Fitting Parameterized 3-D Models to Images

David Lowe

Presented by Ray Jones

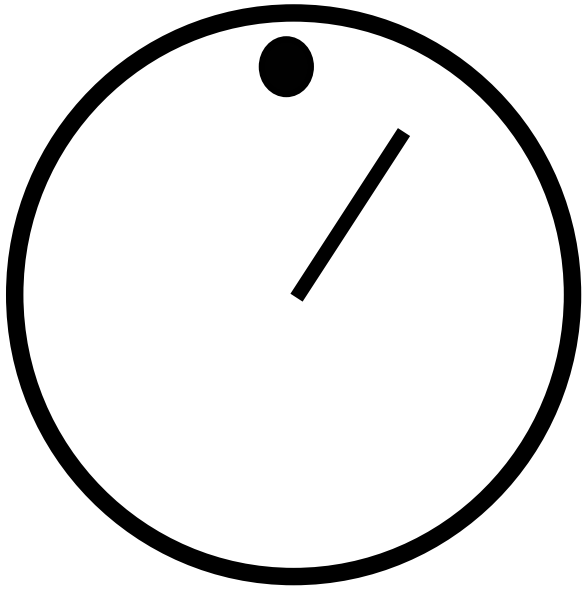
Goal:

Given a parameterized model, solve for all viewpoint and model parameters from a single image.

Example: Clock (One hand, 2D view)

- Model parameter: position of hand (1D angle? 2D vector?)

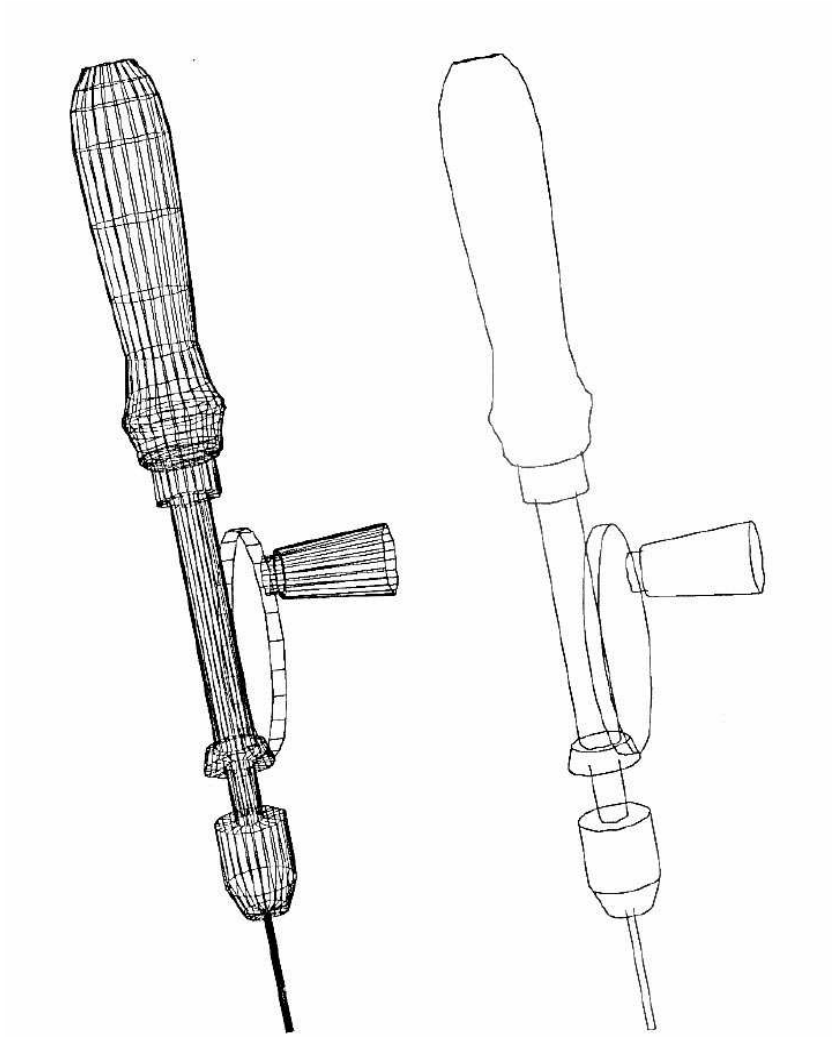
- Viewing parameters: translation and rotation



