Robust Estimation

Introduction. A common computational problem in vision is to estimate the parameters of a model from image data.

Examples of parameterized models to be fit to image data include lines and ellipses, camera calibration models, image motion models, 3D planar regions, 3D models, and human face models.

Key Difficulties:

- The models must be fit to noisy image data.
- Initial guesses for the models must be generated automatically.
- Multiple occurrences of the models are often represented in the data. But the number and types of models are typically unknown a priori, and therefore must be determined from the data.
- The data typically contains (structured) outliers, i.e., observations that do not belong to the model being fitted. These must somehow be ignored during the model fitting.

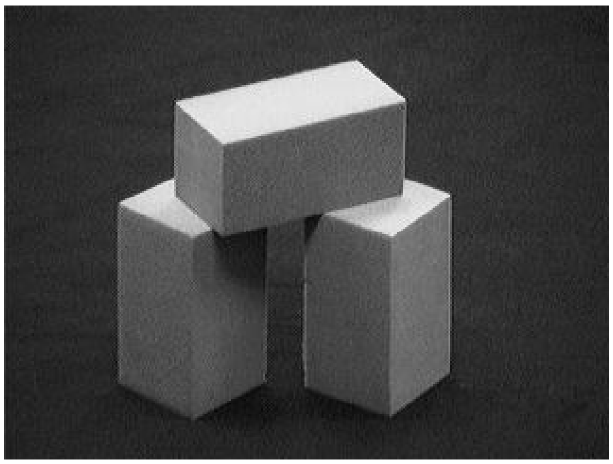
Reading on Estimation: See Chapter 15 of the text (skip section 15.6 for now).

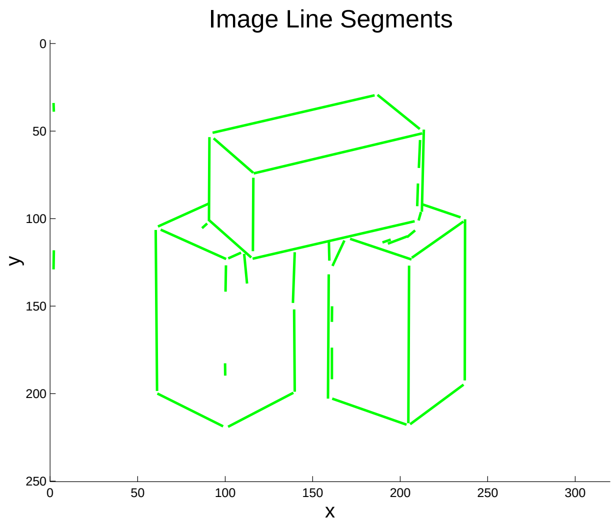
Working Example Problem: Fitting Lines

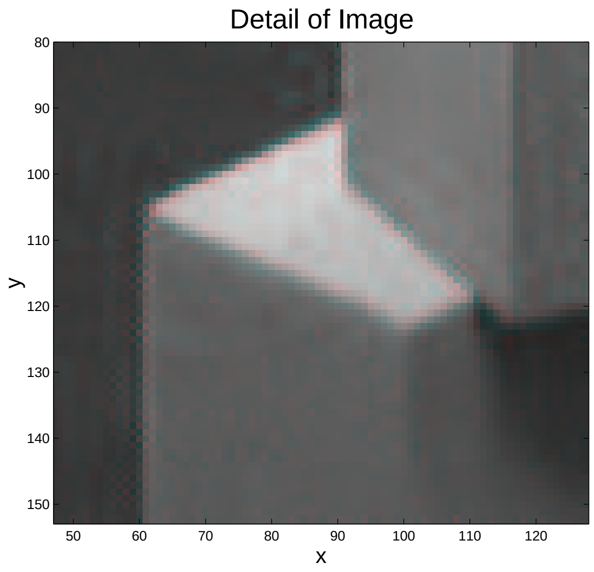
Example Problem: Find the best fitting line(s) to a set of image edgel positions,

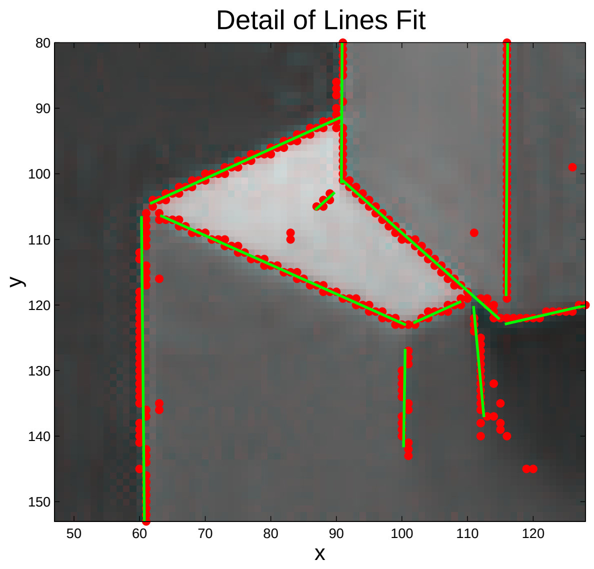
$$D = \{\vec{x}_k\}_{k=1}^{K}.$$

For simplicity, we ignore edgel strengths and orientations here.









Robust line estimator output. The Canny edgel positions $\{\vec{x}_k\}$ are marked (red dots), along with the fitted line segments (green).

Current Goals

We use the example of estimating image lines to:

- Introduce robust M-estimation and show how it deals with outliers.
- Discuss leverage points.
- Introduce methods for generating initial guesses.
- Introduce issues in model selection.
  - We have more to say about this difficult issue later in this course. Here we briefly consider selecting the number of models (i.e., lines), but not the model type (eg. lines versus curves).
- Discuss the general types of errors to be expected.

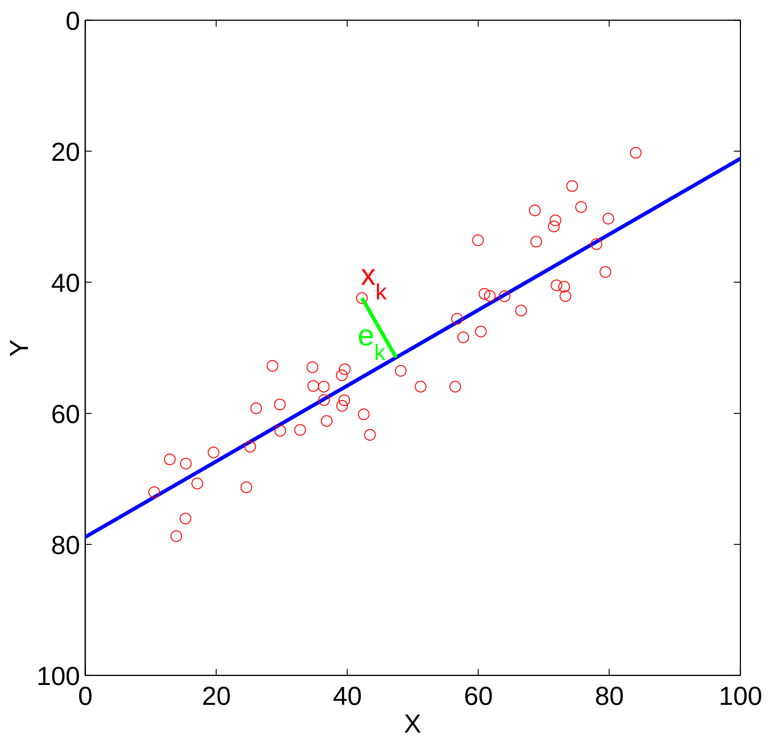
Perpendicular Error

The equation for points $\vec{x}$ on one infinite line L is,

$$\vec{n} \cdot \vec{x} + c = 0.$$

Here we normalize $||\vec{n}|| = 1$. The error in an observed point $\vec{x}_k$ (relative to the line L) is defined to be the perpendicular distance

$$e(\vec{x}_k; \vec{n}, c) \equiv \vec{n} \cdot \vec{x}_k + c.$$



Note that the error perpendicular to the line is used. This is different from linear regression models where the error is only in the y component.

