6.1 2D Feature-Based Alignment

Outline

- Linear Alignment Problems
- Non-Linear Alignment Problems

Outline

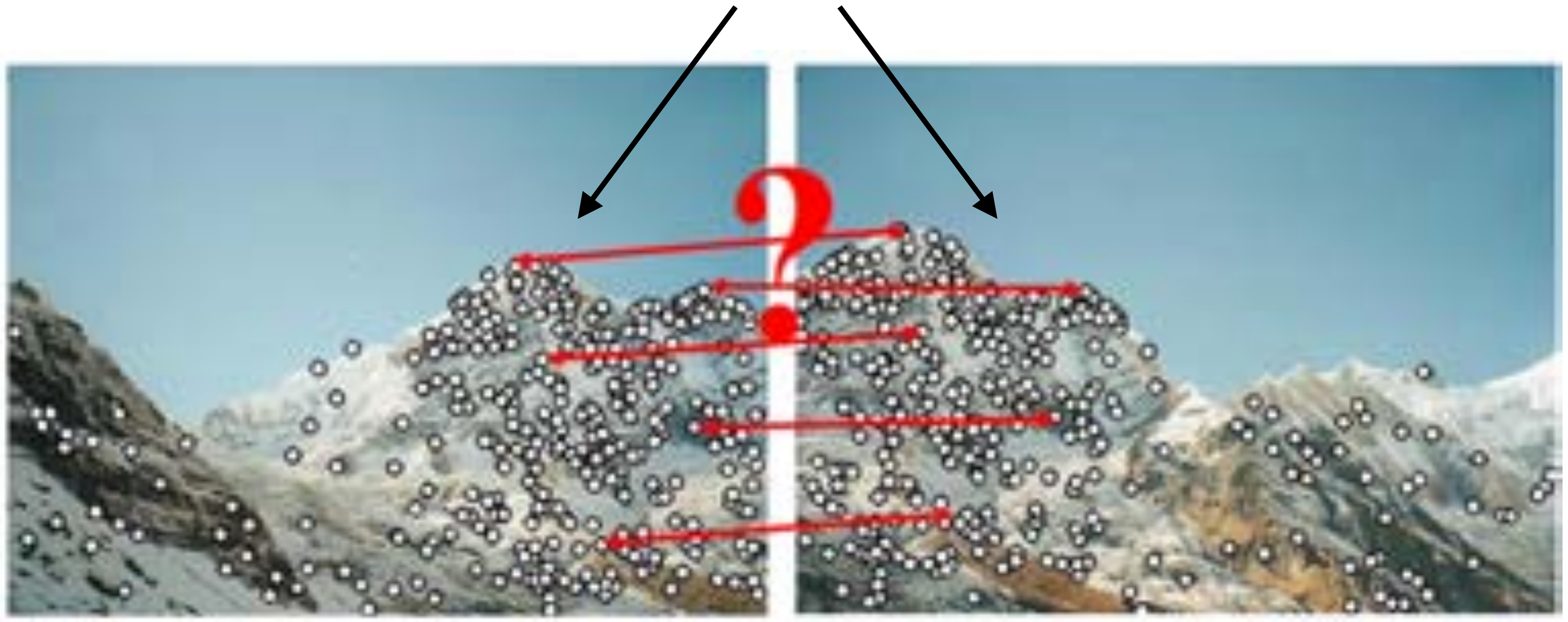
- **Linear Alignment Problems**

- Non-Linear Alignment Problems

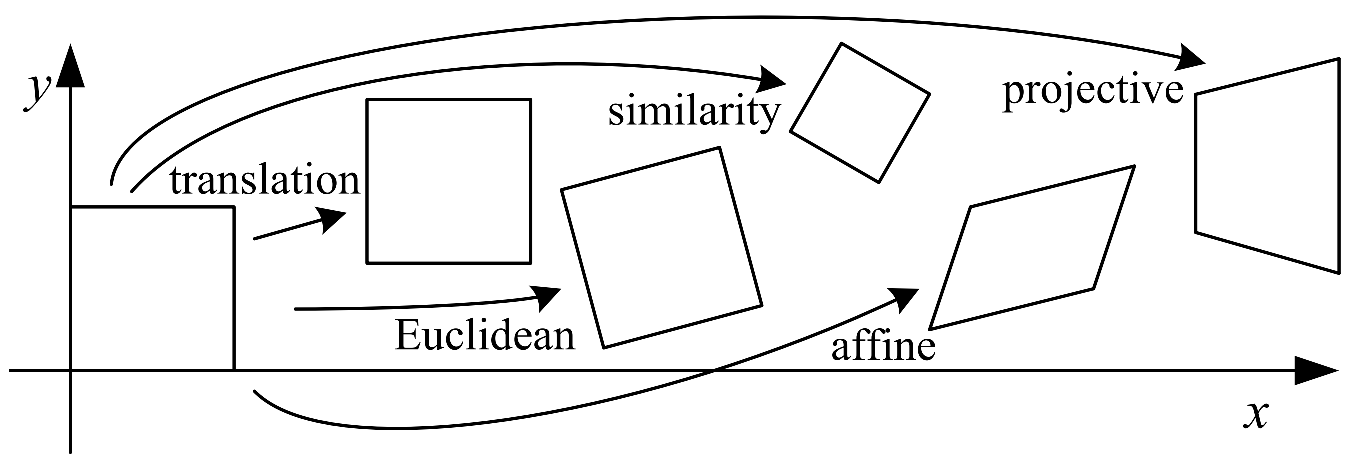
Global Parametric Spatial Transformations

- We assume a set of matched 2D points in two images of the same object or scene.
- How can we determine the global parametric spatial transformation $\boldsymbol{f}$ that relates them?

$$\{(\boldsymbol{x}_i, \boldsymbol{x}'_i)\}$$



$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$$



Least Squares

- If $\boldsymbol{f}$ in fact captures the true relationship between the matched points aside from additive Gaussian iid noise, then the maximum likelihood solution is to minimize the sum of squared residuals:

$$E_{\mathrm{LS}} = \sum_i \|\boldsymbol{r}_i\|^2 = \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i\|^2,$$

where

$$\boldsymbol{r}_i = \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i = \hat{\boldsymbol{x}}'_i - \tilde{\boldsymbol{x}}'_i$$

Linear Transformations

- For some simple global transformations, the amount of motion $\Delta\boldsymbol{x} = \boldsymbol{x}' - \boldsymbol{x}$ is a linear function of the parameters $\boldsymbol{p}$, mediated by the Jacobian $\boldsymbol{J}(\boldsymbol{x})$ of the transformation $\boldsymbol{f}$ with respect to the motion parameters $\boldsymbol{p}$:

$$\Delta\boldsymbol{x} = \boldsymbol{x}' - \boldsymbol{x} = \boldsymbol{J}(\boldsymbol{x})\boldsymbol{p},$$

where

