# Notes on Robust Estimation[1]

David J. Fleet  Allan Jepson

March 30, 2005

## 1 Robust Estimataion.

The field of robust statistics [3, 4] 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."

## 2 Robust Regularization.

**Denoising (revisited).** Let us again consider the problem of denoising a one-dimensional input signal. A generalization of the previous regularization formulation is to minimize the objective function constraints:

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

where $\rho$ is an error norm and $\sigma_d$ and $\sigma_s$ are scale parameters. The least-squares formulation discussed in the previous regularization notes is the special case in which $\rho(z, \sigma) = z^2$. But the least-squares approach is notoriously sensitive to outliers; the problem being that outliers contribute too much to the overall solution.

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

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

For least-squares estimation, the influence of outliers increases linearly and without bound (Fig. 1).

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:

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

The Lorentzian error norm ($\rho$) is plotted along with its influence function ($\rho'$) in Fig. 1. Examination of the $\rho'$-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 [3] for other options).

The least-squares regularization formulation assumes that the signal is equally smooth throughout. Figure 2 shows a "step" input with added noise. Linear regularization removes the noise appropriately, but it also blurs the step edge. Robust regularization smooths the smooth part while keeping the step sharp.

[1] Based on a handout by David Fleet and David Heeger at Stanford University.

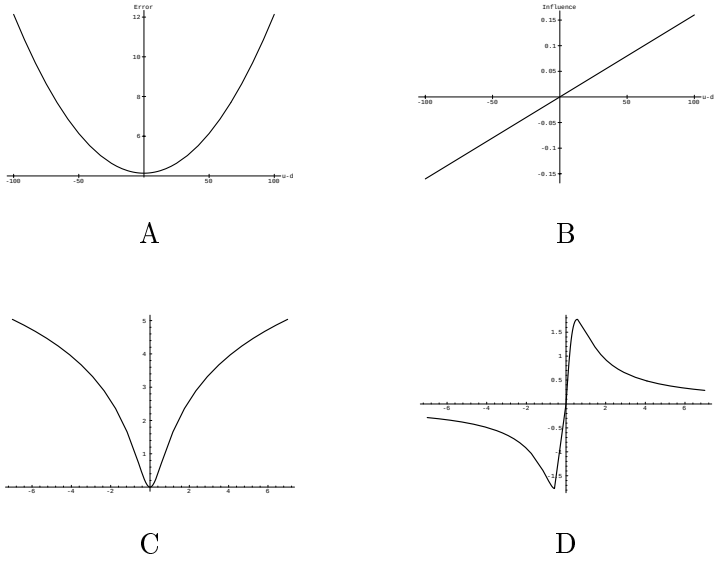


Figure 1: **A:** Least-square error ($\rho$) function. **B:** Least-squares influence ($\rho'$) function. **C:** Lorentzian error ($\rho$) function. **D:** Lorentzian influence ($\rho'$) function.

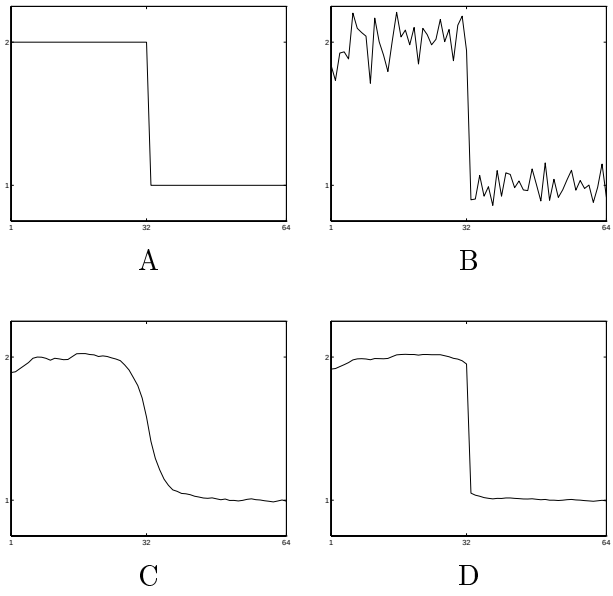


Figure 2: **A:** Original step signal. **B:** Noisy step signal. **C:** Regularization result using quadratic error norm for both the data constraint and the smoothness constraint. **D:** Regularization result using a quadratic norm for the data constraint, but a robust Lorentzian error norm for the smoothness constraint.

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:

$$z_m^{(i+1)} = z_m^{(i)} - \frac{f'(z_m^{(i)})}{f''(z_m^{(i)})}, \tag{3}$$

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. 1, we write its first and second derivatives with respect to $v[k]$:

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

For example, using a quadratic error function (least-squares), substituting for the derivatives in Eq. 3 gives:

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

which is identical to the Jacobi iteration introduced in the linear regularization notes.

Robust formulations typically result in nonconvex optimization problems. To find a globally optimal solution when the objective function is nonconvex we choose a robust $\rho$-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].

