voltages at the nodes of the grid. Each output node is connected through resistors with resistance R to its neighbors. Each output node is also connected through resistors with conductance G to the input voltage sources.

To determine the behavior of this circuit, we write the current flowing through each resistor. The currents through the resistors labeled R are:

$$\begin{aligned} I[x-1] &= 1/R\,(v[x-1] - v[x]) & (11) \\ I[x] &= 1/R\,(v[x] - v[x+1]). & (12) \end{aligned}$$

The current through the resistor labeled G is:

$$I[x-1] - I[x] = G\,(v[x] - u[x]). \tag{13}$$

Substituting the first two equations into the third and reorganizing gives:

$$v[x] + \frac{1}{RG}\left(-v[x-1] + 2v[x] - v[x+1]\right) = u[x],$$

which is the same as Eq. 3 with $\lambda = 1/(RG)$.

Endpoints. The gradient descent (Eq. 5) must obviously be handled differently at the two endpoints, $x = 1$ and $x = N$, because only one of the two neighbors exists. The regularization framework provides a natural way to handle these endpoints. Taking the derivative of Eq. 9 with respect to $v[1]$, setting it equal to zero, and solving for $v[1]$ gives a slightly different iterative equation:

$$v^{(i+1)}[1] = \begin{cases} \frac{1}{1+\lambda}\left(u[1] + \lambda v[2]\right) & \text{for } x \in P \\ \\ v[2] & \text{otherwise} \end{cases} .$$

The equation for $v[N]$ is analogous.

Two-Dimensional Images. For two-dimensional images, using the (first-order) membrane smoothness constraint, we wish to minimize:

$$\begin{aligned} E(v) &= \sum_{x,y \in P} (v[x,y] - u[x,y])^2 \\ &\quad + \lambda \sum_{\text{all}\, x,y} (v[x+1,y] - v[x,y])^2 + (v[x,y+1] - v[x,y])^2 \end{aligned} \tag{14}$$

where P is a subset of pixels where the measurements u are available. Taking derivatives with respect to $v[x,y]$ and setting them equal to zero gives a linear system of equations that has the same form as before. The only difference is that the linear filter $g[x,y]$ is now 2-dimensional. For the (first-order) membrane model smoothness constraint:

$$g = \begin{pmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{pmatrix}.$$

One can again solve for v iteratively. For the (first-order) membrane smoothness model, and ignoring the edge pixels (for the sake of simplicity):

$$v^{(i+1)}[x,y] = \begin{cases} \frac{1}{1+4\lambda}(u[x,y] + \lambda\, s^{(i)}[x,y]) & \text{when}\, x,y \in P \\ \frac{1}{4}\, s^{(i)}[x,y] & \text{otherwise} \end{cases}, \tag{15}$$

where $s[x,y]$ is the sum of the 4 nearest neighbors, i.e., $v[x-1,y] + v[x+1,y] + v[x,y-1] + v[x,y+1]$.

2 Robust Regularization.

The field of robust statistics [4, 7] is concerned with estimation problems in which the data contains gross errors, or *outliers* that do not conform to the statistical assumptions for the majority of the data (e.g., Gaussian noise). The main goals of robust statistics are: "*(i)* To describe the structure best fitting the bulk of the data, *(ii)* To identify deviating data points (outliers) or deviating substructures for further treatment, if desired."

Denoising (revisited). Let us again consider the problem of denoising a one-dimensional input signal. A generalization of the formulation in Eq. 2 is to minimize the following set of constraints:

$$E(v) = \sum_{x=1}^{N} \rho_d(v[x] - u[x], \sigma_d) + \lambda \sum_{x=1}^{N-1} \rho_s(v[x+1] - v[x], \sigma_s), \tag{16}$$

where ρ_d and ρ_s are error norms and σ_d and σ_s are scale parameters. The least-squares formulation (Eq. 2) is the special case in which $\rho(z,\sigma) = z^2$. But the least-squares approach

