Non-Linear Least Squares

Pascal Fua
IC-CVLab

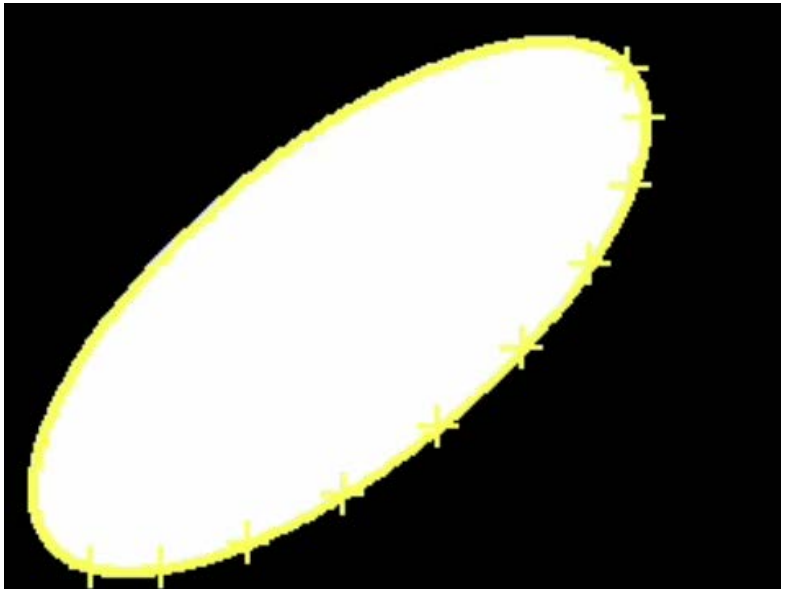
Fitting Ellipses

For each point :

$$d_a(\mathbf{x}_i,\mathbf{X}) = au_i^2 + bu_iv_i + cv_i^2 + du_i + ev_i + f$$
$$= \begin{bmatrix} u_i^2 & u_iv_i & v_i^2 & u_i & v_i & 1 \end{bmatrix} \bullet \mathbf{X}$$

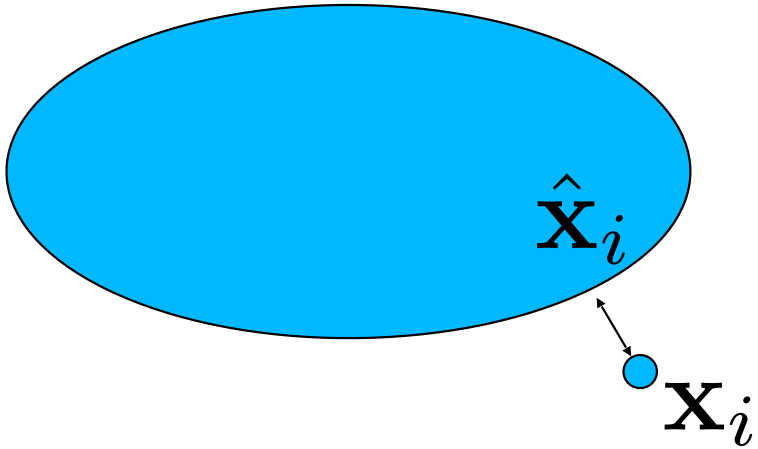
Minimize :

$$\sum_i d_a(\mathbf{x}_i,\mathbf{X})^2 = \|\mathbf{AX}\|^2 \text{ subject to } \|\mathbf{X}\| = 1$$



$$\text{where } \mathbf{A} = \begin{bmatrix} u_1^2 & u_1v_1 & v_1^2 & u_1 & v_1 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ u_n^2 & u_nv_n & v_n^2 & u_n & v_n & 1 \end{bmatrix}$$

Algebraic vs Euclidean Distances



For each point :

$$d_a(\mathbf{x}_i,\mathbf{X}) = au_i^2 + bu_iv_i + cv_i^2 + du_i + ev_i + f$$ Algebraic distance

$$d_e(\mathbf{x}_i,\mathbf{X}) = d(\mathbf{x}_i,\hat{\mathbf{x}}_i)$$ Euclidean distance

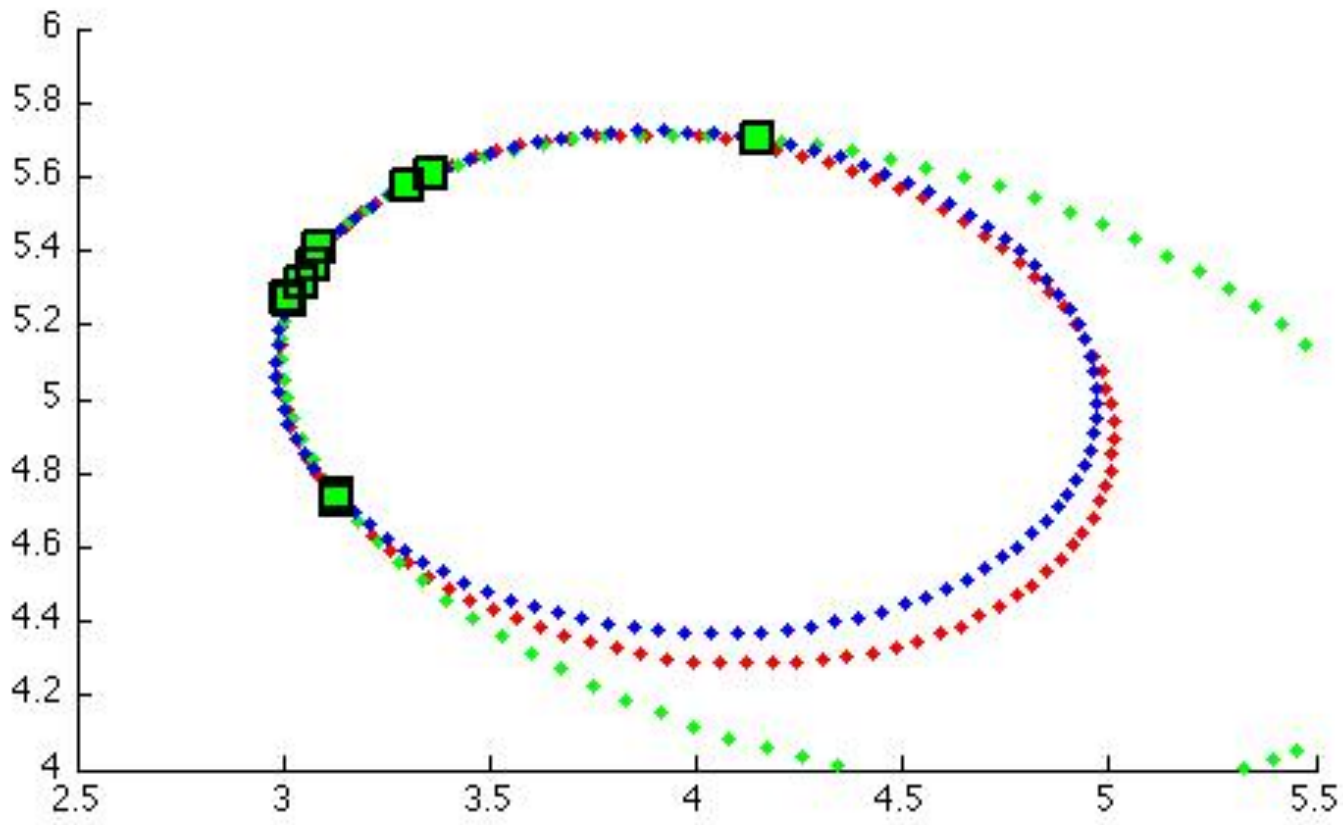
Minimize :

$$\sum_i d_e(\mathbf{x}_i,\mathbf{X})^2 \text{ instead of } \sum_i d_a(\mathbf{x}_i,\mathbf{X})^2$$

Euclidean fit Algebraic fit

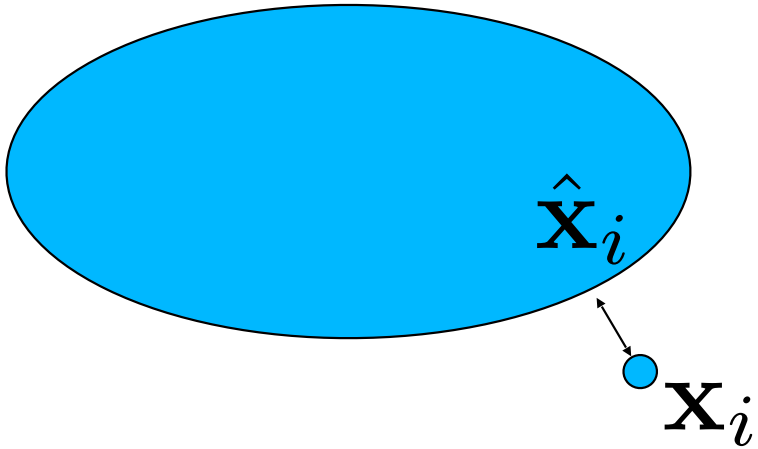
Euclidean vs Algebraic Fits

Green squares: Slightly noisy data points.



Red: Ground truth
Green: Algebraic Fit
Blue: Euclidean Fit.

Algebraic vs Euclidean Distances



For each point :

$$d_a(\mathbf{x}_i,\mathbf{X}) = au_i^2 + bu_iv_i + cv_i^2 + du_i + ev_i + f$$

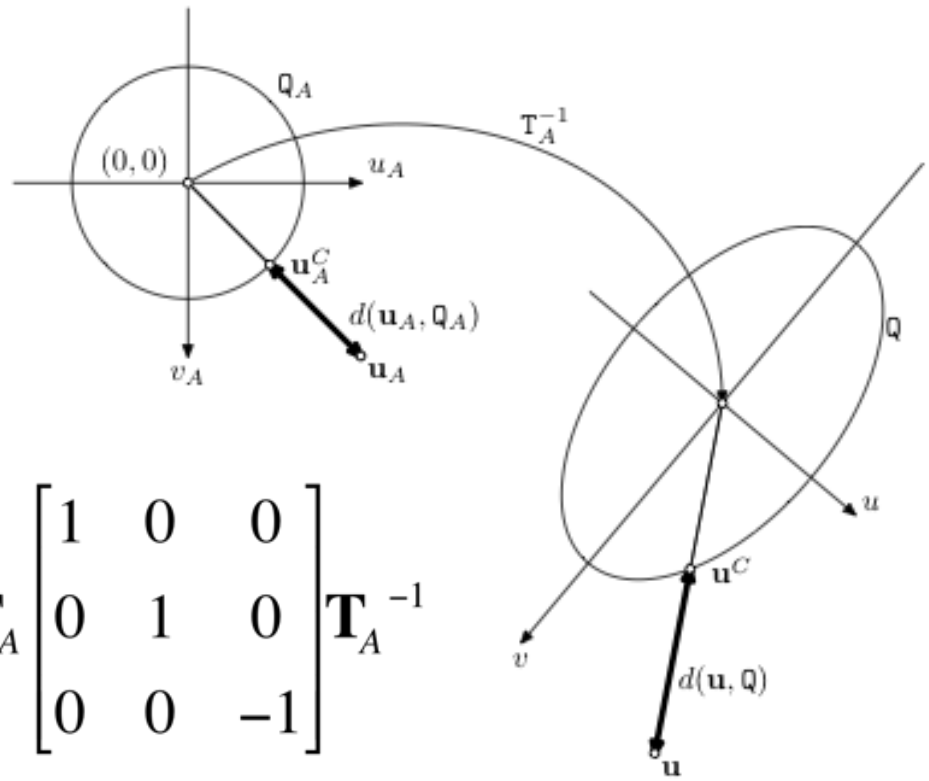
$$d_e(\mathbf{x}_i,\mathbf{X}) = d(\mathbf{x}_i,\hat{\mathbf{x}}_i)$$

Minimize :

$$\sum_{\mathrm{i}} d_e(\mathbf{x}_i,\mathbf{X})^2 \text{ instead of } \sum_{\mathrm{i}} d_a(\mathbf{x}_i,\mathbf{X})^2$$

$\rightarrow$Non - linear minimization is required.