Least Squares Problem

Given a set of edgel positions $D = \{\vec{x}_k\}_{k=1}^K$, consider estimating the line parameters $(\vec{n}, c)$ by minimizing the squared error

$$\mathcal{O}(\vec{n}, c) \equiv \sum_{k=1}^{K} (\vec{n}^{\,T}\vec{x}_k + c)^2, \quad \text{for } ||\vec{n}|| = 1. \tag{1}$$

By the theory of Lagrange multipliers, the solution must satisfy

$$\frac{\partial \mathcal{L}}{\partial(\vec{n}, c, \lambda)}(\vec{n}, c, \lambda) = \vec{0}, \quad \text{for } \mathcal{L}(\vec{n}, c, \lambda) \equiv \mathcal{O}(\vec{n}, c) - \lambda(||\vec{n}||^2 - 1).$$

The derivatives of $\mathcal{L}$ with respect to $\vec{n}$, c, and λ give (respectively):

$$\begin{aligned} &\sum_{k=1}^{K} 2(\vec{n}^{\,T}\vec{x}_k + c)\vec{x}_k - 2\lambda\vec{n} = \vec{0}, \\ &\sum_{k=1}^{K} 2(\vec{n}^{\,T}\vec{x}_k + c) = 0, \\ &||\vec{n}||^2 - 1 = 0. \end{aligned} \tag{2}$$

These can be simplified using

$$\vec{m} = \frac{1}{K}\sum_{k=1}^{K} \vec{x}_k, \quad C = \frac{1}{K}\sum_{k=1}^{K} (\vec{x}_k - \vec{m})(\vec{x}_k - \vec{m})^T,$$

namely the mean and covariance of the data.

Least Squares Solution

From (2) it follows that the least squares solution $(\vec{n}, c)$ must satisfy

$$\begin{aligned} C\vec{n} &= \mu\vec{n}, \quad ||n|| = 1, \\ c &= -\vec{n}^T\vec{m}. \end{aligned} \tag{3}$$

That is, $\vec{n}$ must be an eigenvector of C and c must be chosen such that the estimated line passes through the mean of the points.

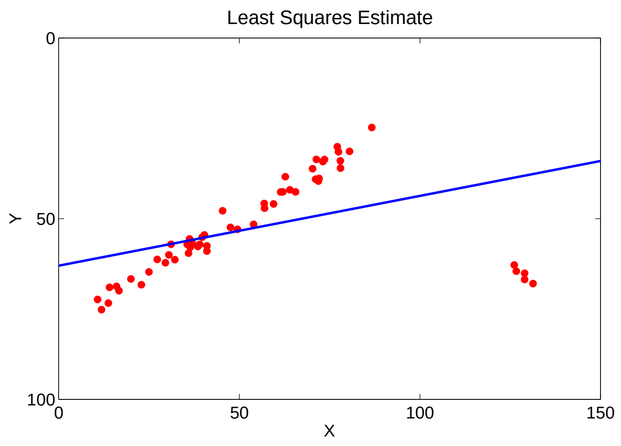
In order for the solution to be a local minimum, it can be shown that $\vec{n}$ must be the eigenvector for the *minimum* eigenvalue of C.

The solution is therefore unique and easy to compute. No initial guess is required.

Least Squares and Outliers

Unfortunately, least squares solutions are sensitive to outliers in the data.

For example, consider the set of edgel point data $\{\vec{x}_k\}_{k=1}^{K}$ (red dots).



The least squares estimate (blue line, above) is strongly influenced by the small cluster of outliers.

This approach is therefore not directly suitable for fitting data with outliers or when multiple solutions are expected.

Hough Transform

The Hough transform (HT) is an early method for dealing with outliers and multiple solutions in parameter estimation problems.

For our working example, image lines $\vec{n}^T(\theta)\vec{x} + c = 0$ are parameterized by θ and c, with $\vec{n}(\theta) = (\cos(\theta), \sin(\theta))^T$.

A discrete voting space is formed by quantizing $\theta \in [0, \pi)$ and c. For each edgel $\vec{x}_k$, and each discrete θ_i, we add one vote to the bin (θ_i, c_j) which has minimal absolute error, that is

$$\vec{n}^T(\theta_i)\vec{x}_k + c_j \in [-\delta c/2, \delta c/2). \tag{4}$$

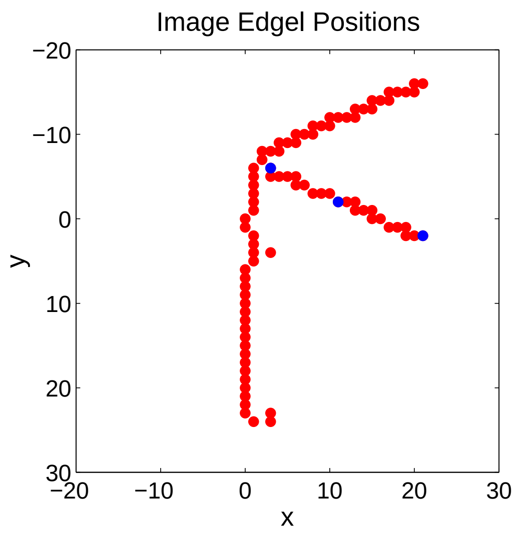
Here δc is the bin spacing for c.

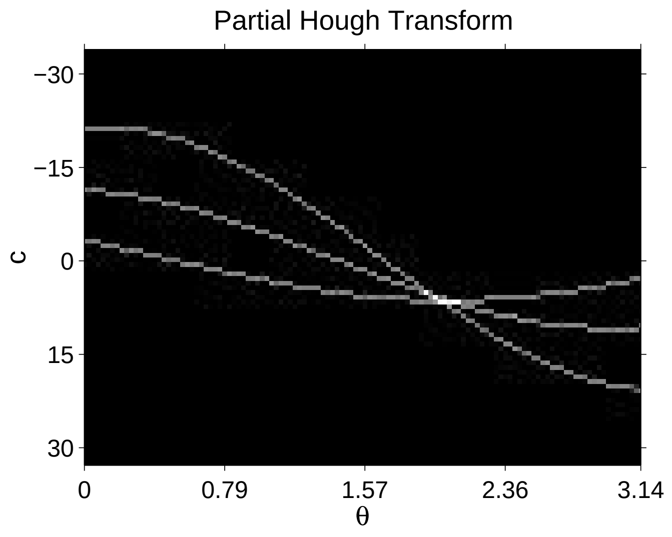
Eqn. (4) dictates that each edgel $\vec{x}_k$ votes for a sinusoidal curve in the Hough space (see example below).

The accumulation of votes for all the edgels $\vec{x}_k$ provides the Hough transform H.

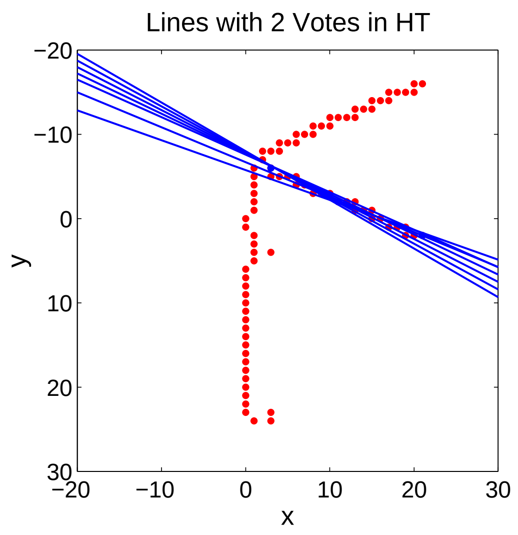
Hough Example

For example, the Hough transform for only the three blue edgels below is shown on the right.





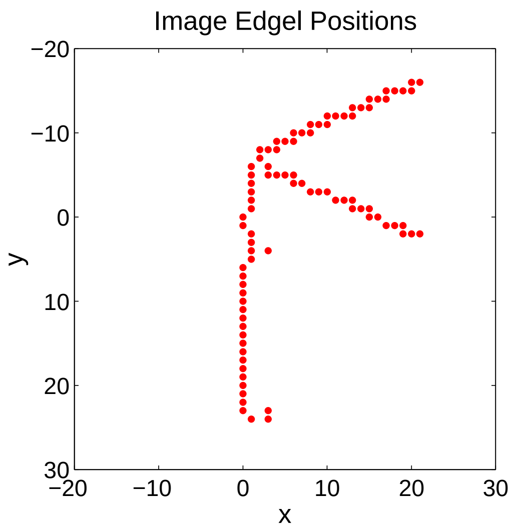
The maximum number of votes here is 2, not 3, since the three edgels are not precisely colinear. The centers of the bins with 2 votes provide the parameters for blue lines below.