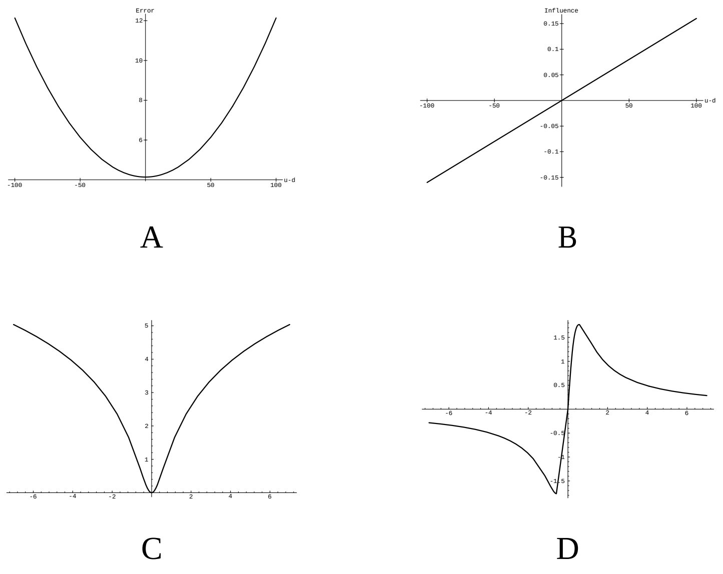

Figure 4: **A:** Least-square error (ρ) function. **B:** Least-squares influence (ρ') function. **C:** Lorentzian error (ρ) function. **D:** Lorentzian influence (ρ') function.

is notoriously sensitive to outliers; the problem being that outliers contribute too much to the overall solution.

To analyze the behavior of a ρ-function, we consider its derivative (denoted ρ') which is called the *influence function*. The influence function characterizes the bias that a particular measurement has on the solution. For example, the quadratic ρ-function has a linear ρ'-function:

$$\rho(z) = z^2, \quad \rho'(z) = 2z.$$

For least-squares estimation, the influence of outliers increases linearly and without bound (Fig. 4).

To increase robustness, an estimator must be more forgiving about outlying measurements; that is it should increase less rapidly. For example, consider the *Lorentzian* error norm:

$$\rho(z, \sigma) = \log\left(1 + \frac{1}{2}\left(\frac{z}{\sigma}\right)^2\right), \qquad \rho'(z, \sigma) = \frac{2z}{2\sigma^2 + z^2}. \tag{17}$$

The Lorentzian error norm (ρ) is plotted along with its influence function (ρ') in Fig. 4. Examination of the ρ'-function reveals that when the absolute value of a residual increases beyond a threshold, its influence decreases. The Lorentzian is one of many possible robust error functions (see [1] for other options).

The least-squares regularization formulation assumes that the signal is equally smooth throughout. Figure 5 shows a "step" input with added noise. Linear regularization

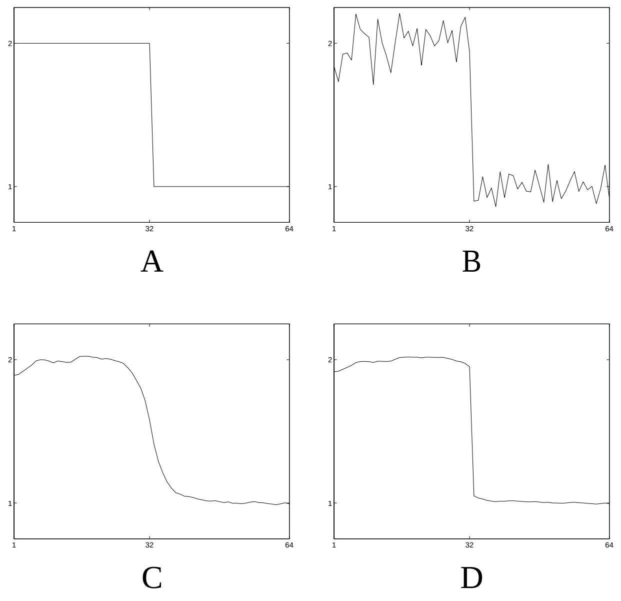

Figure 5: **A:** Original step signal. **B:** Noisy step signal. **C:** Regularization result using quadratic error norm for both the data constraint and the smoothness constraint. **D:** Regularization result using a quadratic norm for the data constraint, but a robust Lorentzian error norm for the smoothness constraint.

removes the noise appropriately, but it also blurs the step edge. Robust regularization smooths the smooth part while keeping the step sharp.

Likewise, the least-squares regularization formulation assumes that the error/noise in the measurements is independent and identically distributed (IID). Consider a signal in which the k^{th} input measurement is very badly corrupted. This will force the corresponding output value to be way off to minimize the squared difference between the two: $(u[k] - v[k])^2$. This also forces neighboring output values to be affected to minimize the smoothness terms: $(v[k+1] - v[k])^2$ and $(v[k] - v[k-1])^2$. This effect ripples from one neighboring sample to the next, so that the one bad measurement throws off the entire solution. Robust regularization allows the data constraint to be violated near the bad measurement.

Robust regularization can interpolate missing data, simply by leaving out the data constraint for those samples (as was done above for linear regularization). Robust regularization, like linear regularization, can be extended to two or more dimensions, and it can handle endpoints and edge pixels in a natural way.