Euclidean Distance to an Ellipse



$$\mathbf{u}^t\mathbf{Q}\mathbf{u} = 0 \text{ where } \mathbf{Q} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & 1 \end{bmatrix} = \mathbf{T}_A \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{T}_A^{-1}$$

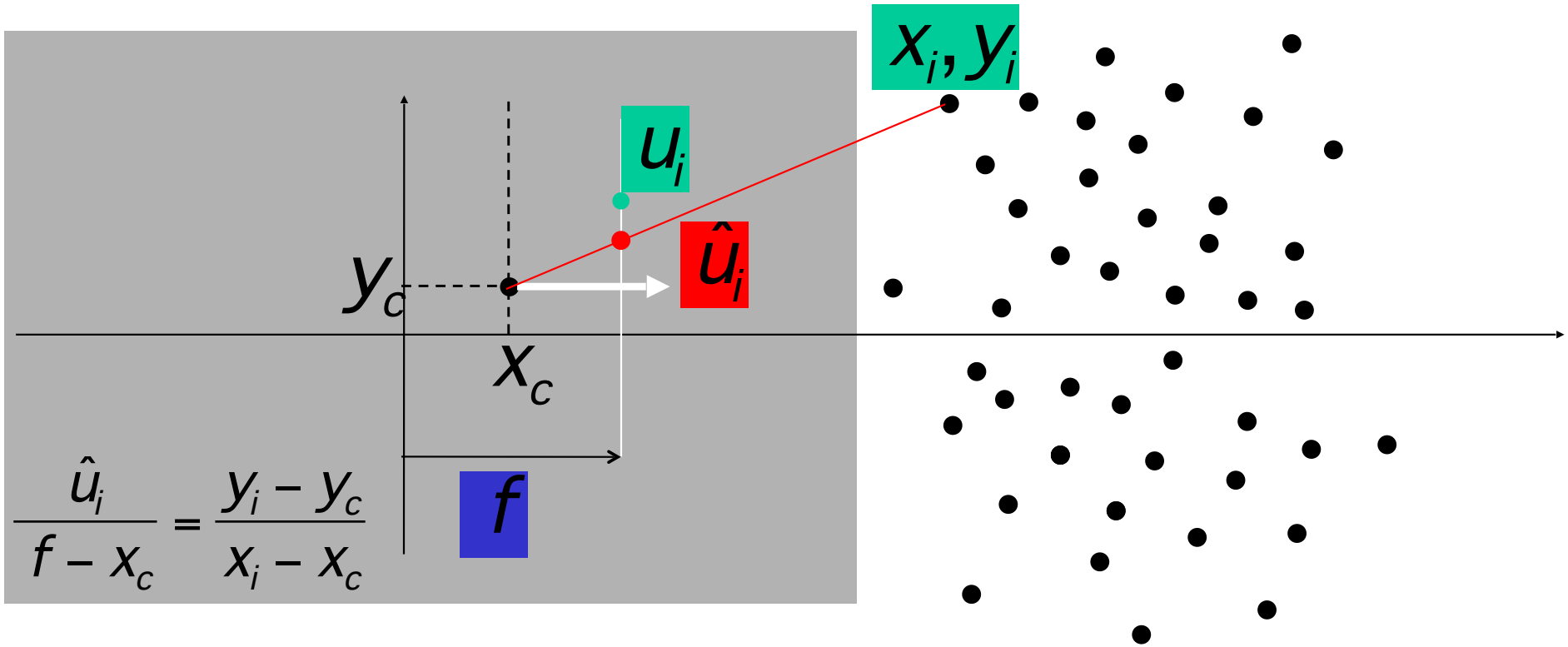
$$\mathbf{u}_A = \mathbf{T}_A^{-1}\mathbf{u} = [u_A, v_A, 1]$$

$$\mathbf{u}_A^C = [u_A, v_A, \sqrt{u_A^2 + v_A^2}]$$

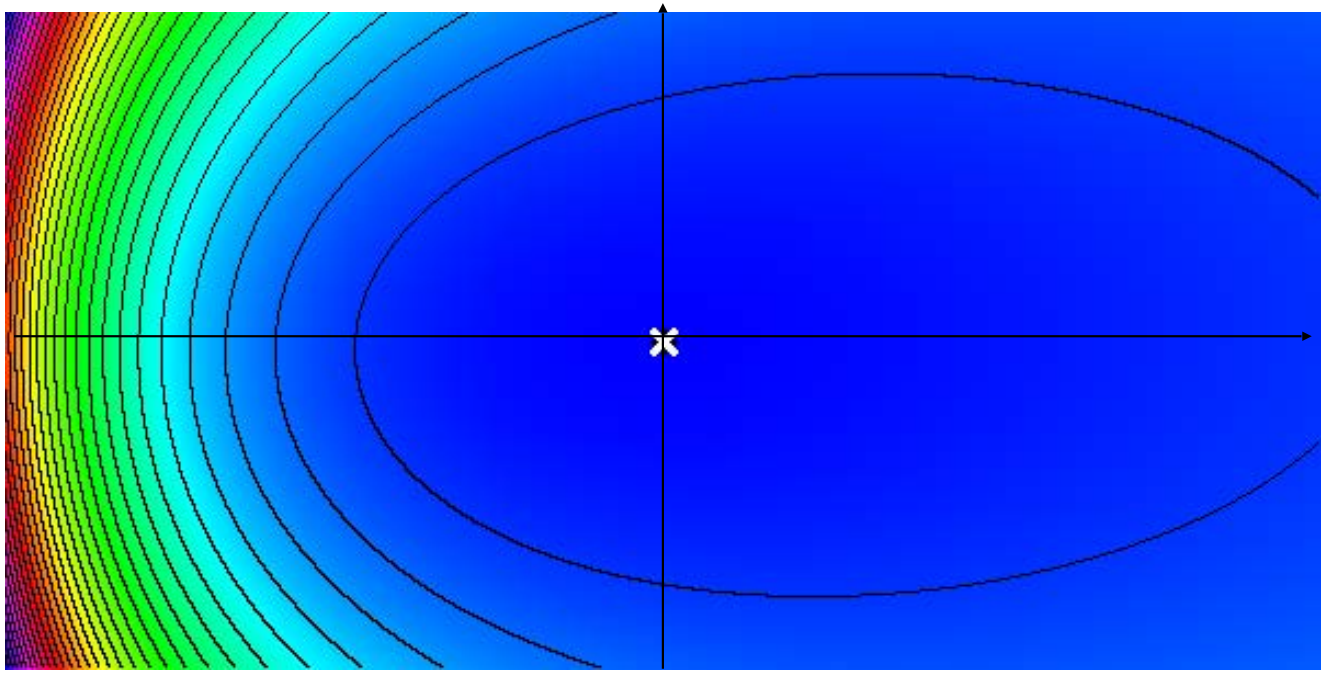
$$\mathbf{u}^C = \mathbf{T}_A \mathbf{u}_A^C$$

1D Camera

- 1D camera under 2D translation, parameterized by (Xc, Yc), the camera center coordinates.
- 100 points taken at random in [400;1000]x[-500;+500]
- True camera position at (0, 0)



Energy Landscape

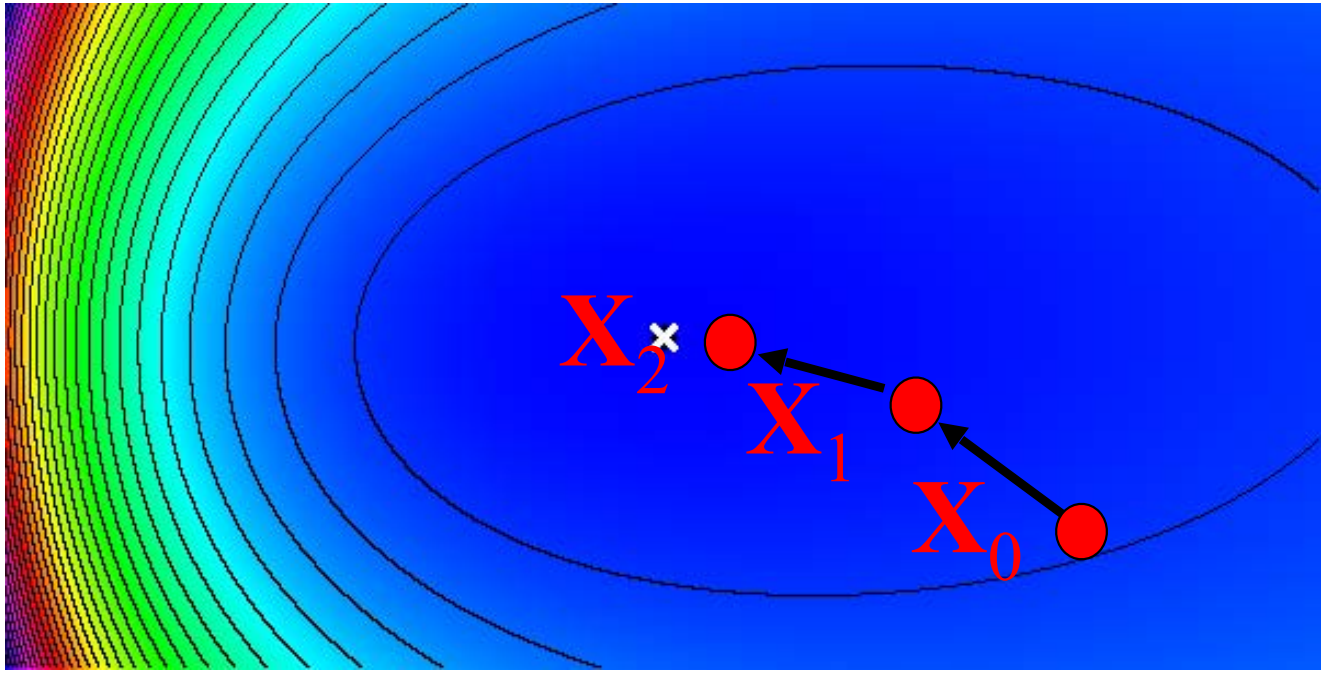


$$F(\mathbf{X}) = \sum_i d(\mathbf{x}_i, \mathbf{X})^2$$

$$\mathbf{X} = (x_c, y_c)$$

$$d(\mathbf{x}_i, \mathbf{X}) = \left[u_i - (f - x_c) \frac{y_i - y_c}{x_i - x_c} \right]$$

Non-Linear Minimization



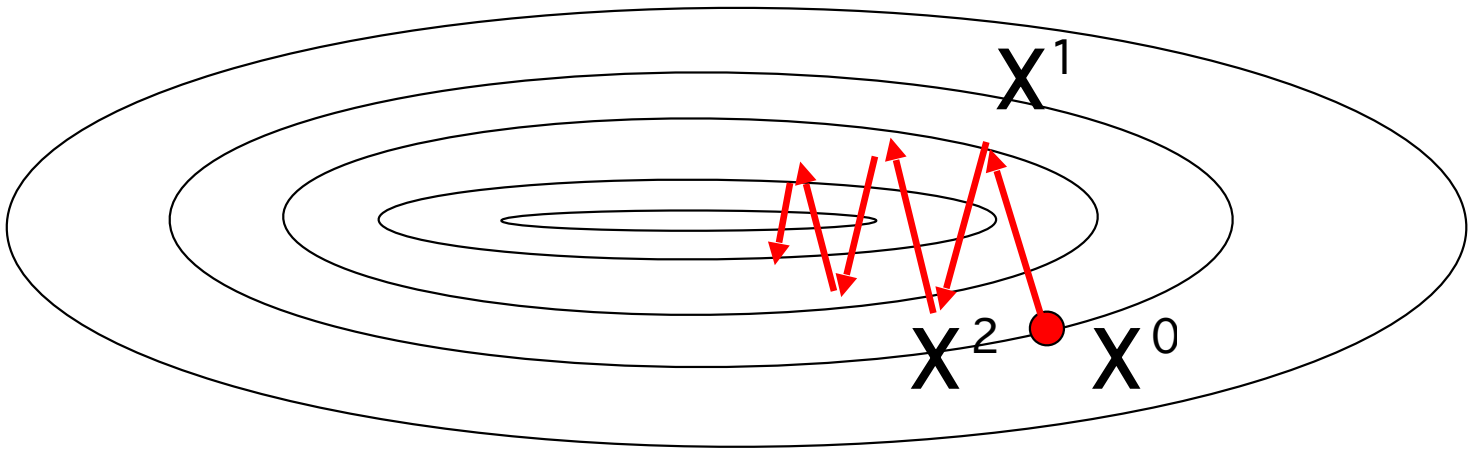
- Gradient descent
- Conjugate gradient
- Levenberg-Marquardt

Gradient Descent

$$\text{Minimize } F(\mathbf{X}) = \sum_i d(\mathbf{x}_i, \mathbf{X})^2 \text{ w.r.t } \mathbf{X}:$$

$$\mathbf{X}^{t+1} = \mathbf{X}^t - \lambda \nabla F(\mathbf{X}^t)$$

$$\nabla F(\mathbf{X}) = \left[\frac{\delta F}{\delta X_1} \cdots \frac{\delta F}{\delta X_n} \right]$$



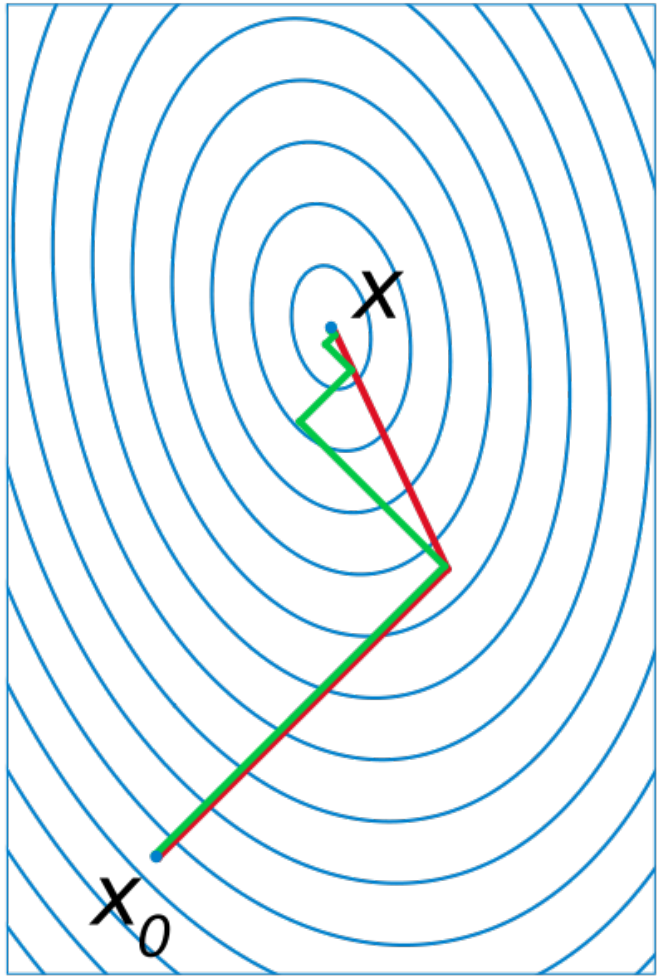
Issues:

- How to choose λ?
- Many iterations in long and narrow valleys.