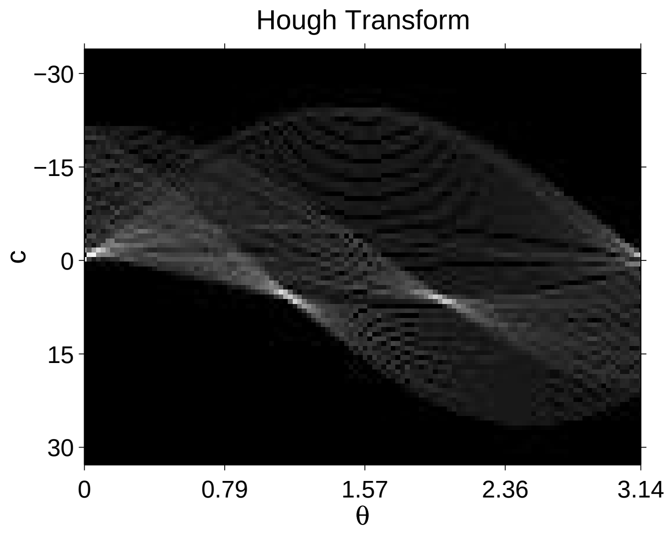


Increasing the bin size δc would allow one (or more) bins to have votes from all three edgels. In practice, selecting appropriate bin sizes for the HT can be tricky.

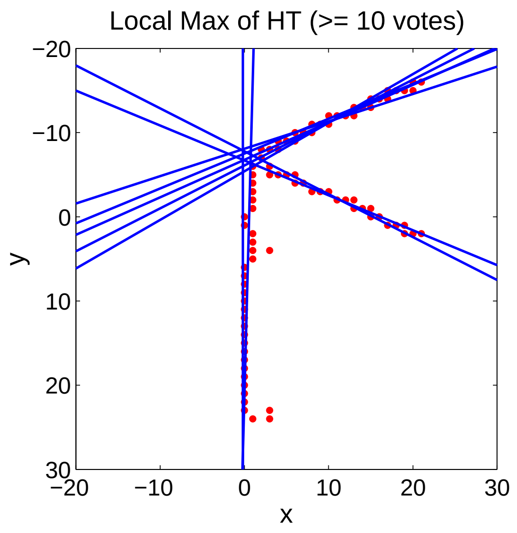
Hough Example (Cont.)

The Hough transform for all the edgels in the previous example is shown on the right. Note the symmetry $H(\theta_i, c_j) = H(\theta_i + \pi, -c_j)$, which is due to the symmetry of the error $e = \vec{n}^T(\theta)\vec{x} + c$.





Three strong peaks are visible in the HT, corresponding roughly to the three line segments apparent in the data. (Enlarge the neighbourhood of these peaks when viewing an electronic copy.)



The blue lines result from local non-max suppression of the HT above, then thresholding the result at 10 or more votes. The results are sensitive to the particular values used for this threshold and the bin sizes. Appropriate values for these parameters often depend on the image data.

Hough Transform Summary

The Hough transform is a reasonable approach for simple data, for which clear peaks can be expected. However,

- **Goldilocks' problem:** It is difficult to set appropriate bin sizes:
  - **Too large:** Poor resolution of parameters.
  - **Too small:** Peaks in HT broken into pieces, each with fewer votes.
  - **Just right:** Isolated peaks at appropriate parameter values.
- Poor detector performance (i.e. the trade-off between false positives and false negatives). In noisy and/or cluttered examples many extraneous bins can have vote counts comparable to, or larger than, the counts for the desired models.
- In practice, fractional votes for neighbouring Hough bins are also included to help smooth the HT and reduce discretization artifacts.
- The number of bins grows exponentially with the dimension of the unkown parameters (eg. n^d for n bins in each of d dimentions).

For more complex data sets, a much better alternative is to use robust M-estimation, which we discuss next.

Robust M-estimation

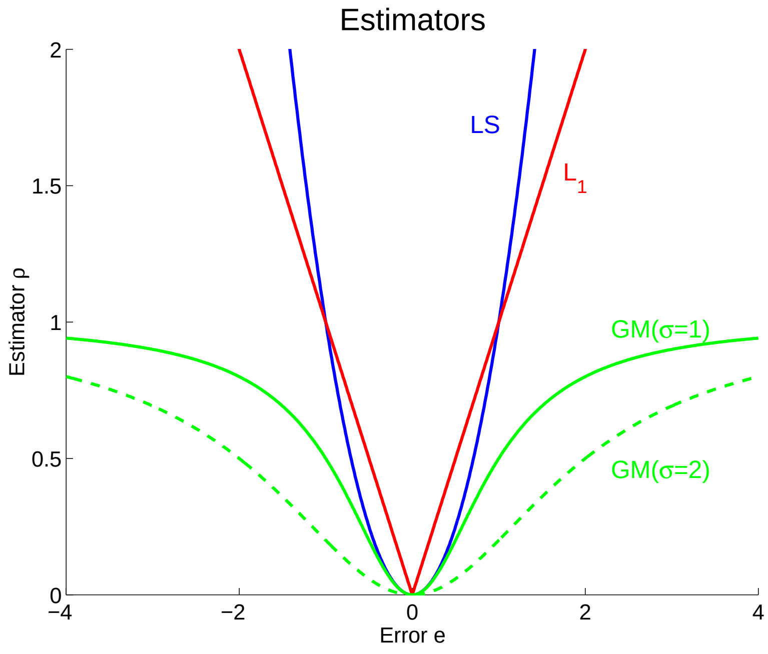
We seek line parameters $(\vec{n}, c)$ which (locally) solve the minimization problem

$$\min \mathcal{O}(\vec{n}, c), \quad \text{for } ||\vec{n}|| = 1, \tag{5}$$

$$\mathcal{O}(\vec{n}, c) \equiv \sum_{k=1}^{K} \rho(e(\vec{x}_k\,;\,\vec{n}, c)). \tag{6}$$

Here $\rho(e)$ is the "estimator", which provides a cost for any given error e. Common choices include:

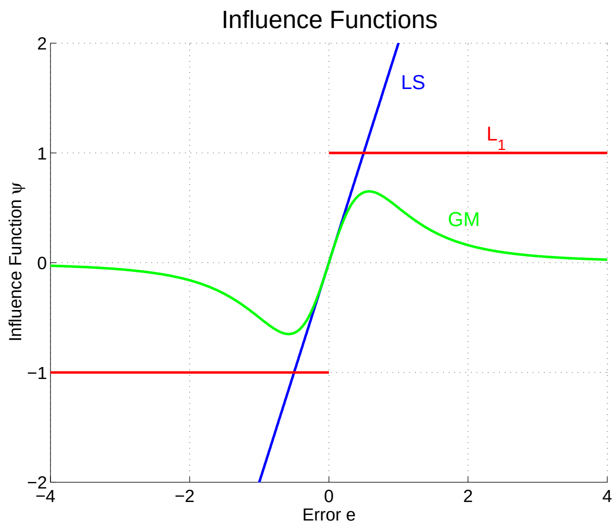
$\rho(e) = e^2,$	Least squares (LS) estimator,
$\rho(e) = \lvert e \rvert,$	L_1 estimator,
$\rho(e) = \dfrac{e^2}{\sigma^2 + e^2},$	Geman-McLure (GM) estimator.

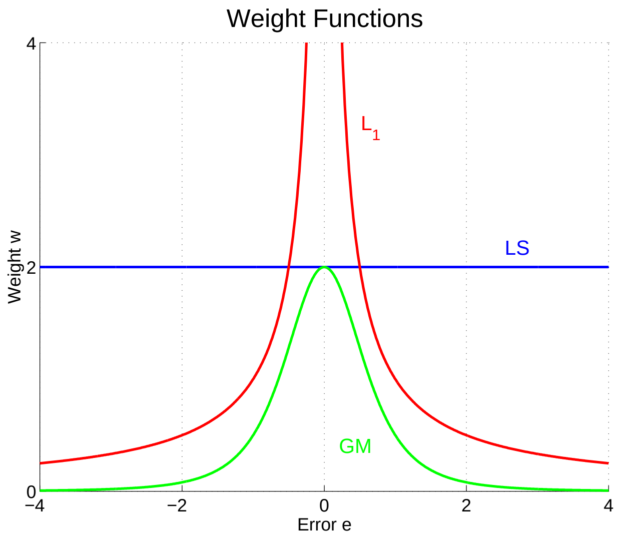


Properties of Estimators

Definition: Given the estimator $\rho(e)$ we define the functions:

$$\psi(e) = \frac{d\rho}{de}(e), \qquad \text{influence function,}$$
$$w(e) = \frac{1}{e}\frac{d\rho}{de}(e), \qquad \text{weight function.}$$