Implementation Given a robust regularization formulation, there are numerous techniques that can be employed to recover the smoothed and interpolated signal. In general, robust formulations do not admit closed form solutions. A standard approach is to use a numerical optimization algorithm such as Newton's method.

Newton's method minimizes a function iteratively. For example, to minimize a function $f(x)$, Newton's method iterates:

$$z_m^{(i+1)} = z_m^{(i)} - \frac{f'(z_m^{(i)})}{f''(z_m^{(i)})}, \tag{18}$$

where z_m is the estimate of the location of the minimum (i.e., $f(z_m) < f(z)$ for all z). The functions $f'(z)$ and $f''(z)$ are, respectively, the first and second derivatives of f.

To apply Newton's method to minimize Eq. 16, we write its first and second derivatives with respect to $v[k]$:

$$\begin{aligned} \frac{\partial E}{\partial v[k]} &= \rho_d'(v[k]-u[k]) + \lambda\,\rho_s'(v[k]-v[k-1]) - \lambda\,\rho_s'(v[k+1]-v[k]) \\ \frac{\partial^2 E}{\partial (v[k])^2} &= \rho_d''(v[k]-u[k]) + \lambda\,\rho_s''(v[k]-v[k-1]) + \lambda\,\rho_s''(v[k+1]-v[k]) \end{aligned}$$

For example, using quadratic error functions (least-squares), substituting for the derivatives in Eq. 18 gives:

$$\begin{aligned} v^{(i+1)}[x] &= v^{(i)}[x] - \frac{(v^{(i)}[x]-u[x]) + \lambda(-v^{(i)}[k-1] + 2v^{(i)}[k] - v^{(i)}[k+1])}{1+2\lambda} \\ &= \tfrac{1}{1+2\lambda}\,(u[x] + \lambda v^{(i)}[k-1] + \lambda v^{(i)}[k+1]), \end{aligned}$$

which is identical to Eq. 5.

Robust formulations typically result in nonconvex optimization problems. To find a globally optimal solution when the objective function is nonconvex we choose a robust ρ-function with a scale parameter, and we adjust the scale parameter to construct a convex approximation. This approximation is readily minimized. Then successively better approximations of the true objective function are constructed by slowly adjusting the scale parameter back to its original value. This process is sometimes called *graduated non-convexity* [2].

For example, Fig. 6 shows a family of Lorentzian error and influence functions for several values of σ. For larger values of σ, the error functions become more like quadratics and the influence functions become more like lines. The nonconvex Lorentzian error function becomes a simple (convex) quadratic when σ is very large. To compute the result in Fig. 5D, I initially set $\sigma_s = 10$ and then gradually reduced it to a value of 0.1.

The graduated non-convexity algorithm begins with the convex (quadratic) approximation so the initial estimate contains no outliers. Outliers are gradually introduced by lowering the value of σ and repeating the minimization. While this approach works well in practice, it is not guaranteed to converge to the global minimum since, as with least squares, the solution to the initial convex approximation may be arbitrarily bad.

Measurements beyond some threshold, τ, can be considered outliers. The point where the influence of outliers first begins to decrease occurs when the second derivative of the

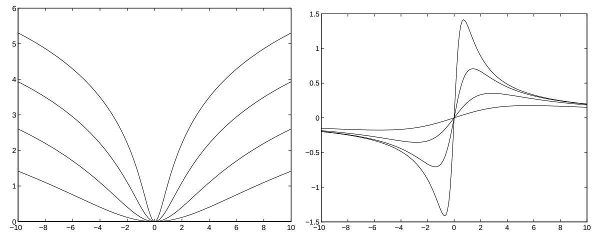

Figure 6: Lorentzian error functions and (left) influence functions (right) for several values of σ (1/2, 1, 2, and 4).