Conjugate Gradient

Take the search direction to be a weighted average of the gradient vector and the previous search directions:

1. Start at $\mathbf{x}_0$.
2. $\mathbf{g}_0 = \nabla F(\mathbf{x}_0)$.
3. For k from 0 to $n-1$:
   (a) Find α_k that minimizes $f(\mathbf{x}_k + \alpha_k \mathbf{g}_k)$.
   (b) $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{g}_k$.
   (c) $\beta_k = \frac{\|\nabla f(\mathbf{x}_{k+1})\|^2}{\|\nabla f(\mathbf{x}_k)\|^2}$.
   (d) $\mathbf{g}_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{g}_k$.
4. $\mathbf{x}_0 = \mathbf{x}_n$ and go to step 2 until convergence.



—> Faster convergence.

Gauss Newton

Minimize $\|F(\mathbf{X})\|^2$, where $F(\mathbf{X}) = [d(\mathbf{x}_1, \mathbf{X}), \ldots, d(\mathbf{x}_n, \mathbf{X})]^T$.

Given the current value of $\mathbf{X}$, find $\mathbf{dX}$ that minimizes

$$\begin{aligned}
\|F(\mathbf{X}+\mathbf{dX})\|^2 &= F(\mathbf{X}+\mathbf{dX})^T F(\mathbf{X}+\mathbf{dX}) \\
\Rightarrow 0 &= \frac{\delta}{\delta \mathbf{X}}(F(\mathbf{X}+\mathbf{dX})^T F(\mathbf{X}+\mathbf{dX})) \\
&= \frac{\delta}{\delta \mathbf{X}}((F(\mathbf{X})+\mathbf{JdX})^T (F(\mathbf{X})+\mathbf{JdX})) \\
&= \mathbf{J}^T (F(\mathbf{X})+\mathbf{JdX})
\end{aligned}$$

First order Taylor expansion

$$\mathbf{J} = \begin{bmatrix} \frac{\delta F_1}{\delta \mathbf{X}_1} & \cdots & \frac{\delta F_1}{\delta \mathbf{X}_n} \\ \cdots & \cdots & \cdots \\ \frac{\delta F_m}{\delta \mathbf{X}_1} & \cdots & \frac{\delta F_m}{\delta \mathbf{X}_n} \end{bmatrix}$$

Gauss Newton

Minimize $\|F(\mathbf{X})\|^2$, where $F(\mathbf{X}) = [d(\mathbf{x}_1, \mathbf{X}), \ldots, d(\mathbf{x}_n, \mathbf{X})]^T$.

Given the current value of $\mathbf{X}$, find $\mathbf{dX}$ that minimizes

$$\begin{aligned}
\|F(\mathbf{X}+\mathbf{dX})\|^2 &= F(\mathbf{X}+\mathbf{dX})^T F(\mathbf{X}+\mathbf{dX}) \\
\Rightarrow 0 &= \frac{\delta}{\delta \mathbf{dX}}(F(\mathbf{X}+\mathbf{dX})^T F(\mathbf{X}+\mathbf{dX})) \\
&= \frac{\delta}{\delta \mathbf{dX}}((F(\mathbf{X})+\mathbf{JdX})^T (F(\mathbf{X})+\mathbf{JdX})) \\
&= \mathbf{J}^T (F(\mathbf{X})+\mathbf{JdX})
\end{aligned}$$

First order Taylor expansion

$$\mathbf{J} = \begin{bmatrix} \frac{\delta F_1}{\delta \mathbf{X}_1} & \cdots & \frac{\delta F_1}{\delta \mathbf{X}_n} \\ \cdots & \cdots & \cdots \\ \frac{\delta F_m}{\delta \mathbf{X}_1} & \cdots & \frac{\delta F_m}{\delta \mathbf{X}_n} \end{bmatrix}$$

Gauss Newton

Minimize $\|F(\mathbf{X})\|^2$, where $F(\mathbf{X}) = [d(\mathbf{x}_1, \mathbf{X}), \ldots, d(\mathbf{x}_n, \mathbf{X})]^T$.

Given the current value of $\mathbf{X}$, solve

$$\mathbf{J}^T \mathbf{J} \mathbf{d}\mathbf{X} = -\mathbf{J}^T F(\mathbf{X}) \ .$$

- Convergence in a single step if d is linear in $\mathbf{X}$.
- But the steps can be too large if it is not.

—>The sum of squares can increase instead of decrease.

Levenberg-Marquardt

Gauss Newton:

$$\mathbf{J}^T\mathbf{J}\mathbf{d}\mathbf{X} = -\mathbf{J}^T F(\mathbf{X}) \ .$$

Levenberg-Marquardt:

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\mathbf{d}\mathbf{X} = -\mathbf{J}^T F(\mathbf{X}) \ .$$

Algorithm:

1. Initialize λ, e.g. $\lambda = 0.001$,
2. $\mathbf{d}\mathbf{X} = -(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T F(\mathbf{X})$.
3. If $\|F(\mathbf{X} + \mathbf{d}\mathbf{X}\| > \|F(\mathbf{X}\|$, $\lambda \leftarrow 10\lambda$ and go to step 2.
4. Otherwise, $\lambda \leftarrow \lambda/10$, $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{d}\mathbf{X}$, and go to step 2.

Levenberg-Marquardt

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\mathbf{dX} = -\mathbf{J}^T F(\mathbf{X})$$

- When λ is small, LVM performs Gauss-Newton iterations.
- When λ is large, LVM performs steepest descent with a small step.

Damped Newton

Second order Taylor expansion:

$$f(\mathbf{x}+\mathbf{dx}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{dx} + \frac{1}{2}\mathbf{dx}^T H(\mathbf{x})\mathbf{dx}$$
$$\nabla f(\mathbf{x}+\mathbf{dx}) \approx \nabla f(\mathbf{x}) + H(\mathbf{x})\mathbf{dx}$$

Introduce a damping term:

$$\text{Regular Newton Method: } H(\mathbf{x})\mathbf{dx} = -\nabla f(\mathbf{x})$$
$$\text{Damped Newton: } (H(\mathbf{x}) + \lambda \mathbf{I})\mathbf{dx} = -\nabla f(\mathbf{x})$$

- $\lambda = 0$: Regular Newton
- $\lambda >> 0$: Gradient descent

Damped Newton vs LVM

$$f(\mathbf{X}) = \frac{1}{2} F(\mathbf{X})^T F(\mathbf{X})$$

$$\mathbf{g}(\mathbf{X}) = \mathbf{J}(\mathbf{X})^T \mathbf{J}(\mathbf{X})$$

$$\mathbf{H}(\mathbf{X}) = \sum_i F_i(\mathbf{X}) \frac{d^2 F_i}{d\mathbf{X}^2} + \mathbf{J}^T \mathbf{J}$$

$$\mathbf{J} = \begin{bmatrix} \frac{\delta F_1}{\delta \mathbf{X}_1} & \cdots & \frac{\delta F_1}{\delta \mathbf{X}_n} \\ \cdots & \cdots & \cdots \\ \frac{\delta F_m}{\delta \mathbf{X}_1} & \cdots & \frac{\delta F_m}{\delta \mathbf{X}_n} \end{bmatrix}$$

Small when the F_i or $\frac{d^2 F_i}{d\mathbf{X}^2}$ are small.

Damped Newton: $(\mathbf{H} + \lambda \mathbf{I}) d\mathbf{X} = -\mathbf{g}$

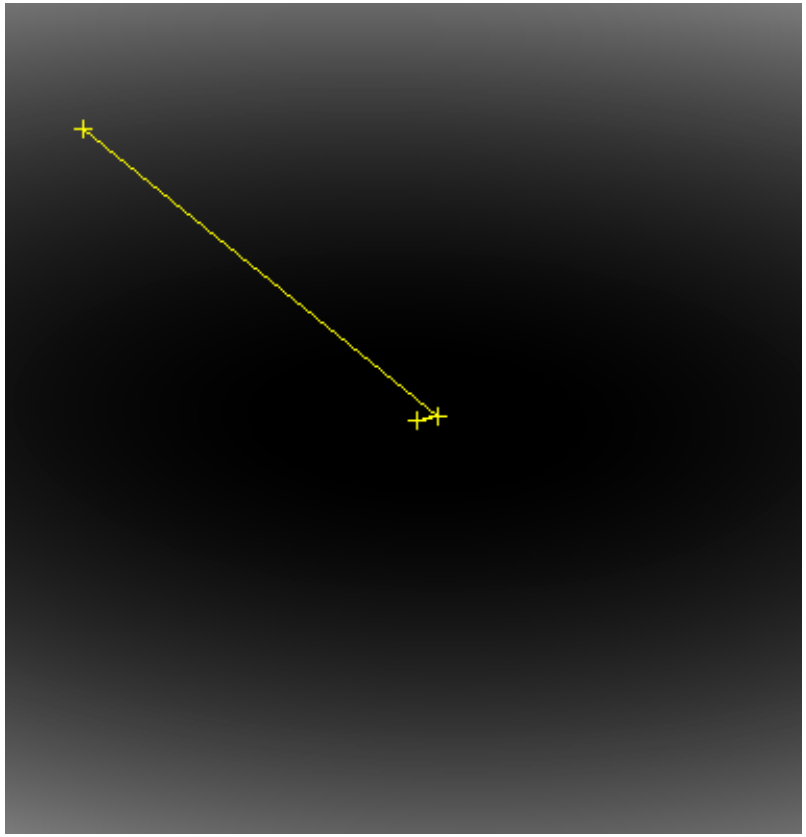
LVM: $(\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}) d\mathbf{X} = -\mathbf{g}$

- In practice, LVM and Damped Newton behave similarly:
  - when the residuals are small (good initialization),
  - when the non-linearities are not too extreme.
- LVM does not require the Hessian and is often preferred for that reason.

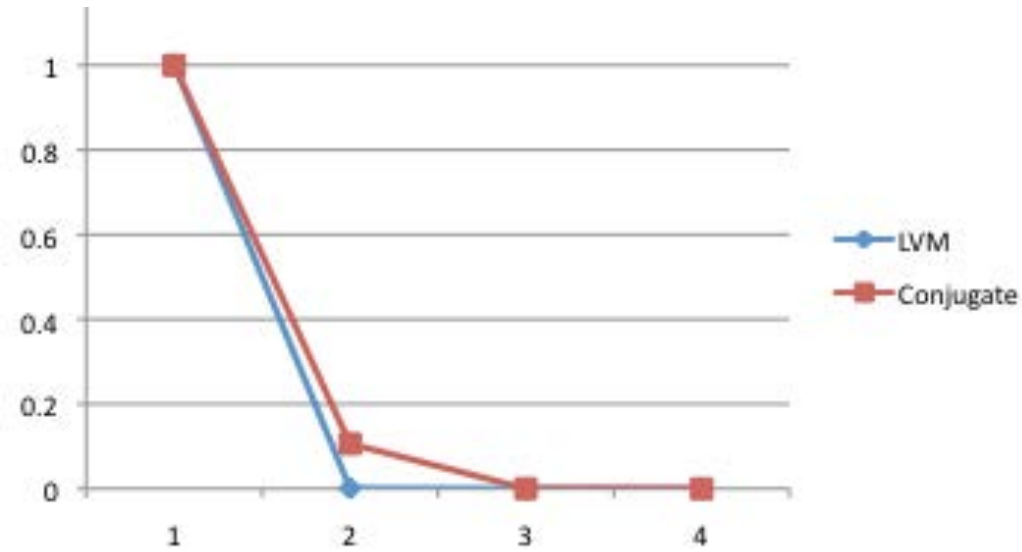
Python Code

```
class LvmOptim:

    def computeStep(self):

        X=self.X
        F,J=self.dataJac(X)                    #Compute current residuals and jacobian
        residF=norm(F)

        for i in range(40):
            dX=lsqSolve(J.T*J,J.T*F,self.lbda)  #Solve the linear system
            newF,_=self.dataJac(X-dX);          #Recompute residuals after step dX
            if(norm(newF)<residF):              #The objective function has improved
                self.lbda/=10                   #Decrease λ for the next iteration
                return(-dX)
            else:
                self.lbda*=10                   #Otherwise increase λ and try again

        return None                             #Failed to find a useful step
```

Optimization Path



Conjugate LVM



About half as many iterations as when using conjugate gradient when n=2.