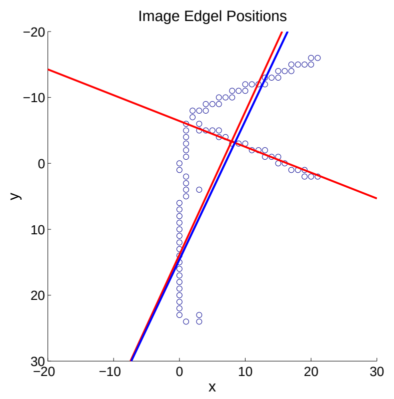


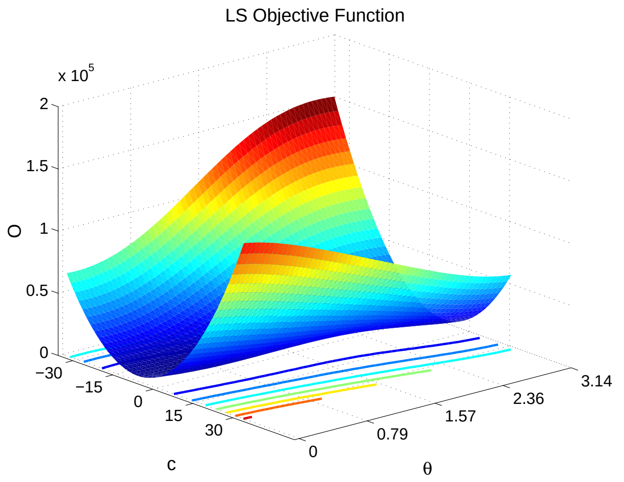
The influence function $\psi(e)$ describes the sensitivity of the overall estimate $(\vec{n}, c)$ on data with error e. Note that, unlike LS, the influence functions for L_1 and GM are bounded. Moreover, GM is a 'redescending estimator', that is, $\psi(e) \rightarrow 0$ as $|e| \rightarrow \infty$.

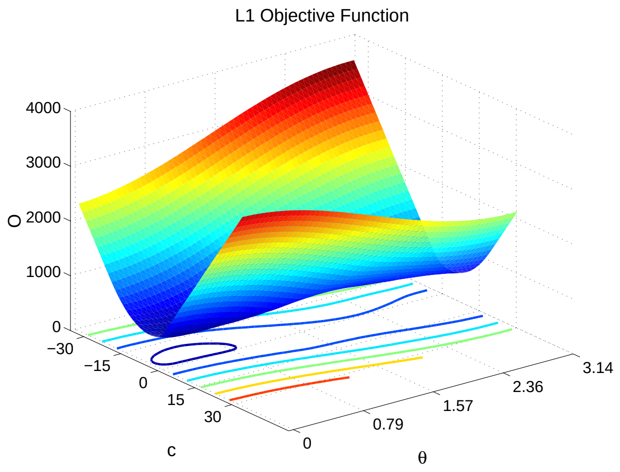
As we show below, $w(e)$ provides the weights in the iteratively reweighted least squares algorithm used to solve for the unknown line parameters $(\vec{n}, c)$. Note that $w(e) \equiv 2$ (i.e., constant) for LS.

Objective Function Examples: LS and L$_1$

Given a data set $\{\vec{x}_k\}_{k=1}^{K}$, we can plot the objective function $\mathcal{O}(\vec{n}(\theta), c)$ for $\vec{n}(\theta) = (\cos(\theta), \sin(\theta))^T$. Note that, since $\vec{n}(\theta + \pi) = -\vec{n}(\theta)$, the objective function is periodic in θ with $\mathcal{O}(\vec{n}(\theta + \pi), c) = \mathcal{O}(\vec{n}(\theta), -c)$.





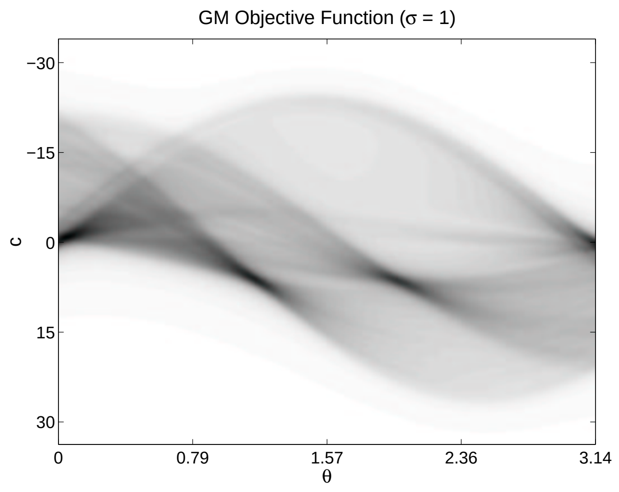


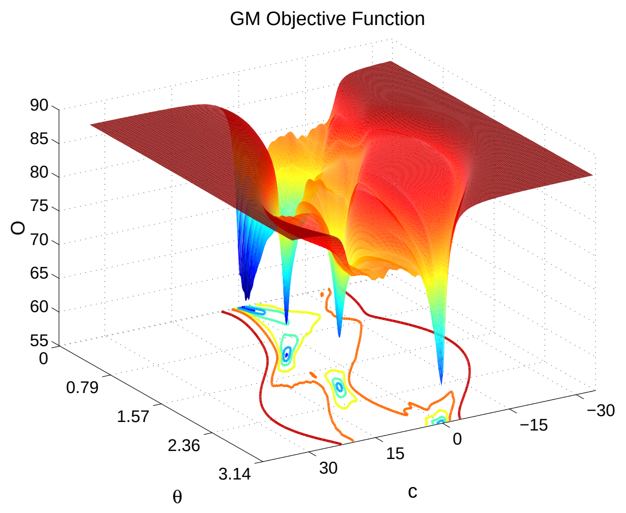
The LS estimator always provides a *unique* local minimum (blue line in top figure). The L1 estimator can have multiple local minima. In this case, there are two local minima (red lines in top figure).

These surfaces (and the fits above) indicate the significant influence of data with large errors.

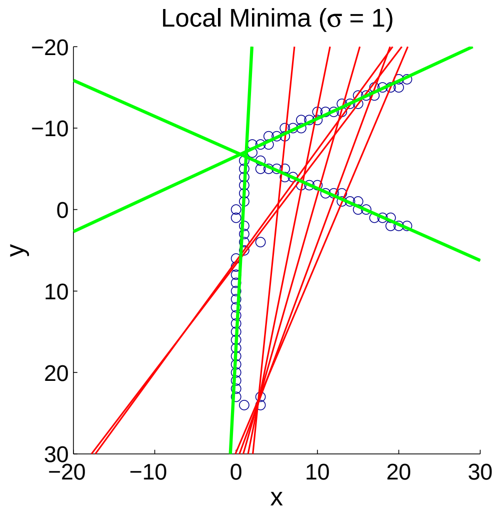
Objective Function Examples: GM($\sigma = 1$)

In contrast, for the GM estimator the influence of a data point with large error (essentially) vanishes. For the same data as before:





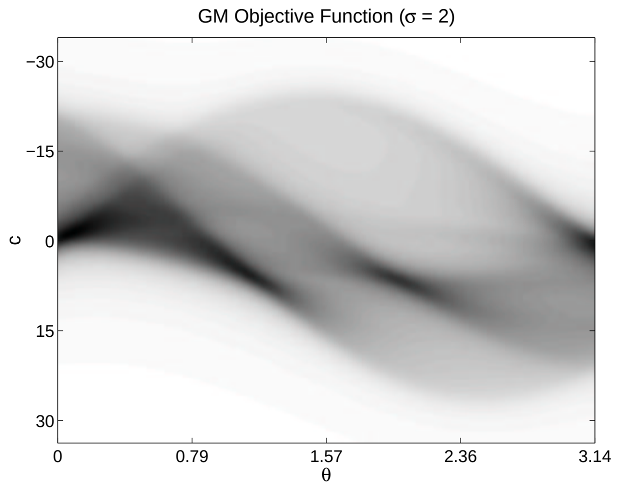
For the GM estimator, the objective function exhibits three strong local minima (see the green lines below). The outer pair of the four deep valleys apparent above are actually parts of the same valley, due to periodicity in θ. There are many weaker local minima (red lines below).