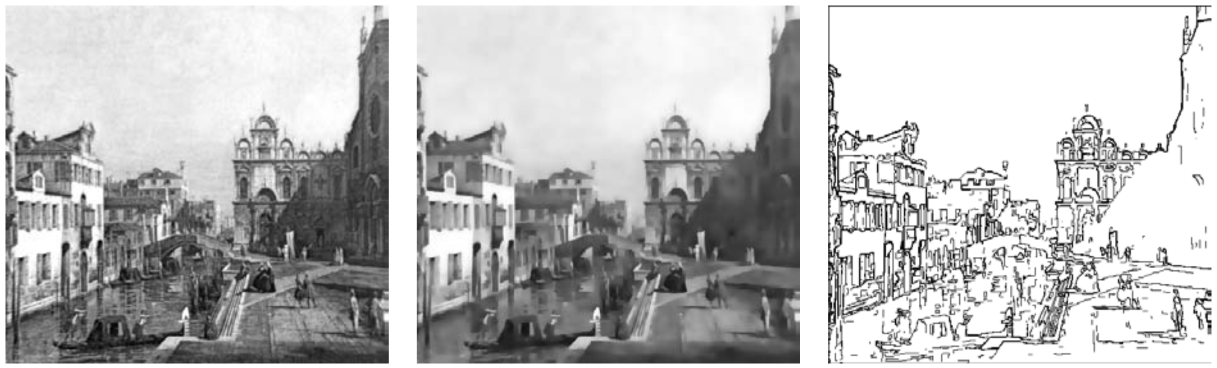

Figure 7: (Left) Input Image. (Middle) Regularization result using a robust Lorentzian error norm for the smoothness constraint. (Right) Edges obtained as outliers at pixels where the Lorentzian error norm was above a threshold.

ρ-function is zero. For the Lorentzian, the second derivative,

$$\rho''(z) = \frac{2(2\sigma^2 - x^2)}{(2\sigma^2 - x^2)^2},$$

is zero when $\tau = \pm\sqrt{2}\,\sigma$. If the maximum expected residual is τ, then choosing $\sigma = \tau/\sqrt{2}$ will result in a convex optimization problem. A similar treatment applies to other robust ρ-functions. Note that this also gives a simple test of whether or not a particular residual is treated as an outlier. In the case of the Lorentzian, a residual is an outlier if $|x| \geq \sqrt{2}\,\sigma$.

Figure 7 shows an example of applying robust regularization to a two-dimensional image, using a quadratic error norm for the data constraint and a Lorentzian error norm for the (first-order) membrane model smoothness constraint. Note that the regularization result is essentially a piecewise constant image, i.e., an image made up of constant regions separated by sharp discontinuities.

Piecewise Smoothness and Segmentation. In linear regularization (Eqs. 2 and 9), the (first-order) membrane model smoothness constraints are satisfied perfectly when the first derivative of the output v is everywhere zero, i.e., when the output is a constant signal. For robust regularization (Eq. 16), the robust first-order smoothness constraints are minimized by a *piecewise* constant signal (see Figs. 5 and 7).

Likewise, the (second-order) thin-plate model smoothness constraints in linear regularization are satisfied when the second derivative of the output is everywhere zero, i.e., when v is a linear ramp. For robust regularization, the robust second-order smoothness constraints are minimized by a signal that is made up of linear ramps separated by sharp discontinuities.

Regularization with Line Processes. Another approach to extending linear regularization has been to add *line processes,* which allows one to recover a piecewise smooth signal/image by marking the specific locations of discontinuities. For example, regularization using the (first-order) membrane model smoothness constraint with a line process minimizes the following error function:

$$E(v,l) = \sum_{x=1}^{N} (v[x] - u[x])^2 + \lambda \sum_{x=1}^{N} [(v[x+1] - v[x])\, l[x] + \Psi(l[x])], \tag{19}$$

where $l[x]$ takes on values $0 \leq l[x] \leq 1$. The line process indicates the presence ($l[x] \to 0$) or absence ($l[x] \to 1$) of a discontinuity between neighboring samples. The function $\Psi(l[x])$ can be thought of as the "penalty" for introducing each discontinuity. Penalty functions typically go to 1 as $l[x]$ tends to 0 and vice versa. Thus, when no discontinuity is present, the smoothness term has the original least-squares form, but when a discontinuity is introduced, the smoothness term is dominated by Ψ. An example penalty function is: $\Psi(z) = (1-z)$. Minimizing Eq. 19 with respect to $v[x]$ and $l[x]$ gives a piecewise smooth signal with breaks where the spatial gradient is too large.

Black and Rangarajan [1] have recently unified the line process regularization formulation with the robust regularization formulation. That is, given a penalty function in Eq. 19, one can derive a robust error norm for the smoothness constraint in Eq. 16, and vice versa, so that the two optimization problems will be identical. For example, using the penalty function $\Psi(z) = z - 1 - \log(z)$ in Eq. 19 is the same as using a Lorentzian error norm for the smoothness constraint in Eq. 16.