In Real Life

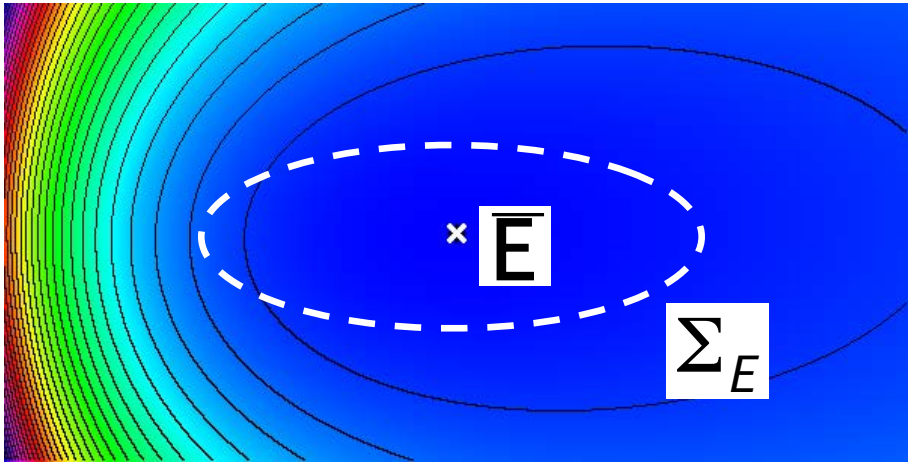
```
from scipy.optimize import least_squares
```

```
x1 = least_squares(func,x0,jacob,bounds=([-np.inf,1.5],np.inf))
```

#Function that computes F

#Function that computes J

Gaussian Noise



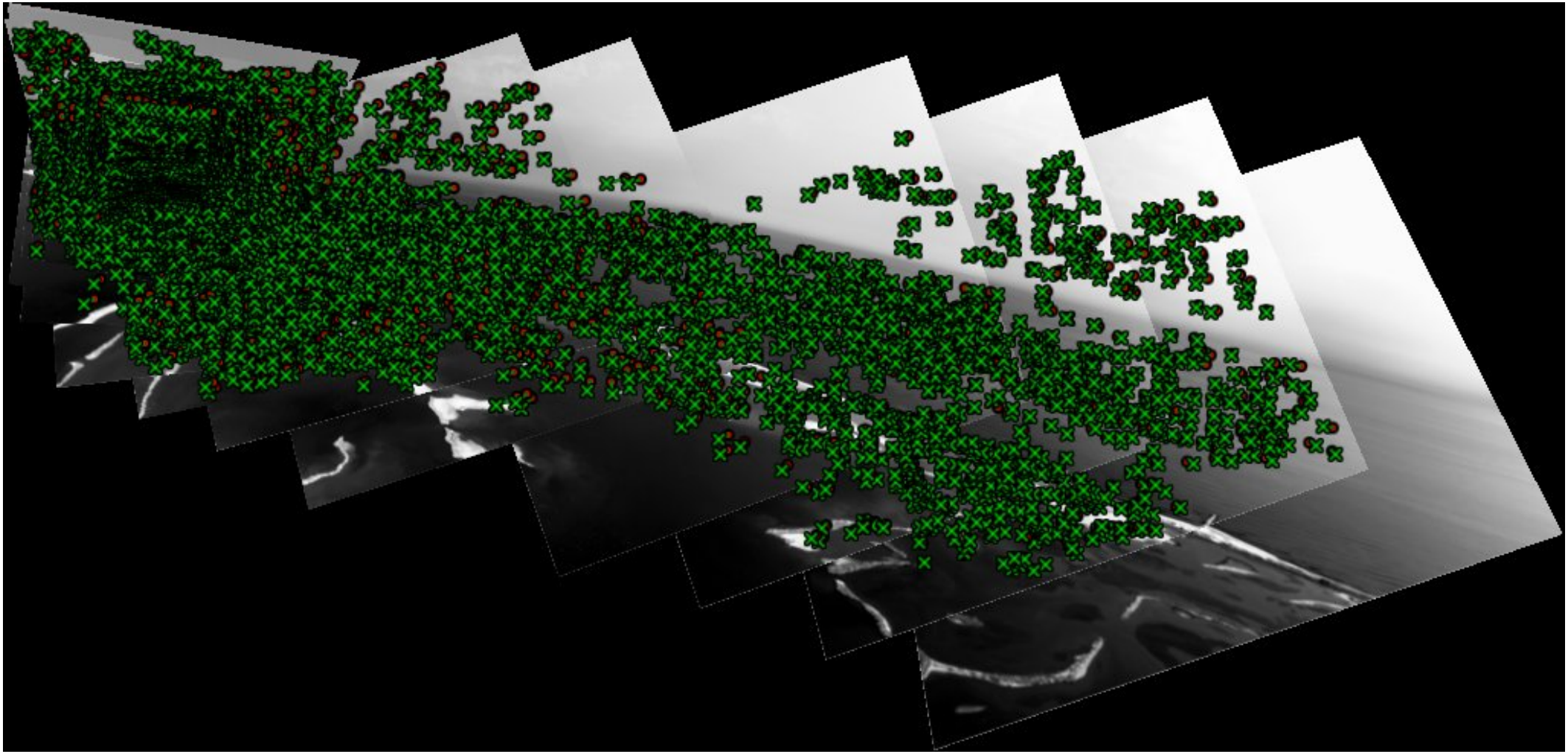
Assuming the image measurements

$$[\mathbf{x}_1, \ldots, \mathbf{x}_m]$$

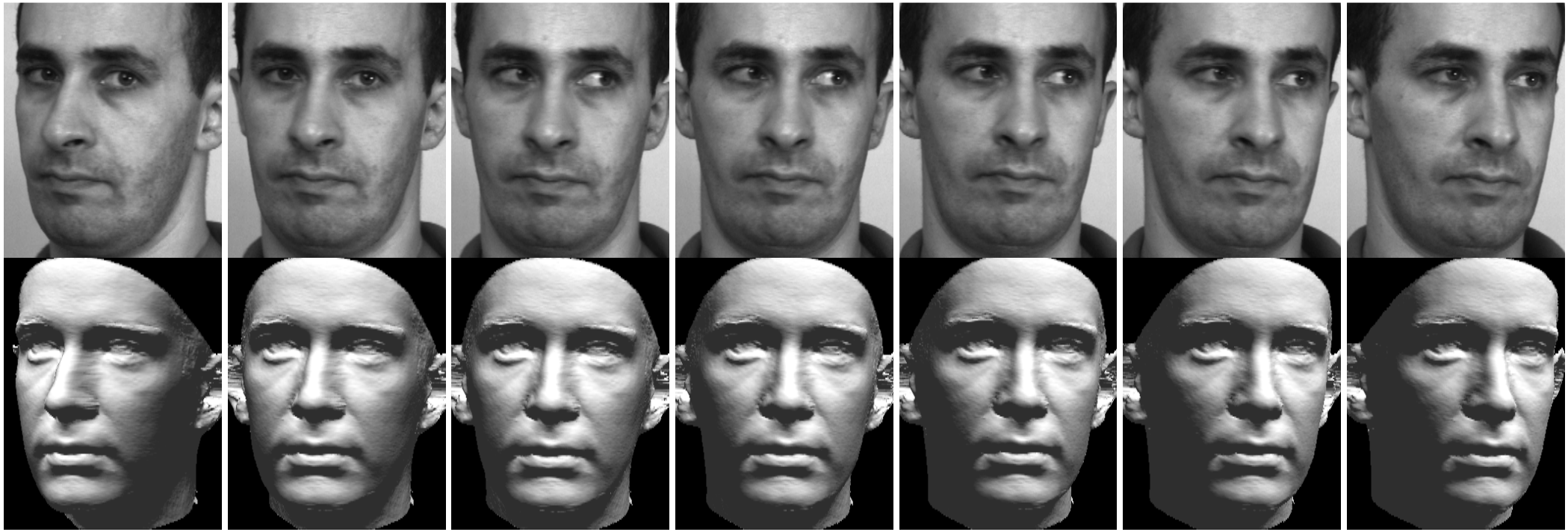
to be Gaussian with covariance $\Sigma_{\mathbf{x}}$, the optimal estimate is also Gaussian with mean $\bar{\mathbf{E}}$ and variance

$$\Sigma_E = (\mathbf{J}^T \Sigma_{\mathbf{x}}^{-1} \mathbf{J})^+ \ .$$

Global Bundle Adjustment



… Reconstructing his Face



Least squares fitting, also known as bundle adjustment, requires estimating:

- the camera parameters;
- the 3D shape parameters.