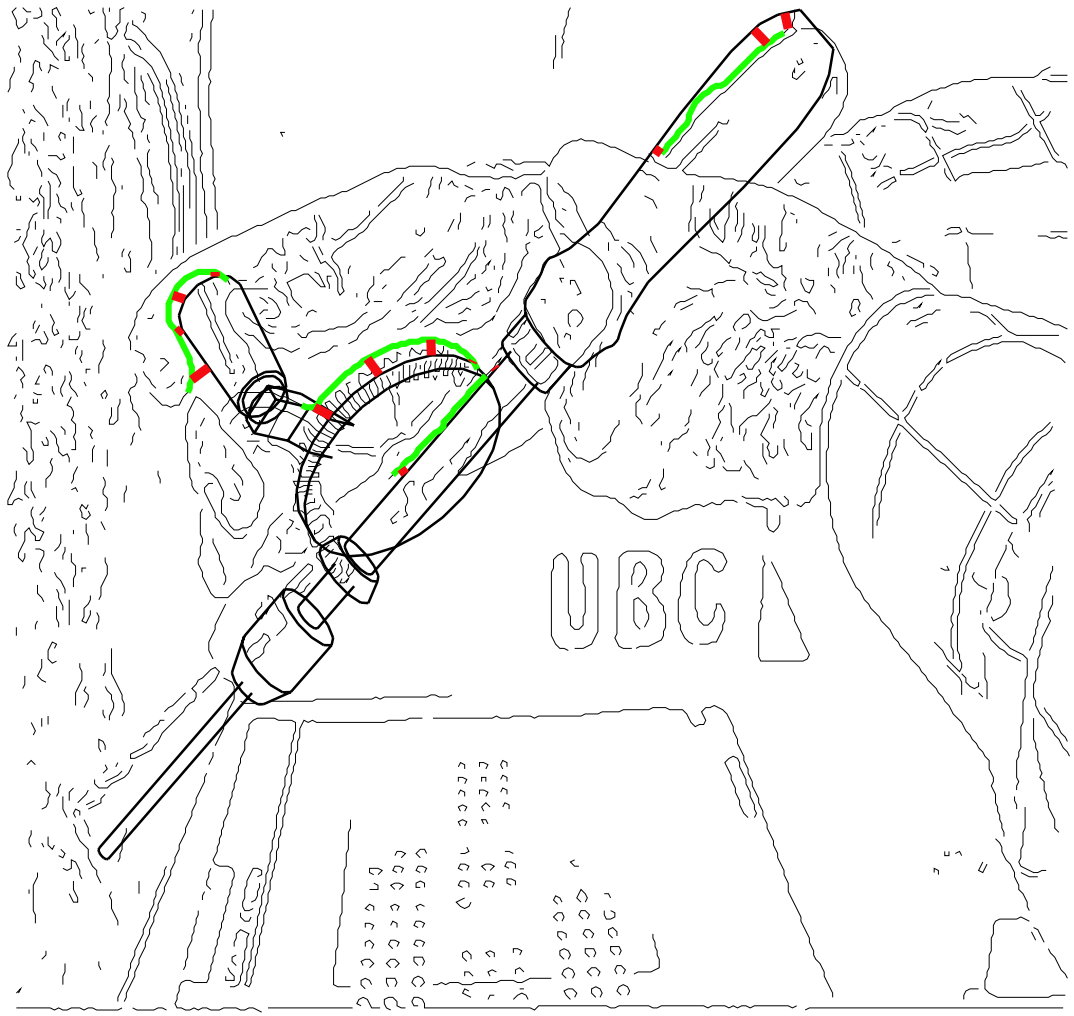
Method: Iteratively match projection of model to image

- Project *model edges* into image (similar to a hidden-line drawing)
- Choose a set of *image edges* to match to model, and points on those edges at which to compute errors
- Compute partial derivatives of mismatches relative to model and viewing parameters.
- Adjust model/viewing parameters (Newton's method)
- Repeat until convergence