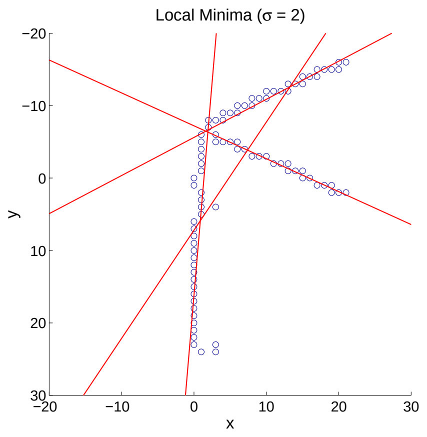


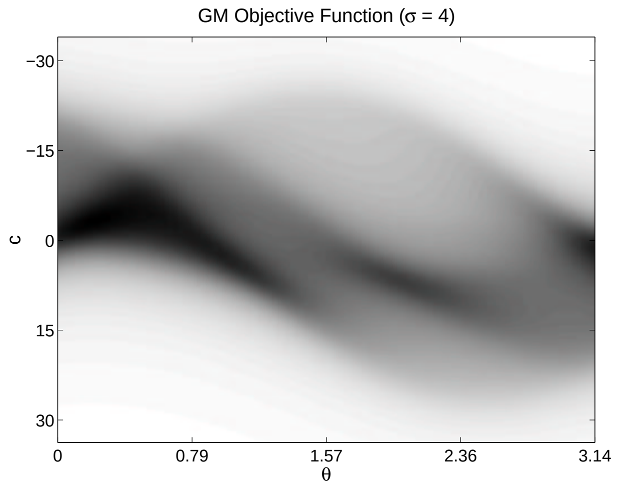
We show only those local minima for which $\mathcal{O} < K - 10$, where K is the number of data points (i.e. the maximum possible value for $\mathcal{O}$).

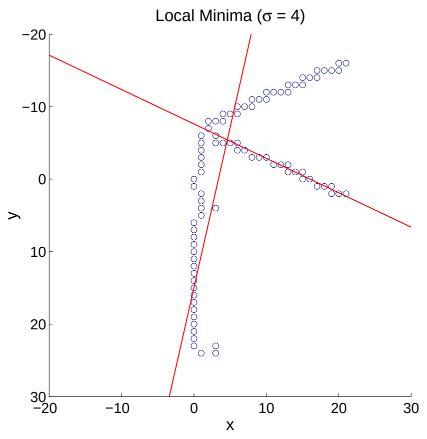
Objective Function Examples: GM, Varying σ

Increasing σ in the GM-estimator smooths the objective function:









- Note there are fewer local minima for larger σ. (We show *all* the local minimum beyond minor fluctuations.)
- As σ is increased, outliers receive greater weights, and therefore the bias in the estimates often increases (eg. bottom right plot).
- As $\sigma \to \infty$ the objective function can be shown to approximate the one for least squares (with a unique solution).

Robust Objective Function and the Hough Transform

The robust objective function $\mathcal{O}(\theta, c)$ is closely related to a smoothed Hough transform.

Theorem. Suppose $\rho(e)$ is a smooth redescending estimator such as GM, with $\rho(e) \to 1$ as $|e| \to \infty$. Let $H(\theta_i, c_j)$ be the Hough transform of K edgels using small bin sizes $\delta\theta$ and δc. Consider the discrete 1D filter kernel $f(c_j)$ obtained by sampling $1 - \rho(c)$ at $c_j = j\delta c$, $j = 0, \pm 1, \pm 2, \ldots$. Define

$$S(\theta_i, c_j) = (f *_2 H)(\theta_i, c_j).$$

Here $*_2$ denotes convolution with respect to the second argument. So $S(\theta_i, c_j)$ is simply $H(\theta_i, c_j)$ blurred in the c-direction by $\rho(c_j)$. Then

$$\mathcal{O}(\theta_i, c_j) = K - S(\theta_i, c_j) + O(\delta\theta + \delta c),$$

as $\delta\theta, \delta c \to 0$. That is, the blurred Hough transform S approximates $\mathcal{O}$, up to a sign change and additive constant.

The smoothness of $\mathcal{O}(\theta, c)$ is useful for locating local minima precisely. Indeed, as mentioned above, the HT is often smoothed in practice. Moreover, on the next few pages we show how to find local minima **without** densely sampling the objective function, which can provide a considerable computational savings over HT.

Weighted Mean and Covariance

We wish to find local minima of $\mathcal{O}(\vec{n}, c)$ without densely sampling the parameter space. For estimators with bounded weight functions, the weighted mean and covariance of the data are useful, namely

$$\vec{m} = \frac{1}{s} \sum_{k=1}^{K} w(e_k) \vec{x}_k, \tag{7}$$

$$C = \frac{1}{s} \sum_{k=1}^{K} w(e_k) (\vec{x}_k - \vec{m})(\vec{x}_k - \vec{m})^T, \tag{8}$$

where $s \equiv \sum_{k=1}^{K} w(e_k)$ and $e_k \equiv \vec{n}^{\,T} \vec{x}_k + c$. We show in the subsequent notes that a necessary condition for $(\vec{n}, c)$ to be a solution of (5) is that

$$C\vec{n} = \mu \vec{n}, \tag{9}$$

$$c = -\vec{n}^{\,T} \vec{m}, \quad ||\vec{n}|| = 1. \tag{10}$$

where μ is the minimum eigenvalue of the 2×2 covariance matrix C. Properties of the solution:

- C typically depends on $(\vec{n}, c)$ through the weights $w(e_k)$ (as does $\vec{m}$), and therefore (9) is a *nonlinear* eigenvalue problem for $\vec{n}$.
- For the LS estimator, the weights are constant (i.e., $w_k = 2$) and (9) is the *linear* eigenvalue problem we saw in (3).
- Eqn. (10) states that the fitted line must pass through the weighted mean $\vec{m}$ (i.e., $\vec{n}^{\,T} \vec{m} + c = 0$).

Notes: Minimizing the Objective Function

Suppose $\rho(e)$ is an estimator with a bounded weight function $w(e)$. We wish to minimize $\mathcal{O}(\vec{n}, c)$, subject to the constraint $||\vec{n}|| = 1$. A standard approach for such a constrained optimization problem is to introduce a Lagrange multiplier λ, and consider

$$\mathcal{L}(\vec{n}, c, \lambda) = \mathcal{O}(\vec{n}, c) - \lambda(||\vec{n}||^2 - 1).$$

A necessary condition for $(\vec{n}, c)$ is that this Lagrangian $\mathcal{L}$ has a stationary point at $(\vec{n}, c, \lambda)$, that is

$$\frac{\partial \mathcal{L}}{\partial(\vec{n}, c, \lambda)}(\vec{n}, c, \lambda) = \vec{0}.$$

For $\mathcal{O}(\vec{n}, c)$ as in (5) the derivatives of $\mathcal{L}$ with respect to $\vec{n}$, c, and λ give (respectively):

$$\begin{aligned} &sA\vec{n} + s\vec{m}c - 2\lambda\vec{n} = \vec{0}, \\ &s\vec{m}^T\vec{n} + sc = 0, \\ &||\vec{n}||^2 - 1 = 0. \end{aligned} \tag{11}$$

Here $s \equiv \sum_{k=1}^{K} w(e_k)$ is the sum of the weights for errors $e_k \equiv \vec{n}^T\vec{x}_k + c$, $\vec{m} \equiv \frac{1}{s}\sum_{k=1}^{K} w(e_k)\vec{x}_k$ is the weighted mean of the data $\{\vec{x}_k\}_{k=1}^{K}$, and $A \equiv \frac{1}{s}\sum_{k=1}^{K} w(e_k)\vec{x}_k\vec{x}_k^T$ is the weighted correlation matrix. Solving the second equation for c gives $c = -\vec{m}^T\vec{n}$. Substituting this into the first equation gives the *(nonlinear) eigenvalue problem* for $\vec{n}$,

$$\frac{s}{2}(A - \vec{m}\vec{m}^T)\vec{n} = \lambda\vec{n}. \tag{12}$$

A short calculation shows that the weighted covariance matrix in (8) satisfies $C = A - \vec{m}\vec{m}^T$. Therefore (12) takes the desired form, $C\vec{n} = \mu\vec{n}$, for $\mu = 2\lambda/s$. Finally, it can be shown that the objective function is locally minimized only when $\vec{n}$ is chosen to be the eigenvector for the *minimum* eigenvalue of C.

Iteratively Reweighted Least Squares Algorithm

Given an initial guess for the line parameters, $(\vec{n}_0, c_0)$, we can update it as follows:

- Compute the weights $w_k = w(\vec{n}_0^T \vec{x}_k + c_0)$.
- Compute $\vec{m}$ and C as in (7) and (8).
- Set $\vec{n}$ to be the eigenvector for the minimum eigenvalue of C.
- Set $c = -\vec{n}^T \vec{m}$.