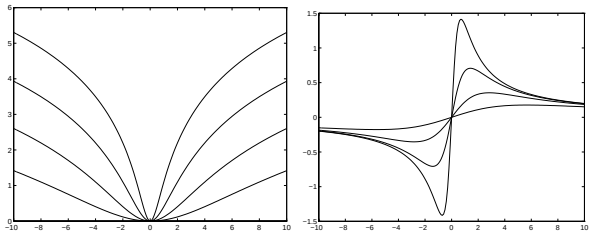

Figure 3: Lorentzian error functions and (left) influence functions (right) for several values of $\sigma$ (1/2, 1, 2, and 4).

For example, Fig. 3 shows a family of Lorentzian error and influence functions for several values of $\sigma$. For larger values of $\sigma$, 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 $\sigma$ is very large. To compute the result in Fig. 2D, we 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 $\sigma$ 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, $\tau$, can be considered outliers. The point where the influence of outliers first begins to decrease occurs when the second derivative of the $\rho$-function is zero. For the Lorentzian, the second derivative,

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

is zero when $z = \pm\sqrt{2}\,\sigma$. If the maximum expected (absolute) residual is $\tau$, then choosing $\sigma = \tau/\sqrt{2}$ will result in a convex optimization problem. A similar treatment applies to other robust $\rho$-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 $|z| \geq \sqrt{2}\,\sigma$.

Figure 4 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.** In linear regularization, 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. 1), the robust first-order smoothness constraints are minimized by a *piecewise* constant signal (see Figs. 2 and 4).

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.

Figure 4: (Left) Input Image. (Middle) Regularization result using a robust Lorentzian error norm for the smoothness constraint. (Right) Edges obtained at pixels which are indicated to be outliers according to the Lorentzian error norm.

**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:

$$E(v,l) = \sum_{x=1}^{N} (v[x] - u[x])^2 + \lambda \sum_{x=1}^{N} [(v[x+1] - v[x])\, l[x] + \Psi(l[x])], \tag{4}$$

where $l[x]$ takes on values $0 \leq l[x] \leq 1$. The line process indicates the presence ($l[x] \rightarrow 0$) or absence ($l[x] \rightarrow 1$) 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 $\Psi$. An example penalty function is: $\Psi(l) = (1 - l)$. Minimizing Eq. 4 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. 4, one can derive a robust error norm for the smoothness constraint in Eq. 1, and vice versa, so that the two optimization problems will be identical. For example, using the penalty function $\Psi(l) = l - 1 - \log(l)$ in Eq. 4 is the same as using a Lorentzian error norm for the smoothness constraint in Eq. 1.

**Regularization of Vector Fields.** Imagine that a user has interactively specified the spatial displacement for an image warp, but only for a handful of pixels. This is commonly done for image morphing. Regularization can be used to compute a dense spatial warp from the sparse and scattered specification. Keeping the same notation as before, $\vec{u}$ is now the sparse specification of the spatial displacements and $\vec{v}$ is the desired dense spatial displacements. Both are vector-valued so we use vector distance in the objective function

$$E(\vec{v}) = \sum_{[x,y] \in P} \|\vec{v}[x,y] - \vec{u}[x,y]\|^2 + \lambda \sum_{\text{all } [x,y]} \|4\vec{v}[x,y] - \vec{v}[x-1,y] - \vec{v}[x,y-1] - \vec{v}[x+1,y] - \vec{v}[x,y+1]\|^2. \tag{5}$$

This objective function can again be minimized with simple iterative techniques.

A similar process could also be used in painterly rendering. Here a given input image is to be copied in painterly style. The spatial orientation of the individual paint strokes is given by some orientation field. One way to automatically generate constraints on a desirable orientation field is to use strong edges observed in the original image (say, using the Canny edge detector). The orientation of these edges can be represented by a 2D unit vector $\vec{u}[x, y]$. However, these strong edges are only observed at a small subset $P$ of the image pixels. What orientation should we use for paint strokes away from these edges? We could again minimze Eq. 5 to approximately interpolate the observed edge orientations at the pixels in $P$ to a smooth orientation field $\vec{v}[x, y]$ across the full image. Alternatively, a robust estimator could be used on the spatial term in Eq. 5 to allow for spatial discontinuities in the orientation field.

One issue that arises specifically for orientation fields is that we may not want to distinguish the sign of the gradient at an edge. For example, we may wish to consider a vertical edge where the $x$-component of the image gradient is positive to be the same orientation as another vertical edge having a negative $x$-component. If we used the angle of the image gradient to represent the orientation of these two edges, then these angles differ by $\pm\pi$.

A standard trick that is useful here is to use a doubled angle representation for edge orientation. That is, we encode edges with a tangent direction $\vec{t}[x, y] = (\cos(\theta[x, y]), \sin(\theta[x, y]))^T$ in terms of the doubled angle $2\theta[x, y]$. So $\vec{t}[x, y]$ is represented by the vector $\vec{u}[x, y] = (\cos(2\theta[x, y]), \sin(2\theta[x, y]))^T$. In this case, image gradients with the opposite signs have angles $\theta$ which differ by $\pm\pi$, so their doubled angles differ by $\pm 2\pi$. As a result, they are represented by identical vectors $\vec{u}[x, y]$. The orientation field can then be filled in by minimizing an objective function such as the one in Eq. 5, or a robust version of it. Once the vector field $\vec{v}[x, y]$ is computed, the orientation to be used for paint strokes can be set to be half the angle of $\vec{v}[x, y]$.

## 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] 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.

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