Analog VLSI Harware Implementation. Mead and colleagues [9, 5, 6] have used analog VLSI technology to build devices that perform robust regularization. The analog VLSI circuits behave much like a resistive grid (Fig. 3), except that the resistors are replaced with "robust" resistors. The robust resistors are actually circuits made up of several transistors. A true resistor has a linear current-voltage (I-V) curve; Ohm's Law states that the current through a resistor is proportional to the voltage drop across the resistor. The aVLSI circuits, on the other hand have nonlinear I-V curves. In Mead's original circuit design, the robust resistors have hyperbolic tangent I-V curves (Fig. 8). For small voltages, the current through the robust resistor is proportional to the voltage. For large voltages, however, the current does not increase further. It turns out that the I-V curve takes the place of the influence function and the integral of the I-V curve takes the place of the error norm in the robust regularization formulation. Mead's circuit therefore

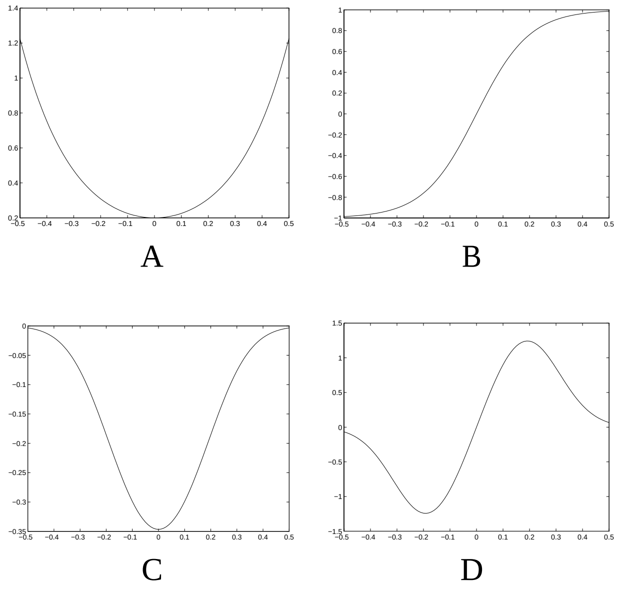

Figure 8: I-V curve / influence function (**B**) and its integral / error norm (**A**) of Mead's aVLSI robust resistor (see Eq. 20). I-V curve / influence function (**D**) and its integral / error norm (**C**) of the aVLSI resistive fuse circuit (see Eq. 21).

minimizes Eq. 16 with:

$$\begin{aligned} \rho(z) &= I_{sat}\, R\, \log\left[\cosh\left(\tfrac{z}{I_{sat}\, R}\right)\right] \\ \rho'(z) &= I_{sat}\, \tanh\left(\tfrac{z}{I_{sat}\, R}\right) \end{aligned} \tag{20}$$

Recent work has developed a more robust aVLSI circuit that has been called a "resistive fuse" [5, 6]. Using the resistive fuse in place of the resistors in a resistive grid minimizes Eq. 16 with:

$$\begin{aligned} \rho(z) &= \tfrac{1}{2\beta}\left[\log\left(e^{\beta(\lambda\, z^2 - \alpha)}\right) - \log\left(1 + e^{\beta(\lambda\, z^2 - \alpha)}\right)\right] \\ \rho'(z) &= \frac{\lambda z}{1 + e^{\beta(\lambda\, z^2 - \alpha)}} \end{aligned} \tag{21}$$

The resistive fuse is more robust than Mead's original circuit. Note in Fig. 8D that the I-V curve (influence function) for the resistive fuse redescends so that large voltages produce negligible currents. That is, outliers have no influence.

References

[1] M J Black and A Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19:57–92, 1996.

[2] A Blake and A Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, 1987.

[3] S Geman and D Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[4] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley and Sons, New York, NY, 1986.

[5] J G Harris, C Koch, and J Luo. A two-dimensional analog VLSI circuit for detection discontinuities in early vision. *Science*, 248:1209–1211, 1990.

[6] J G Harris, C Koch, E Staats, and J Luo. Analog hardware for detecting discontinuities in early vision. *International Journal of Computer Vision*, 4:211–223, 1990.

[7] P J Huber. *Robust Statics*. John Wiley and Sons, New York, 1981.

[8] J Marroquin, S Mitter, and T Poggio. Probabilistic solution of ill-posed problems in computational vision. *J Am Stat Assoc*, 82:76–89, 1987.

[9] C Mead. *Analog VLSI and Neural Systems*. Addison-Wesley, New York, 1989.

[10] T Poggio, V Torre, and C Koch. Computational vision and regularization theory. *Nature*, 317(6035):314–319, 1985.

[11] R Szeliski. Fast surface interpolation using heirarchical basis functions. *IEEE Pattern Analysis and Machine Intelligence*, 12:513–528, 1990.

[12] R Szeliski. Surface modeling with oriented particle systems. *Computer Graphics, ACM SIGGRAPH Proceedings*, 26:185–194, 1992.

[13] D Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Pattern Analysis and Machine Intelligence*, 8(4):413–424, 1986.