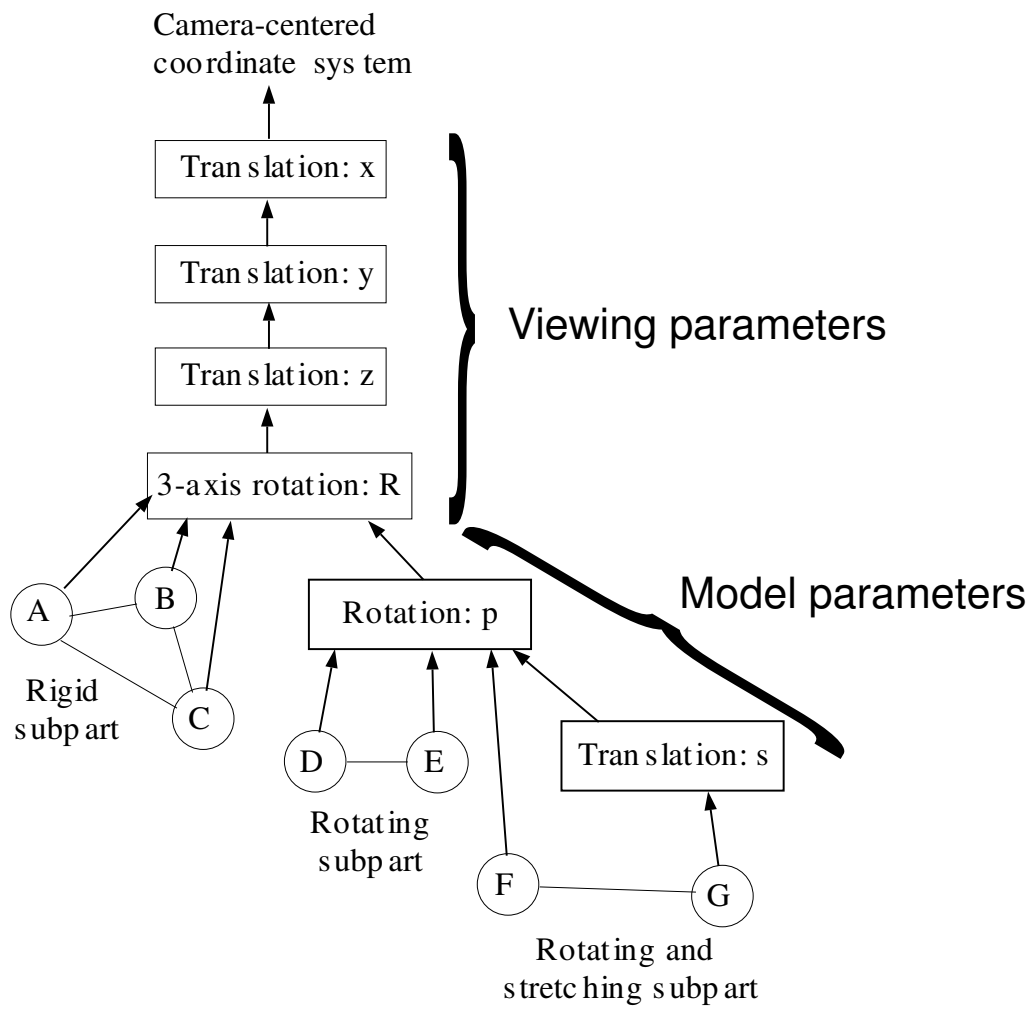
Projecting model edges



Choose image edges and points at which to compute errors



Hierarchical model representation



Updating the parameter estimates

- $\mathbf{p}^{(i+1)} = \mathbf{p}^{(i)} - \mathbf{x}$

- Linearize the error function

$$\mathbf{J}\mathbf{x} = \mathbf{e}, \qquad \mathbf{J}_{ij} = \frac{\partial e_i}{\partial x_j}$$

- This system is overdetermined, so find

$$\arg\min_{\mathbf{x}} ||\mathbf{J}\mathbf{x} - \mathbf{e}||^2$$

- Solve using normal equations:

$$\mathbf{J}^{\mathbf{T}}\mathbf{J}\mathbf{x} = \mathbf{J}^{\mathbf{T}}\mathbf{e}$$