PCA Face Model

$$S = \bar{S} + \sum_{i=1}^{99} a_i S_i$$

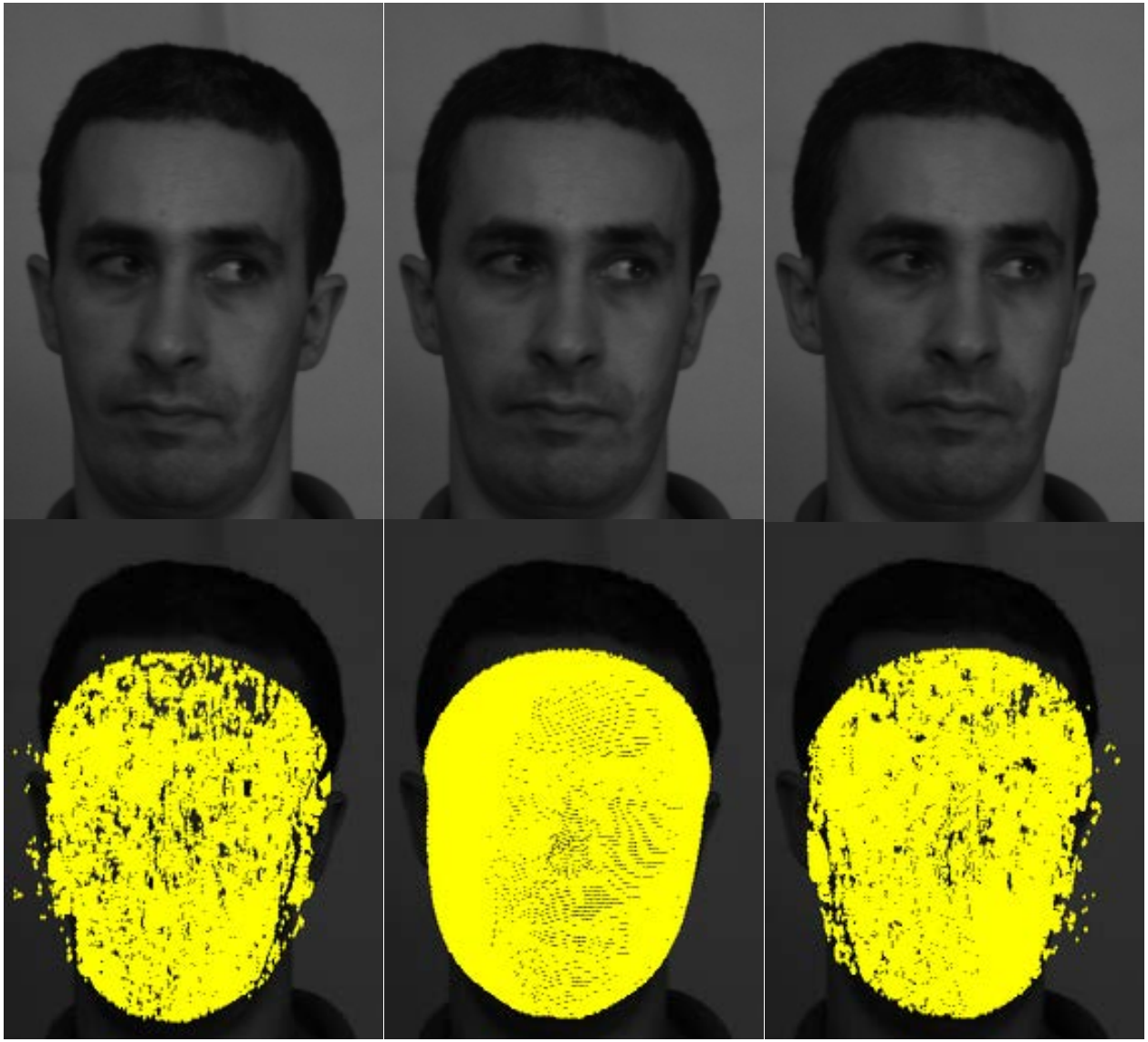
$\bar{S}$: Average shape

S_i : Shape vector

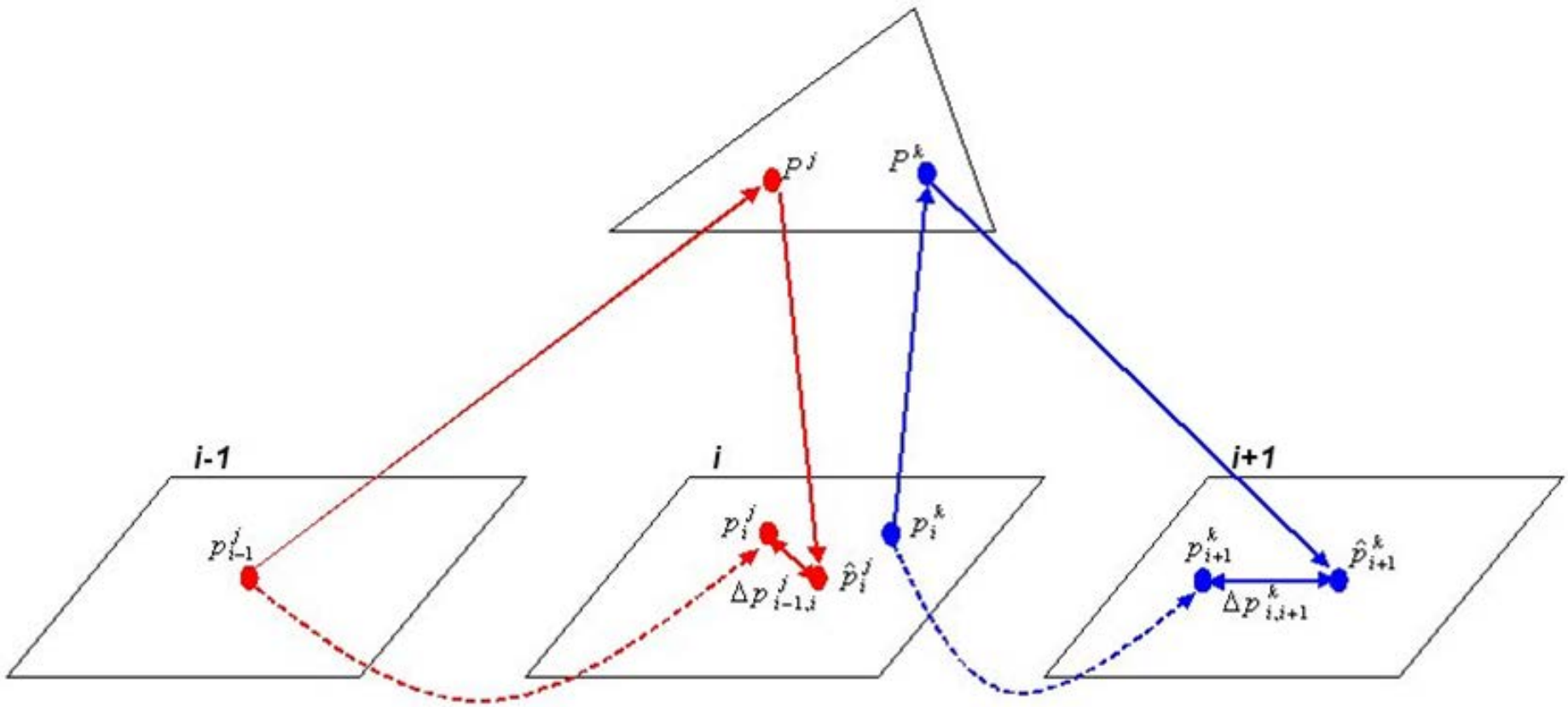
a_i : Shape coefficients

Blanz&Vetter, SIGGRAPH'99

Correspondences

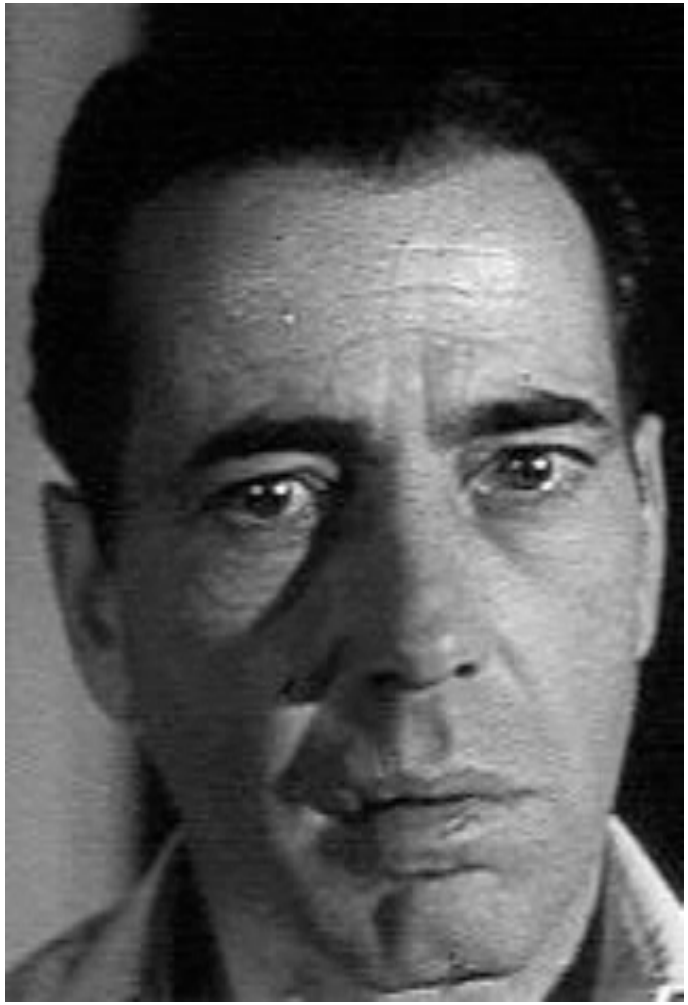


Transfer Function

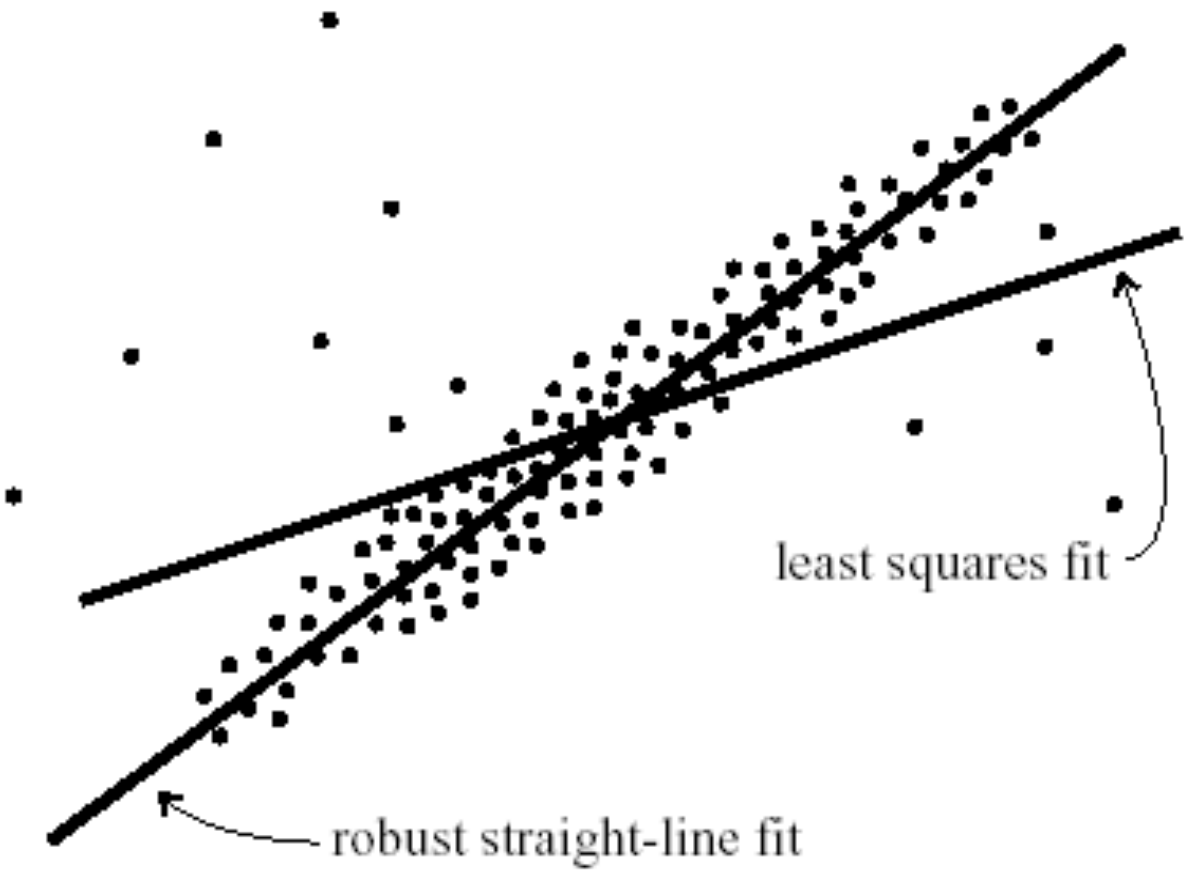


$$F_3(A, C_{i-1}, C_i, C_{i+1}) = \sum_{j \in Q_{i-1}} \left\| \Delta p_{i-1,i}^j \right\|^2 + \sum_{k \in Q_i} \left\| \Delta p_{i,i+1}^k \right\|^2$$

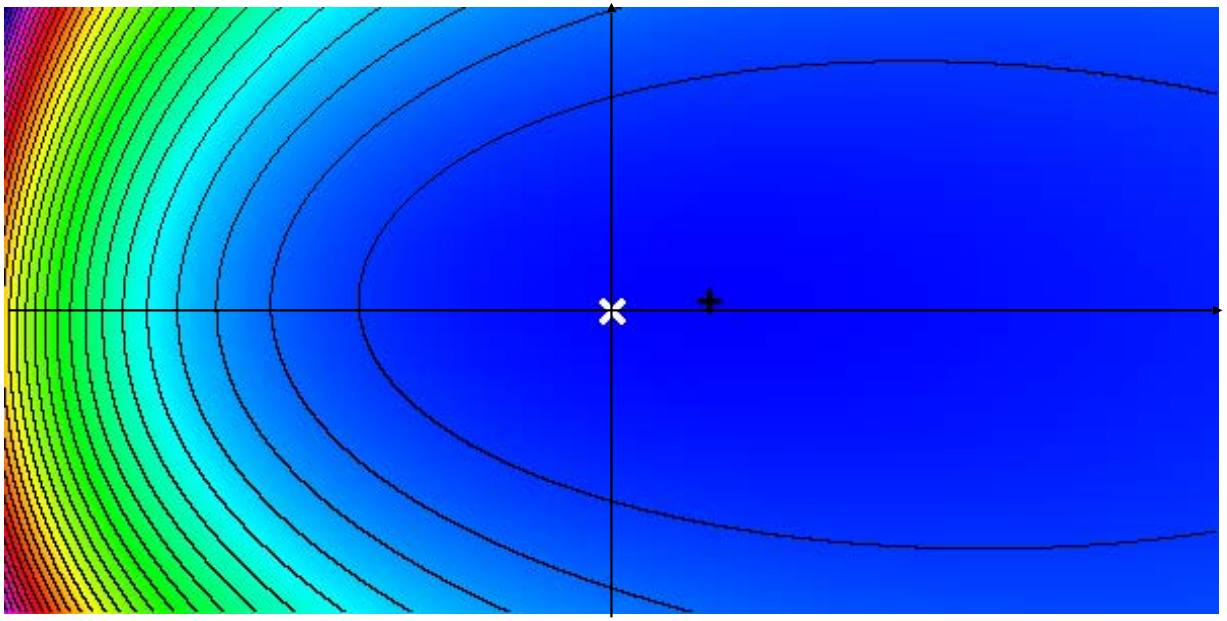
Model From Old Movie



Robust Estimation



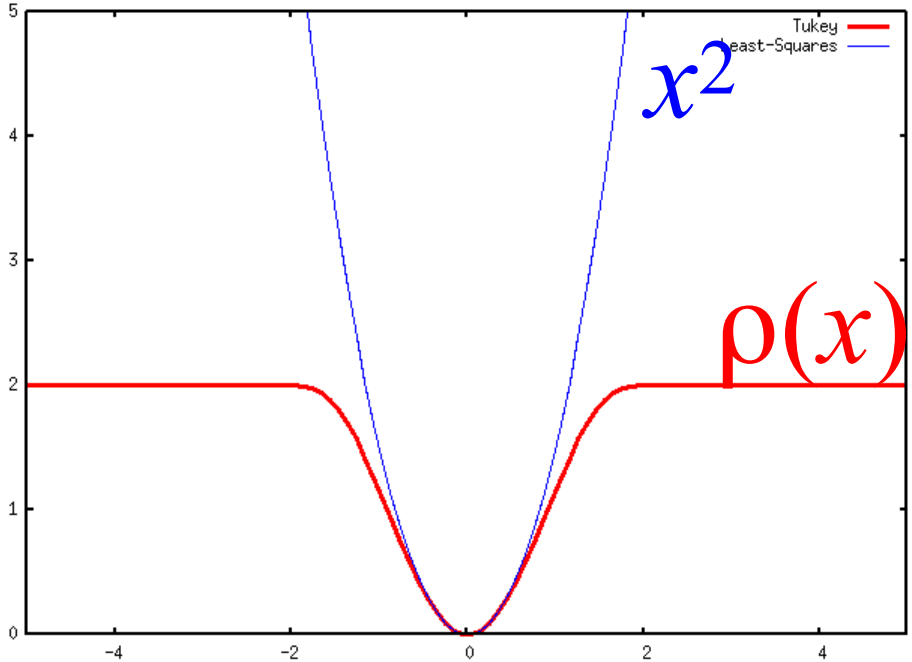
Noisy Correspondences



- Gaussian noise on the projections and 20% outliers.

→ The least-squares solution (black cross) is now far from the true camera position (white cross).