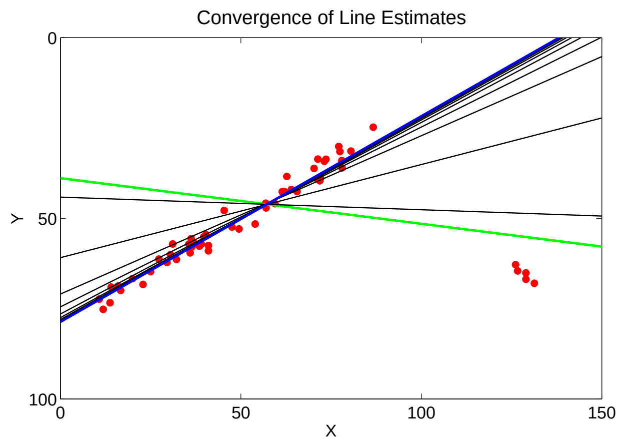
Note that the computed $(\vec{n}, c)$ is not necessarily a local minima of (5), since $\vec{m}$ and C were computed using $(\vec{n}_0, c_0)$ instead of $(\vec{n}, c)$.

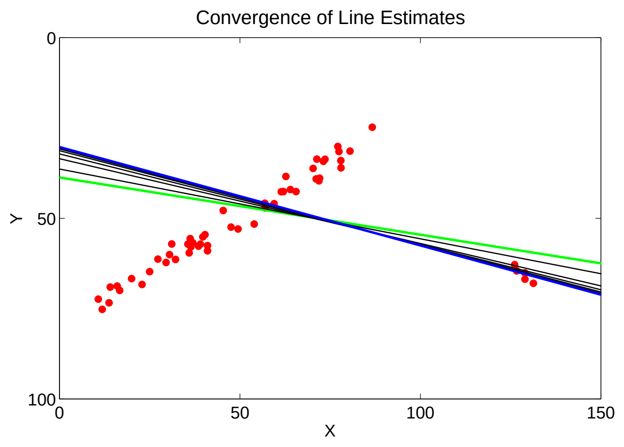
Resetting $(\vec{n}_0, c_0)$ to be the newly computed $(\vec{n}, c)$, we iterate the above steps until the update of both $\vec{n}$ and c are small.

This is known as the iteratively reweighted least squares (IRLS) algorithm. See the Matlab tutorial `robustDemo.m` (available from the course homepage) for a demonstration.

Example Results

For the GM estimator (with σ equal to the standard deviation of the noise for the edgels near the dominant line) the iterations of the IRLS algorithm are shown below (black lines). The initial guesses (green) and the converged states (blue) are also shown.





The solution on the left indicates the outliers have been rejected, while the iterations on the right show the algorithm converging to one of several undesirable local minimum. How can we avoid these?

Analysis of Influence

To understand these extraneous local minima it is useful to first consider the influence of outliers in detail.

Given a solution $(\vec{n}_0, c_0)$ of (5), we consider the effect of one additional outlier $\vec{x}_{K+1}$. We wish to estimate the perturbation to the solution caused by this outlier. The size of this perturbation will determine the *influence* of that outlier.

To simplify the analysis, we freeze the weights for the original data, $w_k = w(\vec{n}_0^T \vec{x}_k + c_0)$ for $k = 1, \ldots, K$, and set $w_{K+1} = w(\vec{n}_0^T \vec{x}_{K+1} + c_0)$. With all these weights frozen, we seek the solution of the weighted least squares problem

$$\min \mathcal{O}_{WLS}(\vec{n}, c, \epsilon), \quad \text{for } ||\vec{n}|| = 1, \tag{13}$$

$$\mathcal{O}_{WLS}(\vec{n}, c, \epsilon) \equiv \left[\sum_{k=1}^{K} \frac{w_k}{2} (\vec{n}^T \vec{x}_k + c)^2 \right] + \epsilon \frac{w_{K+1}}{2} (\vec{n}^T \vec{x}_{K+1} + c)^2. \tag{14}$$

For $\epsilon = 0$ we recover the equations (9) and (10) for the original solution $(\vec{n}_0, c_0)$, while for $\epsilon = 1$ the solution is the first update of the IRLS algorithm when it is given the additional data point $\vec{x}_{K+1}$ and the initial guess $(\vec{n}_0, c_0)$.

Influence of Outliers on Line Parameters

Theorem. Let $\psi_{K+1} = \psi(\vec{n}_0^T \vec{x}_{K+1} + c_0)$ be the influence function evaluated at $\vec{x}_{K+1}$, and $\vec{t}_0$ be a unit vector tangent to the initial line (i.e., $\vec{t}_0 \perp \vec{n}_0$). Then the optimal solution $(\vec{n}(\epsilon), c(\epsilon))$ of (13) satisfies

$$\left.\frac{dc}{d\epsilon}\right|_{\epsilon=0} = -\frac{\psi_{K+1}}{s}, \tag{15}$$

$$\left.\frac{d\vec{n}}{d\epsilon}\right|_{\epsilon=0} = -\frac{\psi_{K+1}}{(\mu_1 - \mu_2)s}\vec{t}_0\vec{t}_0^T(\vec{x}_{K+1} - \vec{m}), \tag{16}$$

where $s \equiv \sum_{k=1}^K w_k$, $\vec{m} = \frac{1}{s}\sum_{k=1}^K w_k \vec{x}_k$, and $\mu_1 > \mu_2$ are the two eigenvalues of $C = \sum_{k=1}^K w_k(\vec{x}_k - \vec{m})(\vec{x}_k - \vec{m})^T$.

The proof of this theorem is left to the reader.

Observations:

- Eqn. (15) shows $dc/d\epsilon$ is proportional to $\psi(e)$, motivating the name ‘influence function’ for $\psi(e)$.
- Eqn. (15) shows the sense in which c is stable when $\psi(e)$ is bounded.
- Eqn. (16) shows $\vec{n}$ *may not be stable*, even for a bounded influence function $\psi(e)$ (since $|\vec{t}_0^{\,T}(\vec{x}_{K+1} - \vec{m})|$ may be large). In addition, notice the effect of $\mu_1 - \mu_2$, which measures the eccentricity of the original data points.
- Points $\vec{x}_{K+1}$ for which $|\psi_{K+1}\vec{t}_0^{\,T}(\vec{x}_{K+1} - \vec{m})|$ is large are called **leverage points**.

Controlling Leverage

The previous Theorem shows that even a redescending estimator such as GM is not robust to leverage points.

The figure on the right on p.21 provides an example where the estimator failed to reject the cluster of outliers in favour of points on the dominant line. For some initial guesses, this cluster has enough leverage to unduly influence the computed solution.

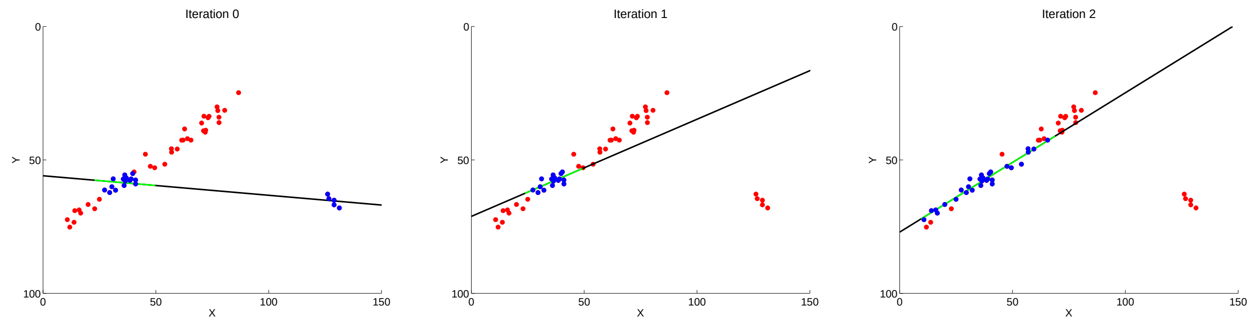
A simple strategy for dealing with leverage points is:

1. Estimate the distribution of data support along the fitted line (eg. project the weights w_k from $\vec{x}_k$ onto the line and blur to get a smooth function).
2. Determine a contiguous region of support along the fitted line, that is, an interval of support without any large gaps.
3. Reduce the weights w_k in the IRLS algorithm for points $\vec{x}_k$ significantly beyond the extent of contiguous support (eg. set such weights to 0).