Efficient computation of partial derivatives

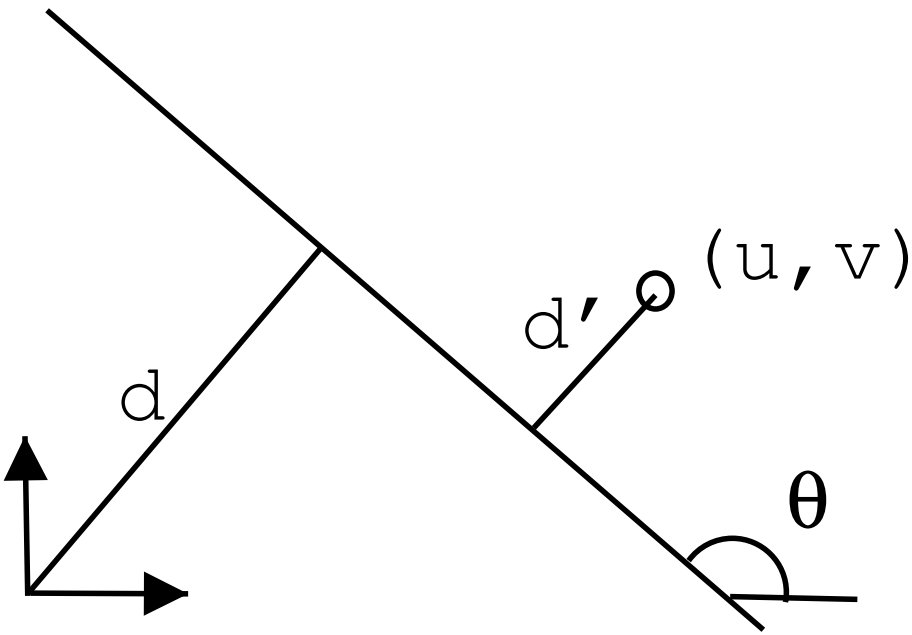
- Every point on the model has a hierarchy of translations associated with it, and is linked to a node in the hierarchy.

- We can precompute the partial derivatives at each node for every point on the model linked to that node, relative to every parameter that affects it in the hierarchy.

- The partials are computed in "camera centered coordinates" (3D), which are then projected to image coordinates (2D).

Distance-to-edge error

- We have points in the image, but don't know their corresponding points on the model, so we use perpendicular distance instead.



Signed perpendicular distance from (u, v) to the line is given by $d' = u \sin\theta - v \cos\theta - d$.

Partial derivative of distance-to-edge

$$\frac{\partial d'}{\partial p} = \sin\theta \frac{\partial u}{\partial p} - \cos\theta \frac{\partial v}{\partial p}$$

The author notes that curved silhouette edges do not require special consideration, because the surface normal is exactly perpendicular to the viewing direction at such points.

[However, it seems that terms for $\frac{\partial \theta}{\partial p}$ should have been included above.]

Stabilization

Use priors on the parameters to stabilize the computation.

$$\begin{bmatrix} \mathbf{J} \\ \mathbf{W} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{e} \\ \mathbf{W}\mathbf{d} \end{bmatrix} \qquad\qquad W_{ii} = \frac{1}{\sigma_i}$$

$\mathbf{d}$ are the priors for the paramaters computed from previous frames (velocity). σ_i is the standard deviation of the ith parameter.

Under the assumption that the initial guess for the parameters is reasonable, $\sigma = \pi/2$ for rotation parameters, $\sigma =$(image size) for translations, ... The weighting above is equivalent to $\sigma = 1$ for image measurements.