Tukey Estimator



$$\rho(d) = \begin{cases} \text{if } |d| \leq c & \frac{c^2}{6}(1-(1-\frac{d}{c}^2)^3) \\ \text{if } |d| > c & \frac{c^2}{6} \end{cases}$$

The Tukey estimator assumes the observables follow a distribution that is a mixture of:

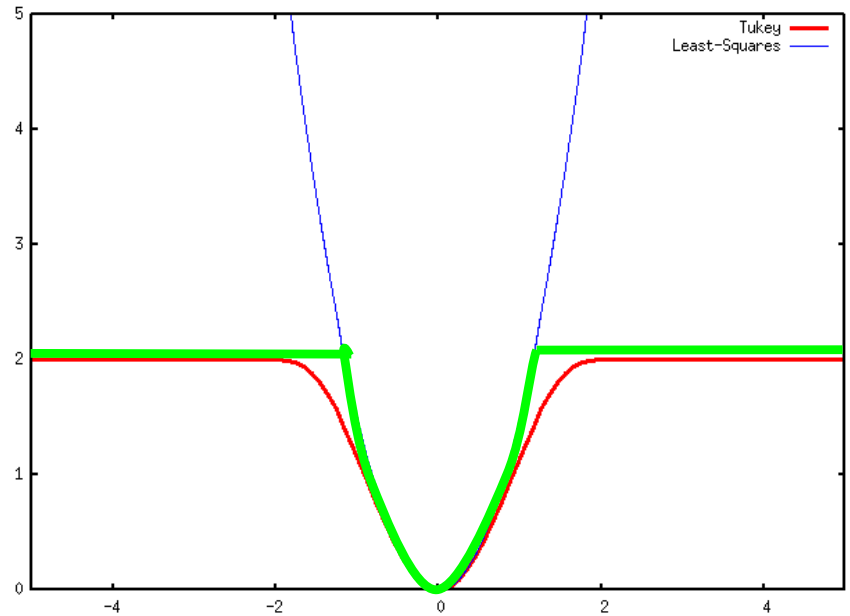
- a Normal distribution, for the inliers,
- a uniform distribution, for the outliers.

Tukey and LVM

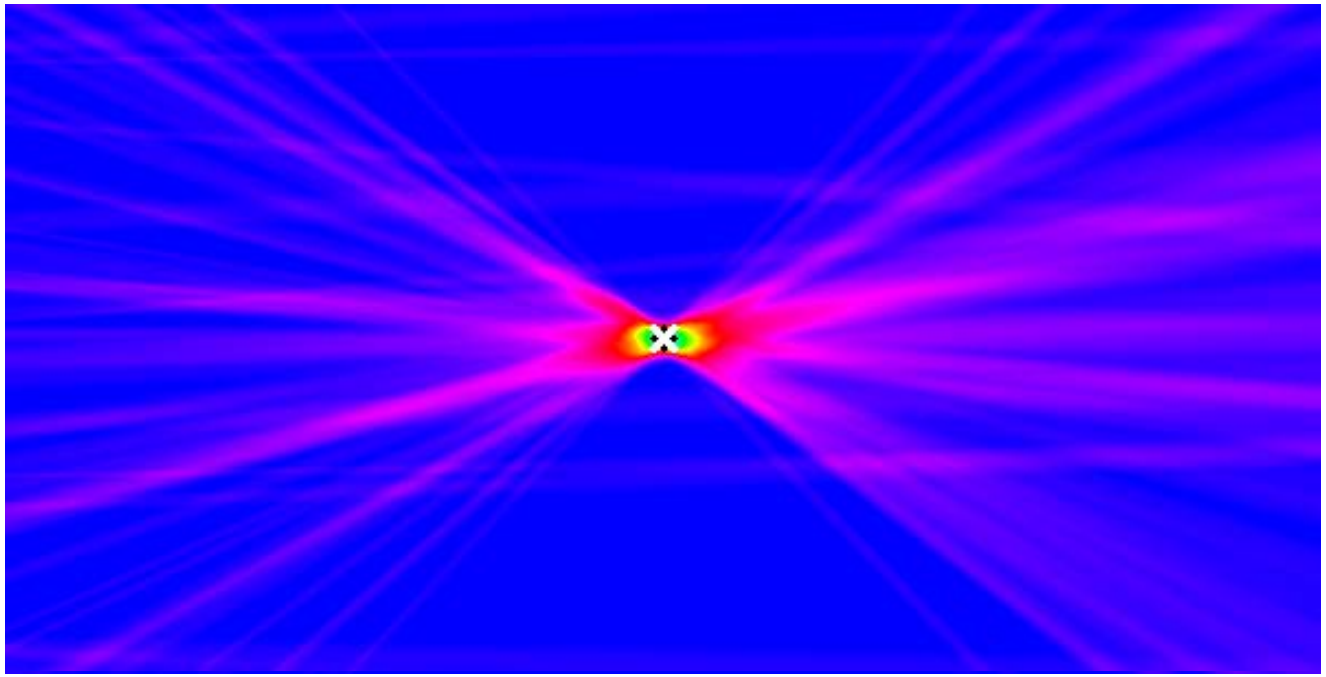
The Levenberg-marquardt algorithm requires the objective function to be a sum of squares.

→ Use the following approximation of the Tukey estimator.

$$\begin{cases} \text{if } |x| \le \tilde{c} & r(x) = \frac{1}{2}x^2 \\ \text{if } |x| > \tilde{c} & r(x) = \frac{1}{2}\tilde{c}^2 \end{cases}$$



Pros and Cons of the Turkey Operator



20% outliers.

50% outliers.

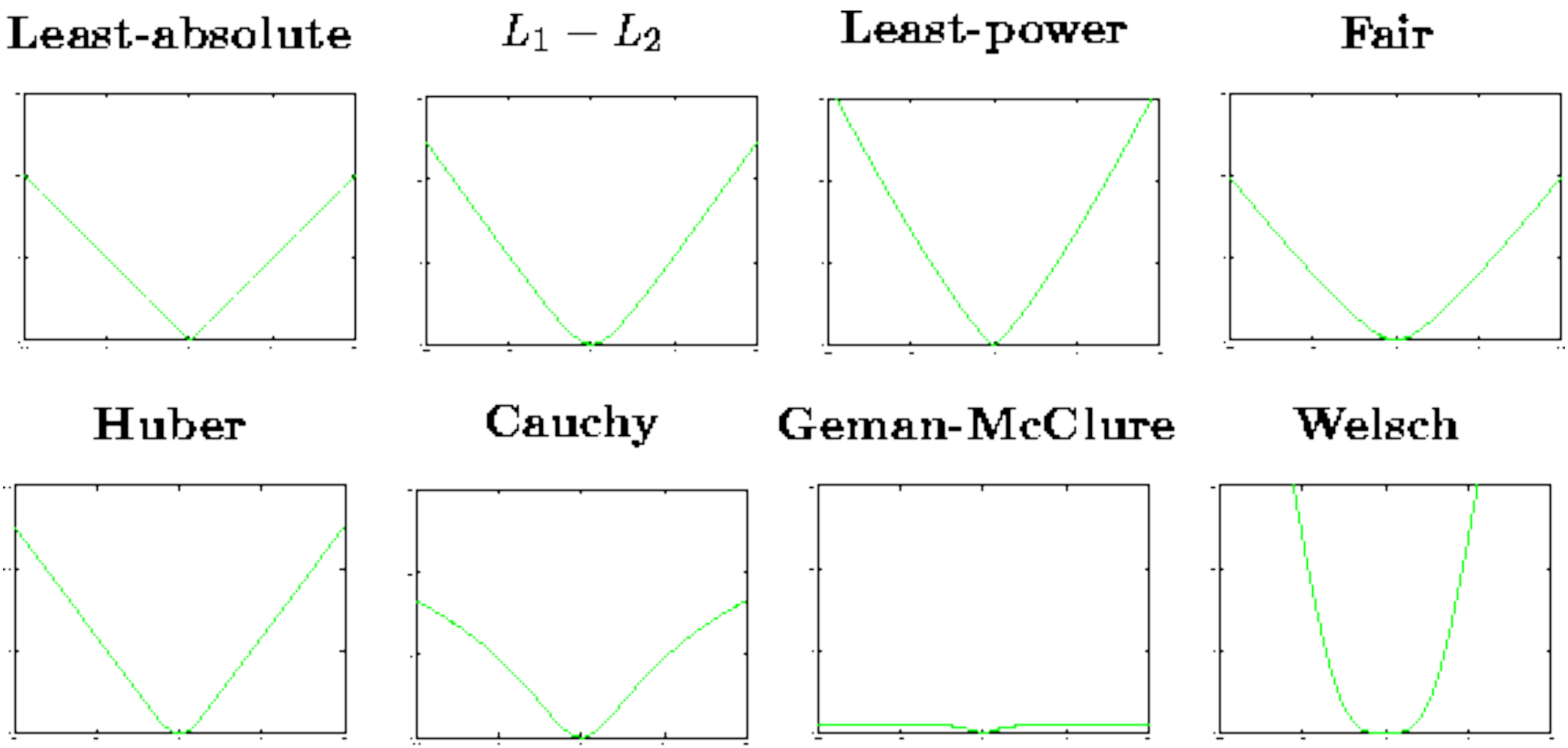
Pros:

- The global minimum is very close to the true one.

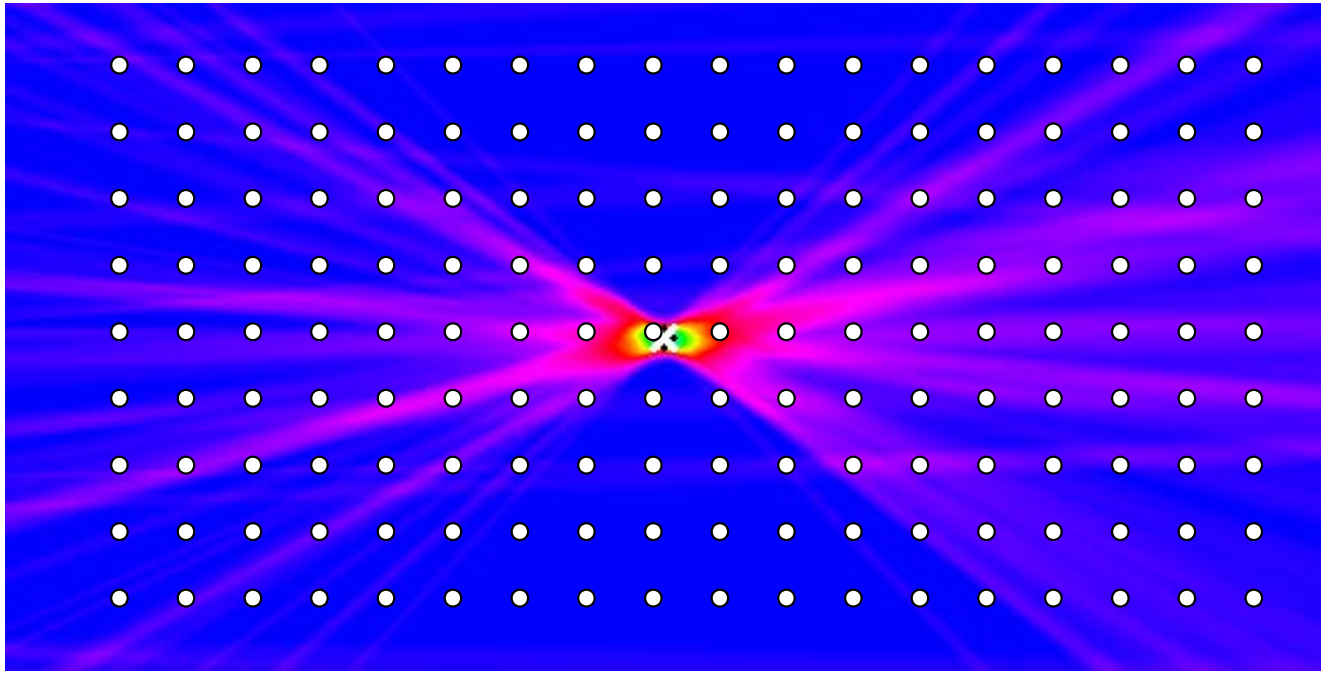
Cons:

- Many local minima
- The objective function is flat when too far from the optimum.