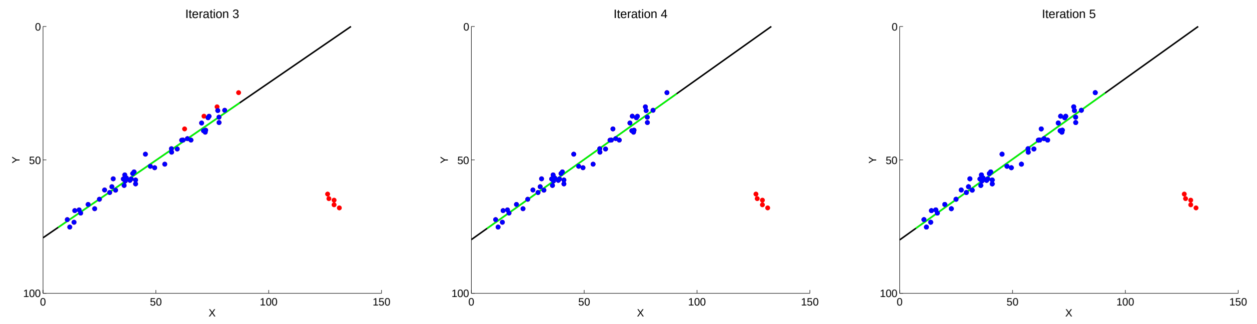
We show an example of this next.

Leverage Example

We can reduce leverage problems by controlling the support interval within the robust line estimation algorithm.



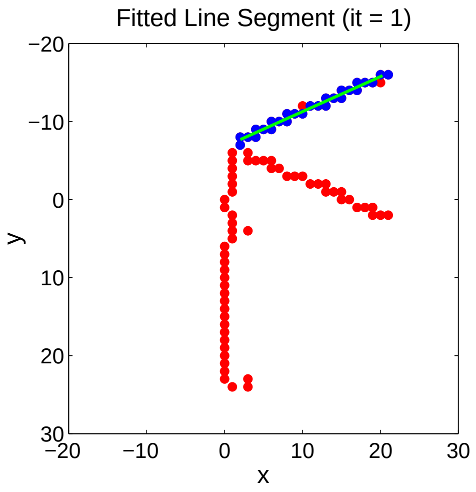
Iterations 0 to 2 above. Current line estimate (black). Edgels with significant weights (blue). The estimate for a contiguous region of support is also shown (green segment).

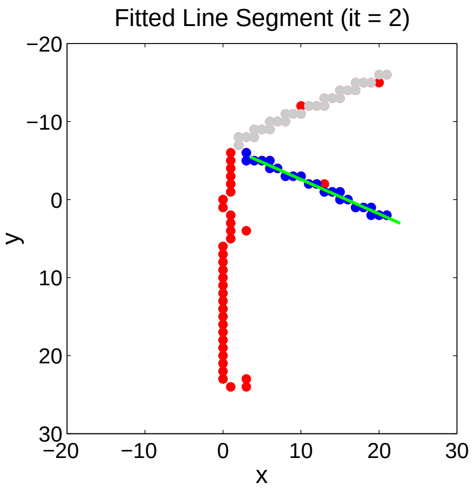


The remaining iterations 3 to 5, at which point the algorithm has converged.

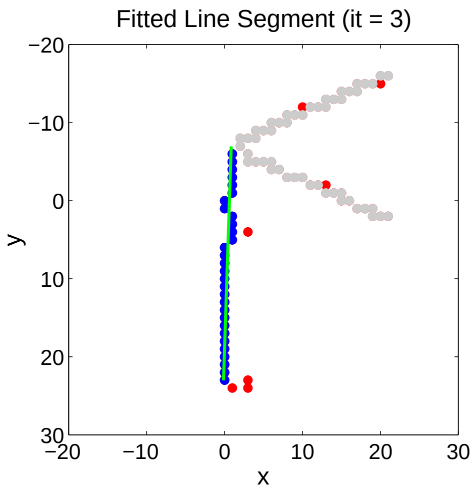
Successive Estimation

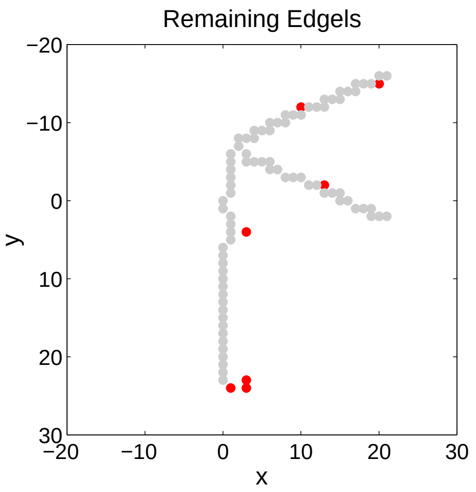
This approach can be applied to find lines one at a time:





The first fitted segment is shown (above left), along with edgels having significant robust weights (blue). These edgels are removed from the active data set (red). A second line segment is fit (above right) to the remaining edgels, and the new supporting edgels (blue) are removed.





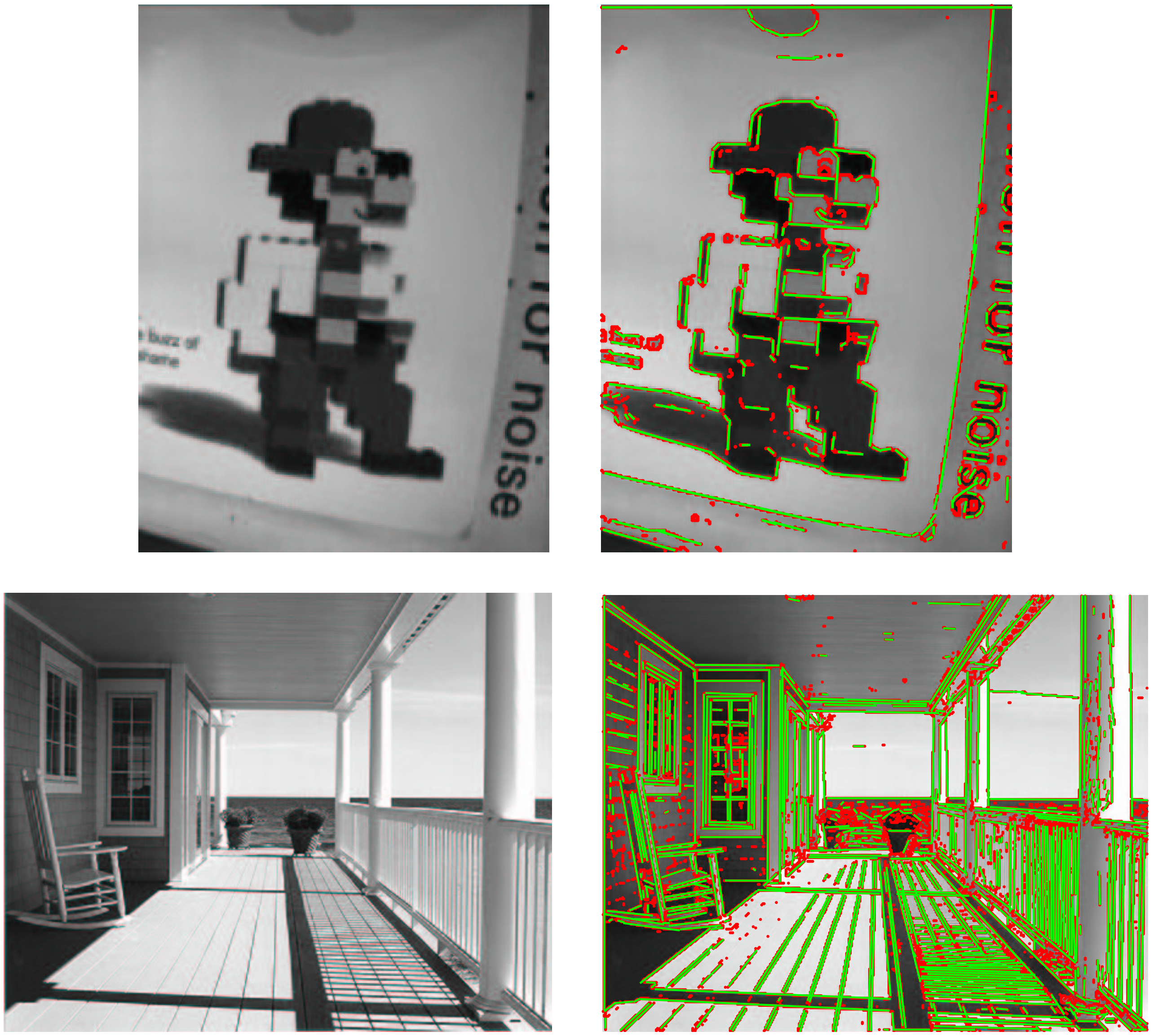
This successive fitting process continues until no further lines with contiguous support regions can be fit to the remaining edgels.