Stablized Normal Equation

$$
\begin{aligned}
\begin{bmatrix} \mathbf{J} \\ \mathbf{W} \end{bmatrix} \mathbf{x} &= \begin{bmatrix} \mathbf{e} \\ \mathbf{Wd} \end{bmatrix} \\
\begin{bmatrix} \mathbf{J^T} & \mathbf{W^T} \end{bmatrix} \begin{bmatrix} \mathbf{J} \\ \mathbf{W} \end{bmatrix} \mathbf{x} &= \begin{bmatrix} \mathbf{J^T} & \mathbf{W^T} \end{bmatrix} \begin{bmatrix} \mathbf{e} \\ \mathbf{Wd} \end{bmatrix} \\
(\mathbf{J^T J} + \mathbf{W^T W})\mathbf{x} &= \mathbf{J^T e} + \mathbf{W^T W d}
\end{aligned}
$$

$\mathbf{W^T W}$ is a diagonal matrix, so stabilization doesn't cost very much. The author claims that stabilization also keeps the matrix from being ill-conditioned enough that solving for $\mathbf{x}$ cannot be done via normal equations.

Forcing Convergence

Replace $\mathbf{W}$ with $\lambda\mathbf{W}$ in the stablized equation

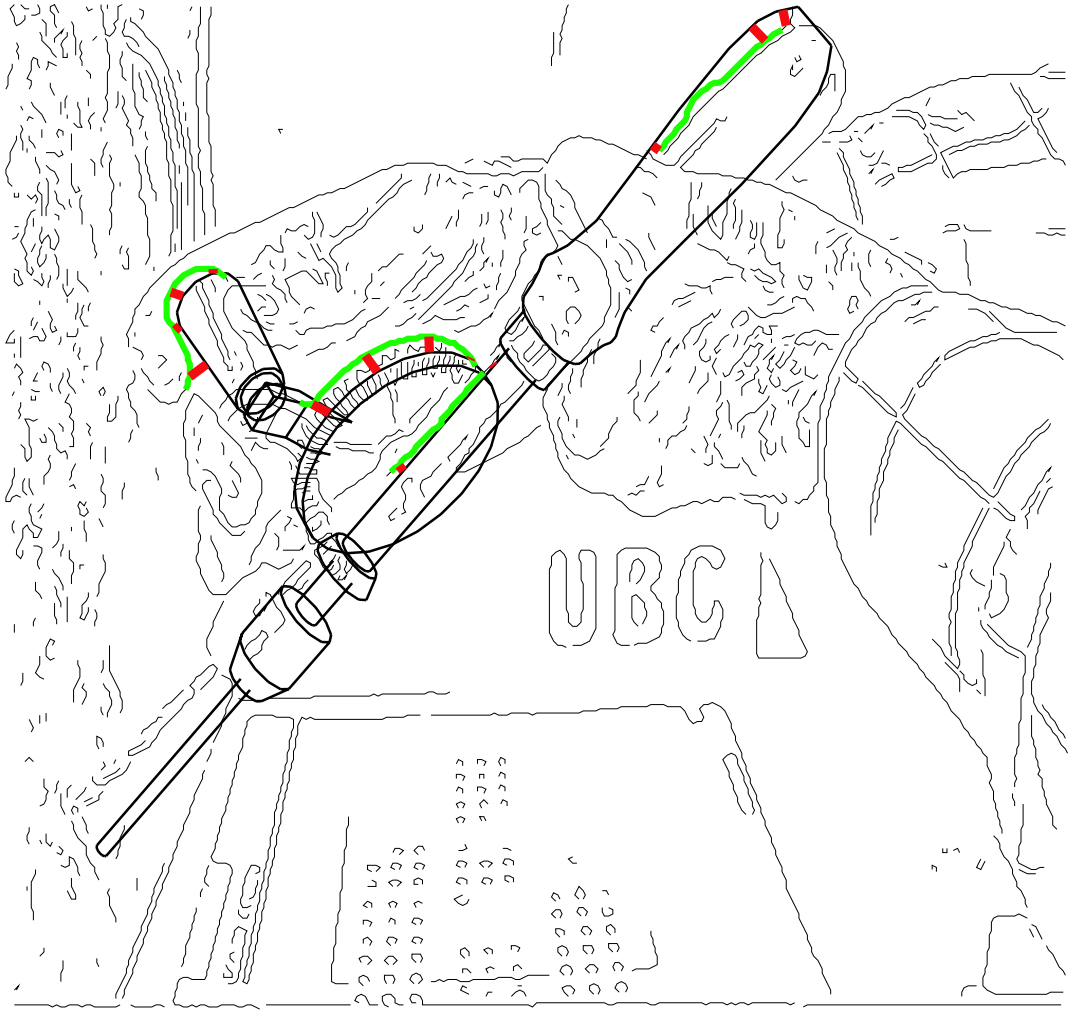
$$\begin{bmatrix} \mathbf{J} \\ \lambda\mathbf{W} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{e} \\ \lambda\mathbf{W}\mathbf{d} \end{bmatrix}$$