Other Robust Estimators



How to Optimize

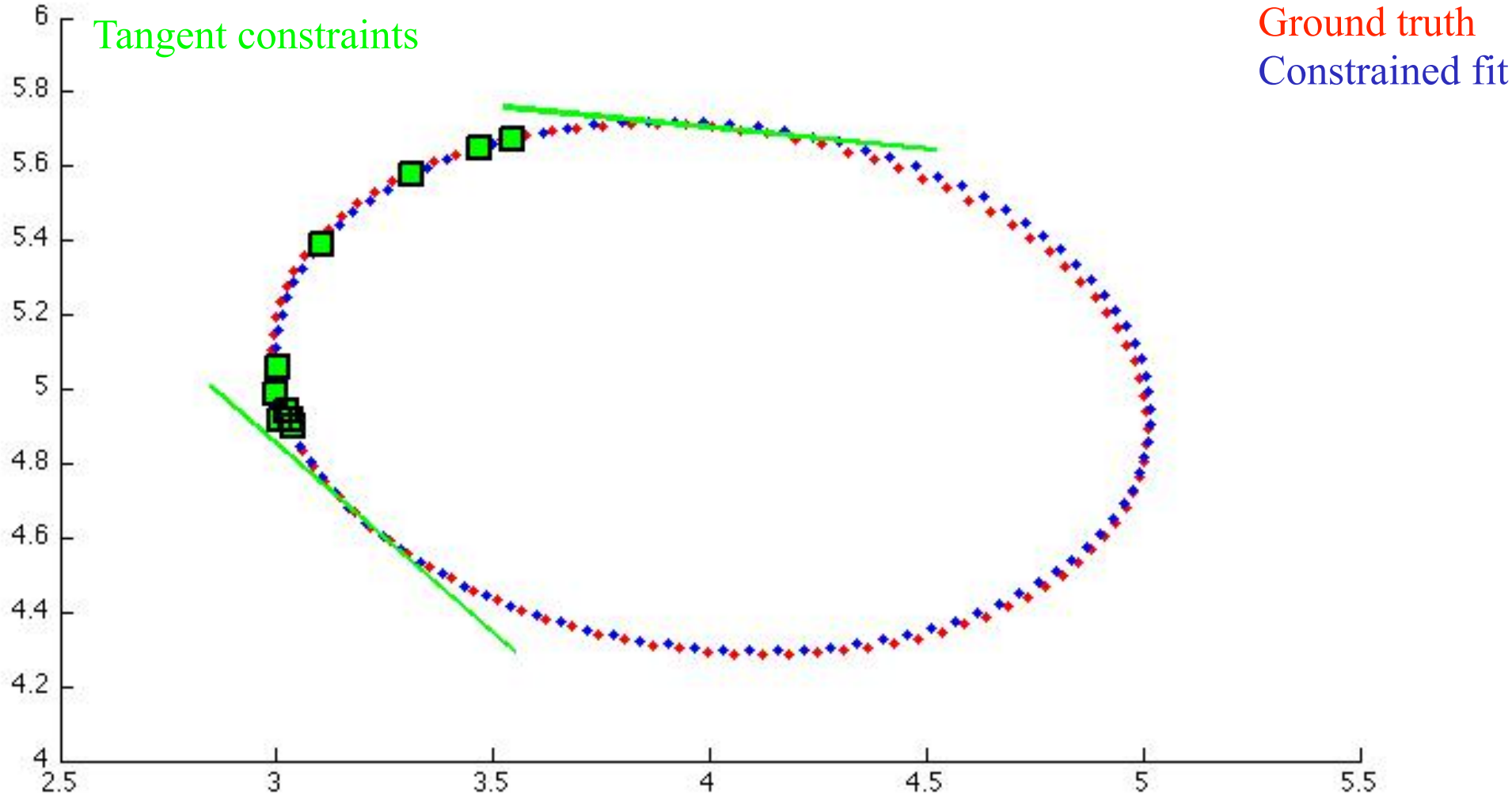


- Sample parameter space uniformly.
- Perform LVM starting from each sample.

—>Infeasible in high dimensional spaces.

—>Find heuristics to initialize (e.g. Hough).

Equality Constraints



$$\text{Minimize } \|F(\mathbf{X})\|^2 \text{ subject to } C(\mathbf{X}) = 0.$$

Constrained LVM

Regular LVM:

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})d\mathbf{X} = -\mathbf{g}$$

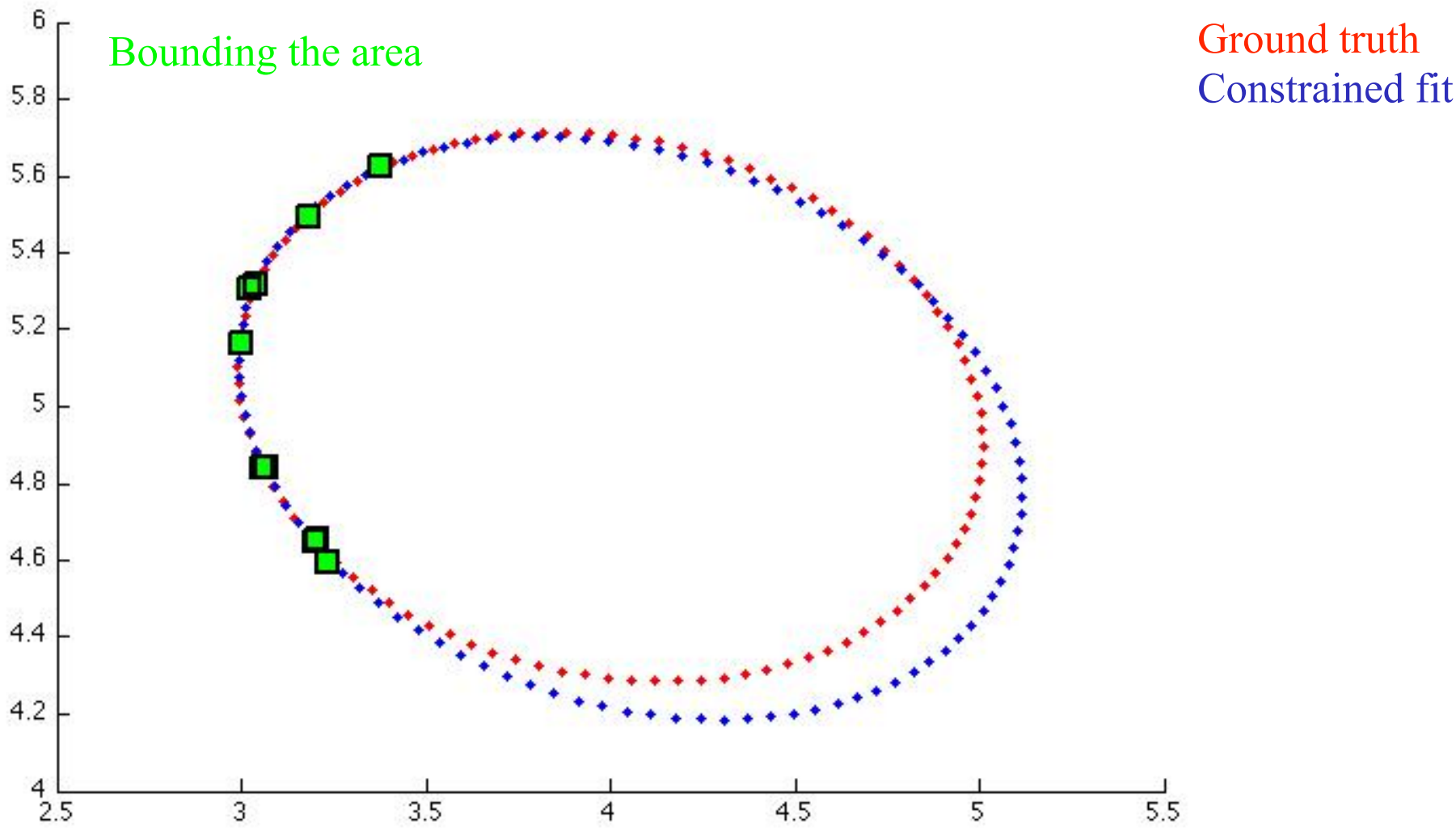
Constrained LVM:

$$\begin{bmatrix} (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}) & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} d\mathbf{X} \\ \Lambda \end{bmatrix} = -\begin{bmatrix} \mathbf{g} \\ C(\mathbf{X}) \end{bmatrix}$$

Lagrange multipliers

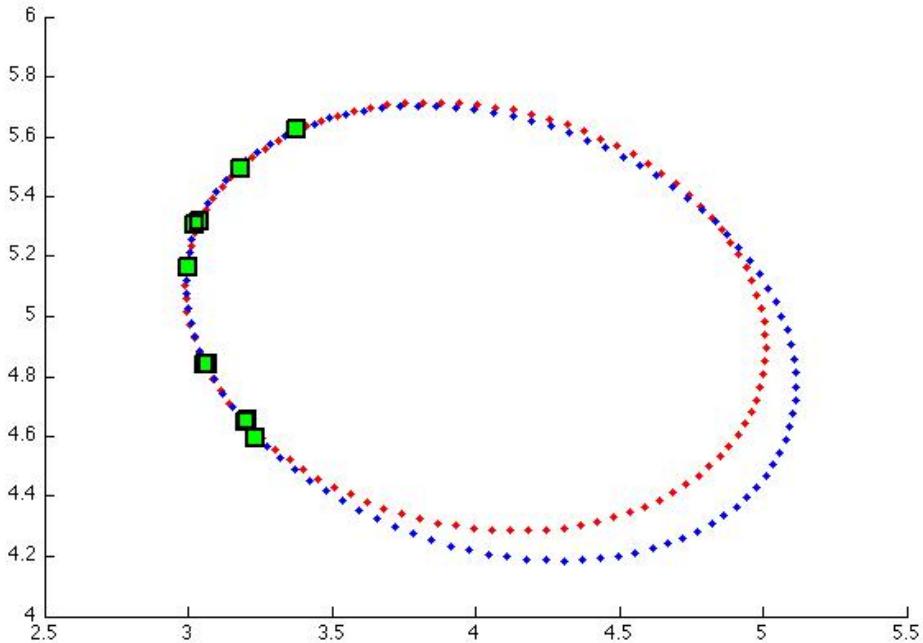
$$\mathbf{A} = \begin{bmatrix} \frac{\delta A_1}{\delta \mathbf{X}_1} & \cdots & \frac{\delta A_1}{\delta \mathbf{X}_n} \\ \cdots & \cdots & \cdots \\ \frac{\delta A_c}{\delta \mathbf{X}_1} & \cdots & \frac{\delta A_c}{\delta \mathbf{X}_n} \end{bmatrix}$$

Inequality Constraints

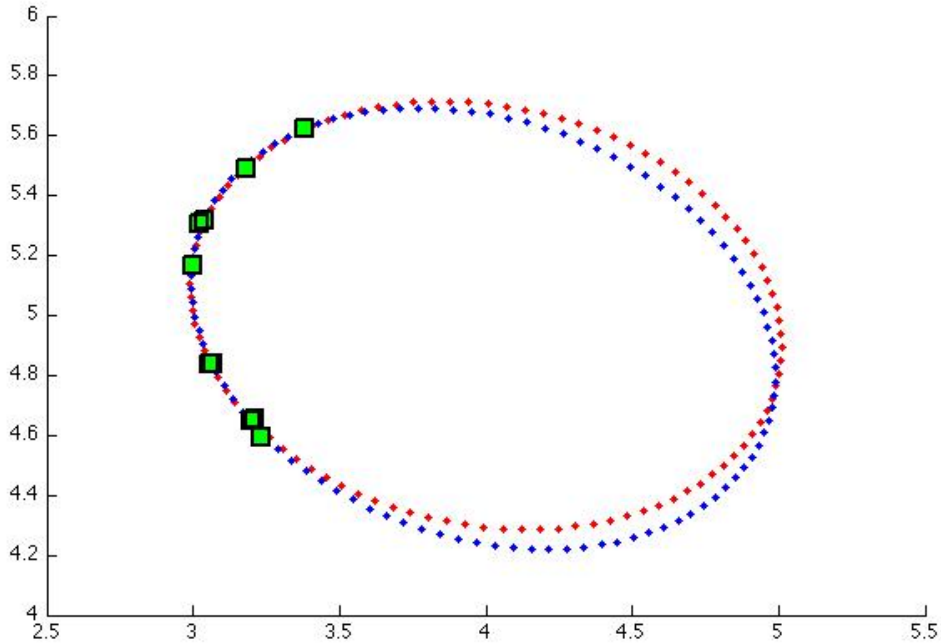


$$\text{Minimize } \|F(\mathbf{X})\|^2 \text{ subject to } C(\mathbf{X}) \leq 0.$$