Outline of a Robust Line Estimation Algorithm

1. Initial Guess.
   - Randomly sample from the data. Eg. Set $(\vec{n}_0, c_0)$ according the the position and orientation of a sampled edgel.
2. Iterative Fitting.
   - Use the iteratively reweighted least squares algorithm to fit the line parameters. Maintain information about the support (from the data) for the line along its length, and use it to downweight data likely to cause leverage problems.
3. Verification.
   - Given a converged solution, decide whether that line has sufficient data support (eg. consider the sum of the weights). Discard the solution if it has insufficient support.
4. Model Selection.
   - For each verified line segment, remove the edgels which provide significant support for it from the data set. Repeat steps 1 through 4.

See page 2 and the next two pages for sample results. Similar strategies can be applied to many other estimation problems, as we discuss later.

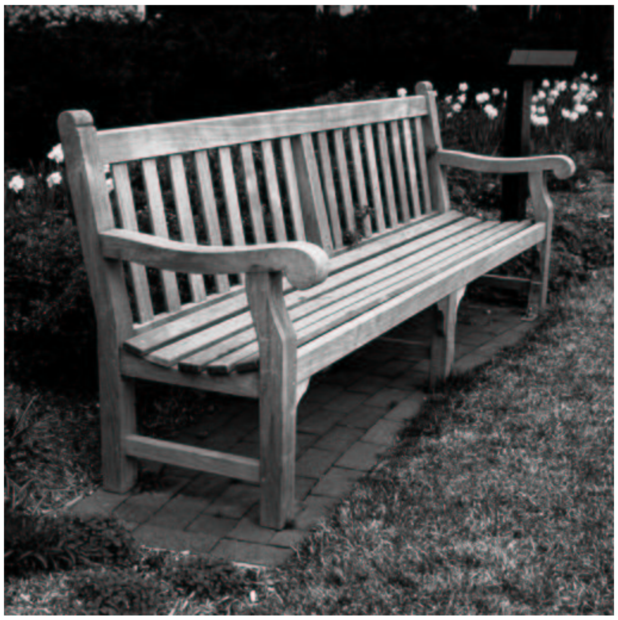
Linefinder Results



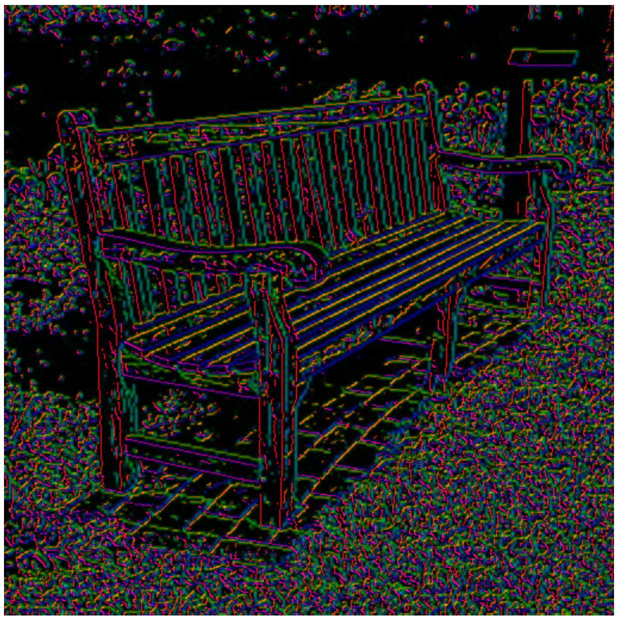
Edgel positions are shown in red above, with fitted lines in green. (In the electronic copy try expanding the view of small regions, such as the rocking chair, to see the fit in detail)

Linefinder Results

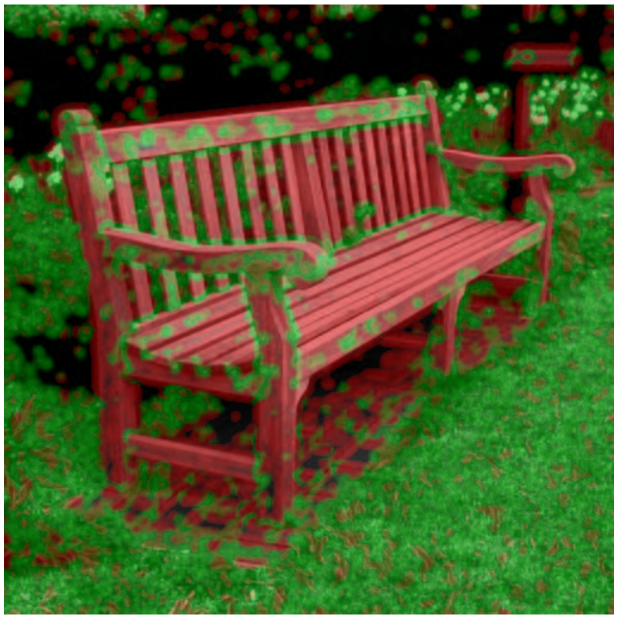
Parkbench Image



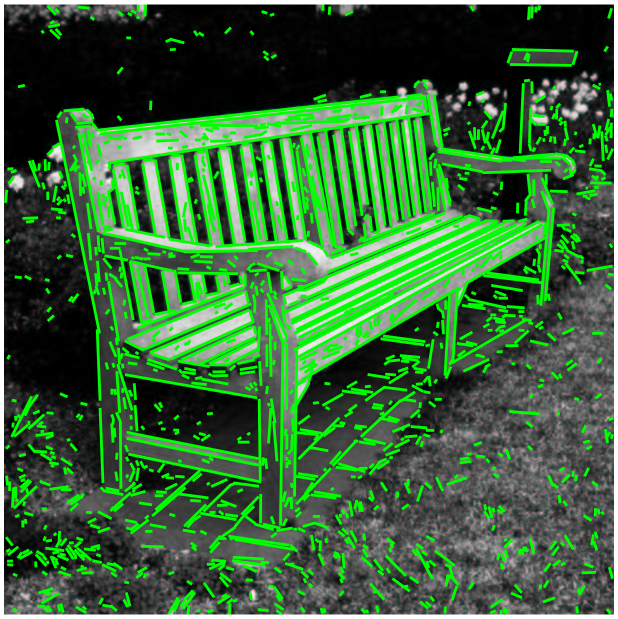
Colour-coded Edgel Orientation



Orientation Tensor



Estimated Lines



Random seeds for lines were obtained by sampling the edgels *only* within oriented regions (i.e., red regions in bottom-left plot).

Types of Errors

There are instances of several general types of errors visible in the previous line finder results:

- **Drop-outs.** Parts or all of a true line that are missing in the estimation results.
- **False positives.** An estimated line which does not correspond to any combination of true lines.
- **Over-segmentation.** Breaking one true line into several estimated segments (possibly colinear, parallel or otherwise).
- **Under-segmentation.** Joining two or more true lines into one estimated segment.
- **Parameter noise.** Small errors in the position and orientation estimates.
- **Model-type errors.** The somewhat inappropriate use of our current model (i.e., line segments fitted to edgels) for curves, texture, or thin bars in the image. More generally, model-type errors occur when the system can fit several types of models (eg. curves versus lines), but selects the wrong choice.

Evaluation of Results

We have not provided any quantitative evaluation of the results which, almost certainly, would provide invaluable data. Why not?

What should we use for ground truth? Human data? Synthetic data?

How should the results from different line-finder algorithms be compared? What evaluation metric should be used? Different algorithms which use the line finder results can be expected to vary in their sensitivity to different types of errors.

In the end what matters is how cost-effective various algorithms are when included in a working robot. (For an example of this style of assessment, see Doughtery and Bowyer, 1998, Objective evaluation of edge detectors using a formally defined framework.)

Here we simply punt (i.e., accept our current gains and move on).

Robust Estimation: Summary

For the estimation of model parameters from image data:

- A redescending M-estimator, such as the Geman-McLure estimator, downweights the influence of data with large errors.
- The use of such an estimator leads to a non-linear optimization problem for the model parameters.
- The optimization problem typically has multiple local minima, some of which provide useful estimates while others are extraneous.
- The iteratively reweighted least squares algorithm provides one way to find local minima of the resulting objective function.
- Leverage points can significantly skew the parameter estimates, and can be controlled by limiting the spatial extent of the data being fit. This can also reduce the number of extraneous minima.
- Initial guesses can be generated by randomly sampling the data (see RANSAC). Another approach uses continuation with decreasing σ (see deterministic annealing and graduated non-convexity).
- One way to choose the number of models is to iteratively fit individual models, removing the data providing support for each model before fitting the next (see also Bayesian model selection).