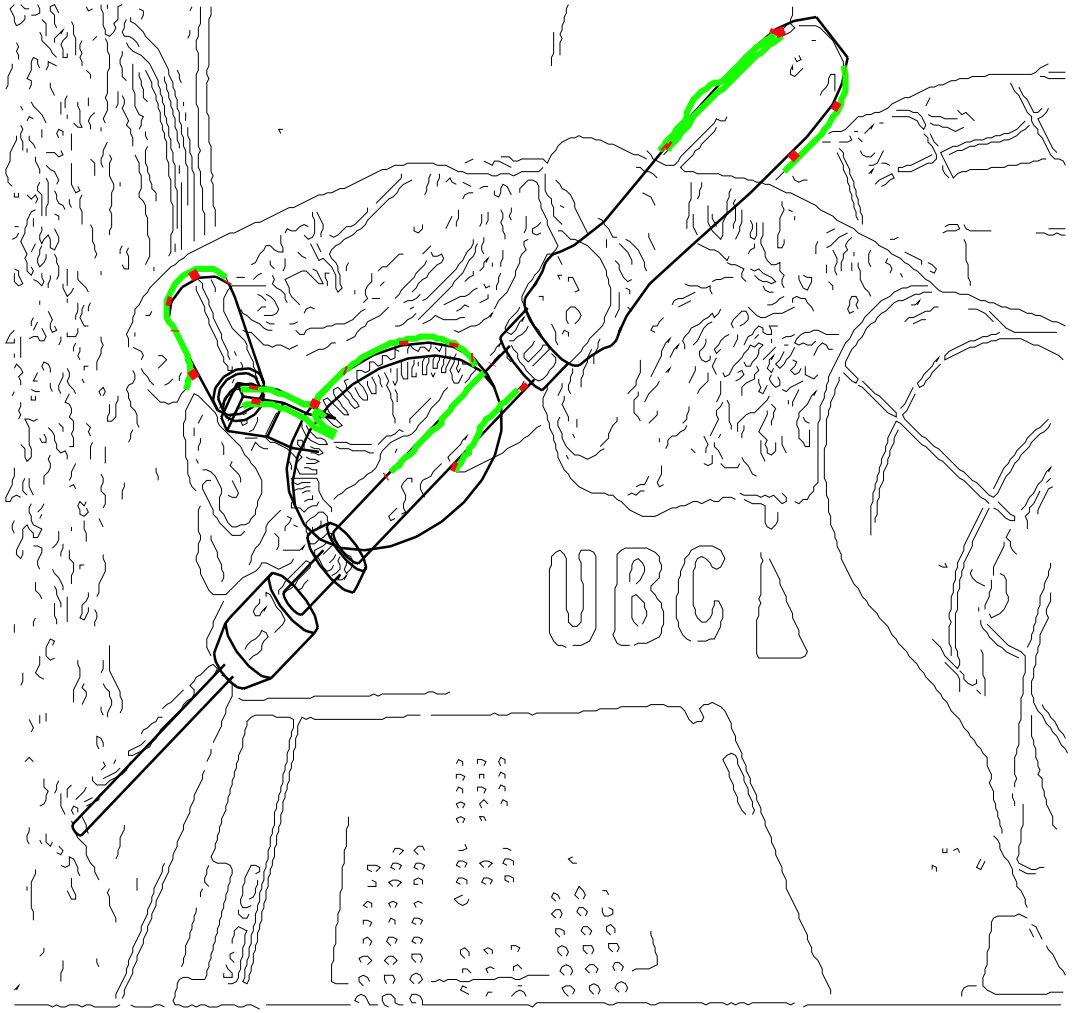
Solving this equation minimizes

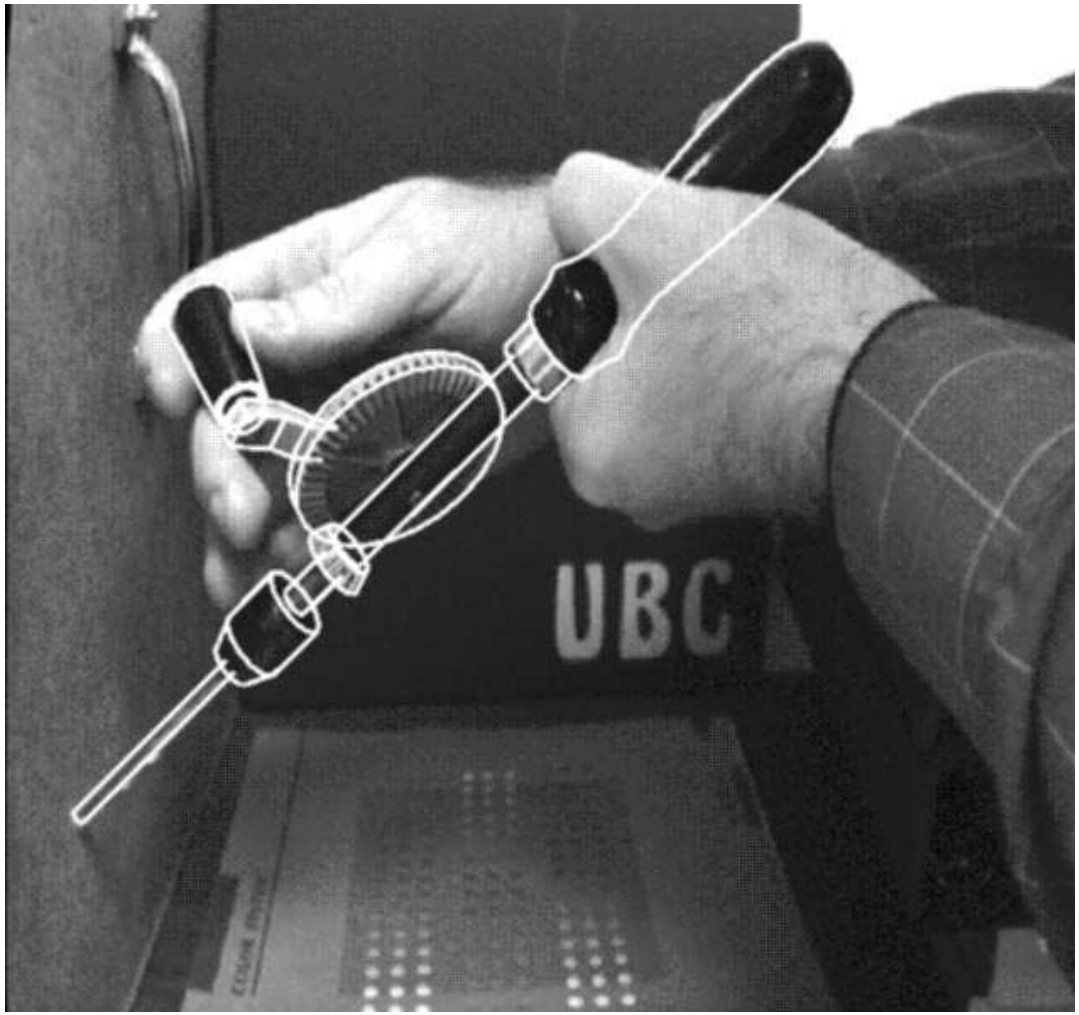
$$||\mathbf{J}\mathbf{x} - \mathbf{e}||^2 + \lambda^2 ||\mathbf{W}(\mathbf{x} - \mathbf{d})||^2$$

λ is increased by a factor of 10 whenever the residual increases, and decreases by a factor of 10 otherwise. As λ increases, the effect is to give more weight to the prior. As it decreases, the algorithm behaves more like Newton's method.

Example from paper


UBC


UBC


UBC