Tightening the Bounds

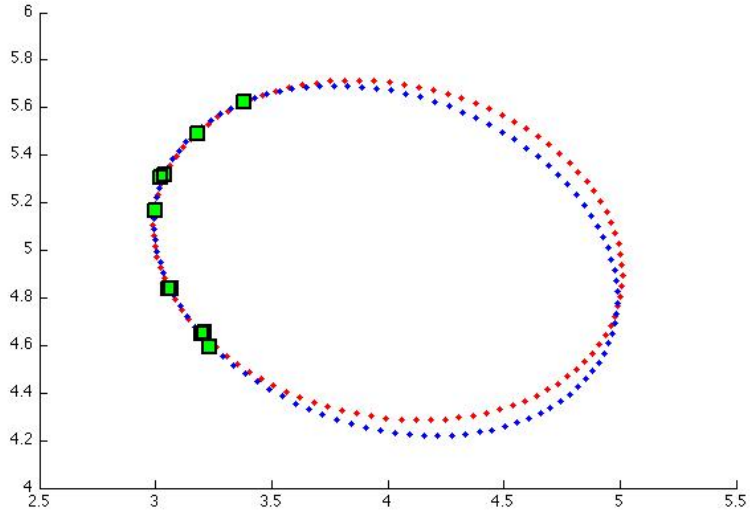


Loose Area Bound

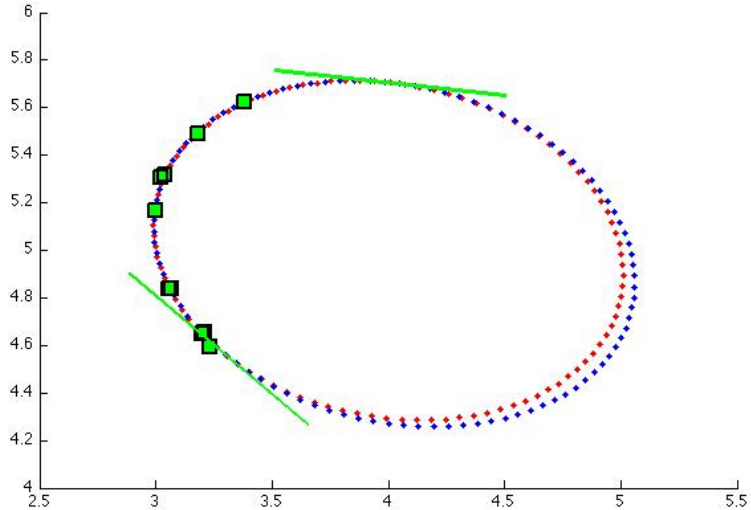


Tighter Area Bound

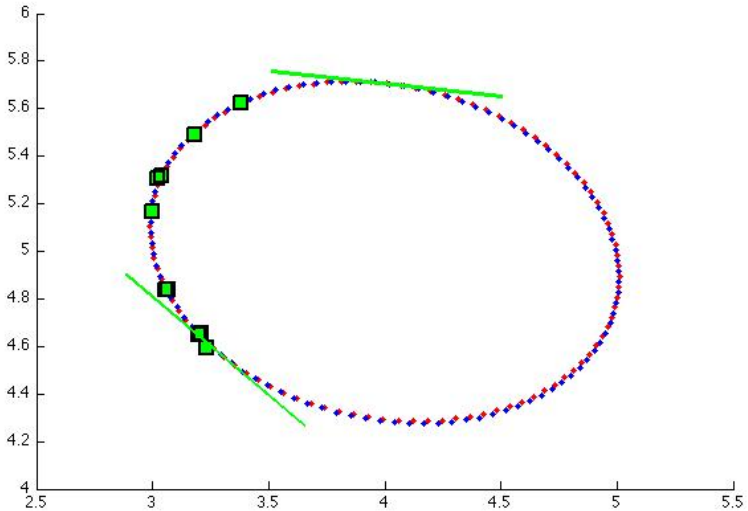
Combining Equalities and Inequalities



Area only

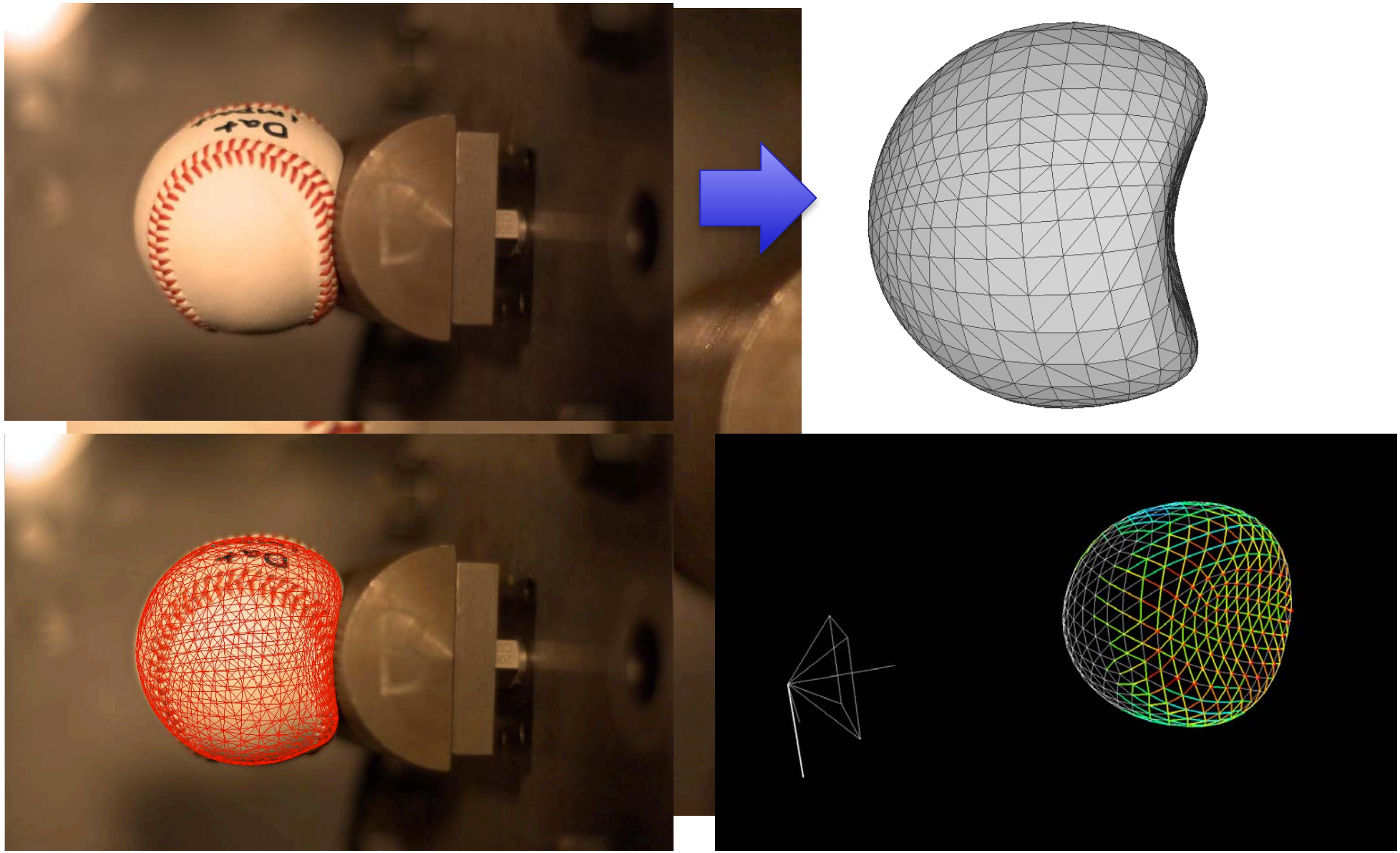


Tangent only

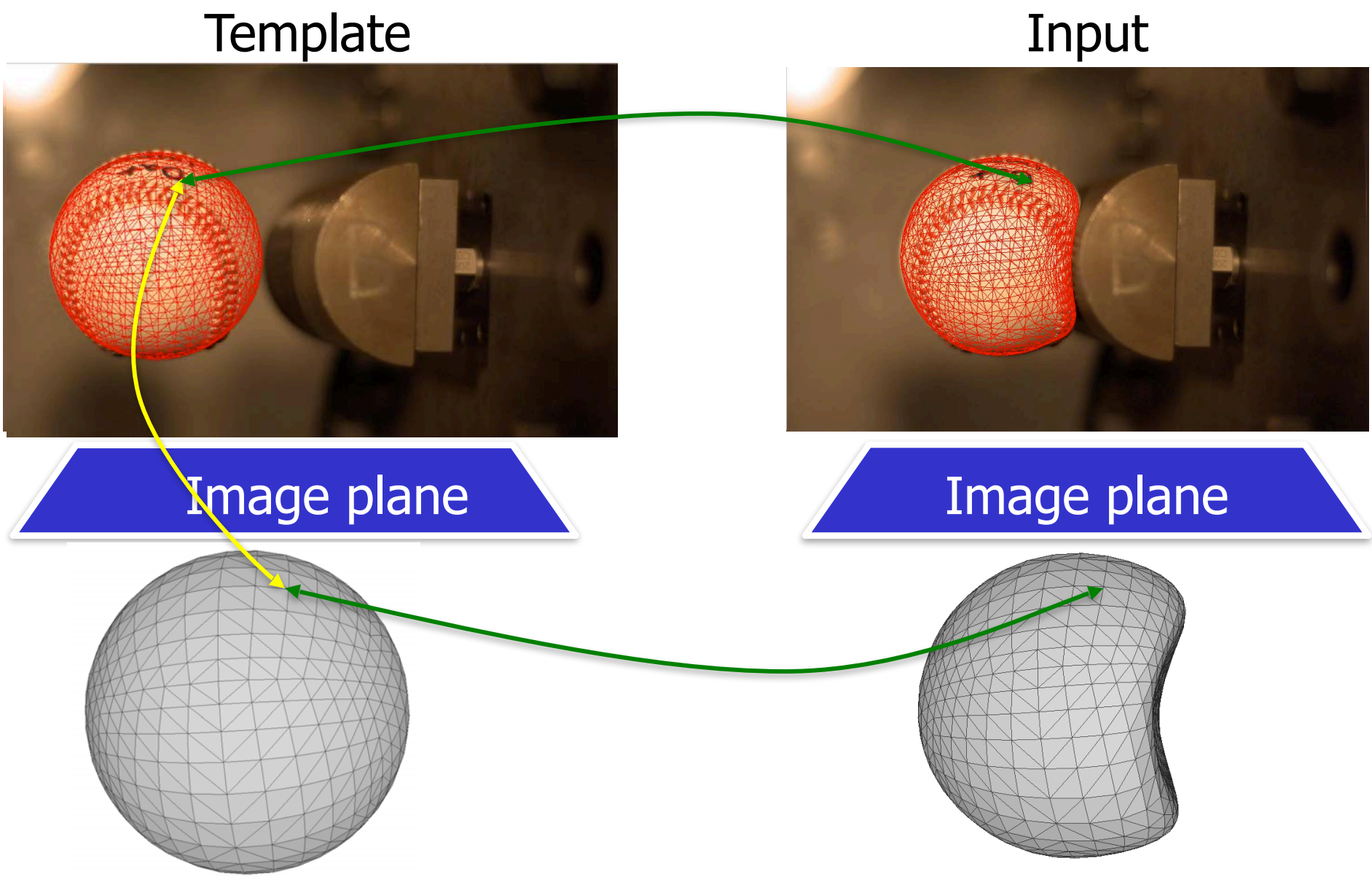


Both

Baseball



Baseball



Problem Formulation

Image re-projection

Regularization

Edge length penalty

$$\text{Minimize } \|\mathbf{MX}\|^2 + \alpha\|\mathbf{AX}\|^2 + \beta \sum_{(i,j)\in\text{edges}} (l_{ij} - l_{ij}^0)^2 \text{ w.r.t } \mathbf{X}, \text{ subject to}$$

$$\forall i, d(\mathbf{X}_i, \mathrm{C}) \geq \text{radius} ,$$

with:

- $\mathbf{X}$ 3D coordinates of all vertices,
- $\mathbf{X}_i$ 3D coordinates of vertex i,
- $l_{ij} = \|\mathbf{X}_i - \mathbf{X}_j\|$,
- C axis of the cylinder.

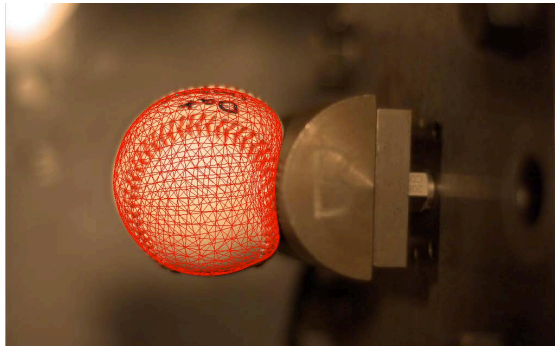
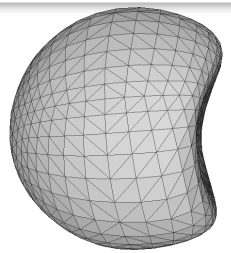
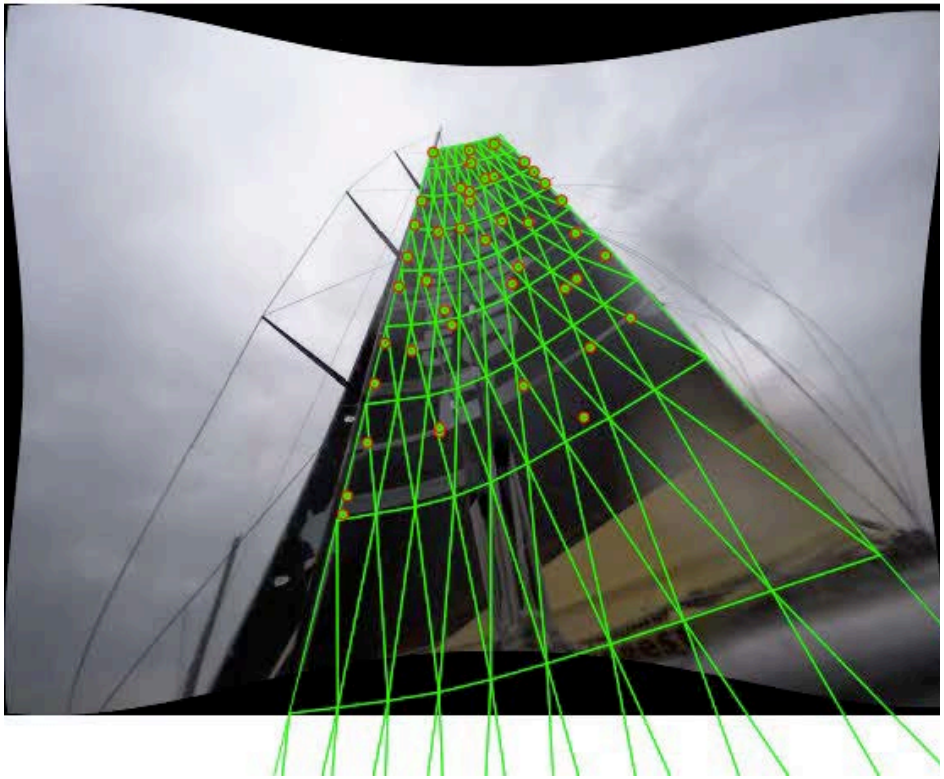


Image plane



SAIL DEFORMATIONS



Distances between mesh vertices must remain constant.

In Short

- LVM-style algorithm allow fast convergence when the residuals are not too non-linear.
- Robust estimators are essential to deal with noise and errors.
- Hard constraints can be added to further increase robustness.