$$\boldsymbol{J}(x) = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial x'}{\partial p_1} & \frac{\partial x'}{\partial p_2} & \cdots & \frac{\partial x'}{\partial p_n} \\ \frac{\partial y'}{\partial p_1} & \frac{\partial y'}{\partial p_2} & \cdots & \frac{\partial y'}{\partial p_n} \end{bmatrix}$$

Transform	Matrix	Parameters p	Jacobian J
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$

Operate on augmented vector $\bar{\boldsymbol{x}}$

End of Lecture
Nov 12, 2018

Linear Regression Framework

$$
\begin{aligned}
E_{\mathrm{LLS}} &= \sum_i \|\boldsymbol{J}(\boldsymbol{x}_i)\boldsymbol{p} - \Delta\boldsymbol{x}_i\|^2 \\
&= \boldsymbol{p}^T \left[\sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i)\right]\boldsymbol{p} - 2\boldsymbol{p}^T\left[\sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\Delta\boldsymbol{x}_i\right] + \sum_i \|\Delta\boldsymbol{x}_i\|^2 \\
&= \boldsymbol{p}^T\boldsymbol{A}\boldsymbol{p} - 2\boldsymbol{p}^T\boldsymbol{b} + c.
\end{aligned}
$$

- To minimize, we set the derivative with respect to the parameters $\boldsymbol{p}$ to 0, yielding

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b},$$

where

$$\boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i)$$

and

$$\boldsymbol{b} = \sum_i \boldsymbol{J}^T\left(\boldsymbol{x}_i\right)\Delta\boldsymbol{x}_i$$

Linear Regression Framework

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b},$$

where

$$\boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad \boldsymbol{b} = \sum_i \boldsymbol{J}^T\left(\boldsymbol{x}_i\right)\Delta\boldsymbol{x}_i$$

- Observations:
  - $\boldsymbol{A}$ is symmetric.
  - $\boldsymbol{A}$ is non-negative definite

    Consider a non-zero parameter vector $\boldsymbol{p}$.

    Note that each term $\boldsymbol{p}^\top \boldsymbol{A}_i \boldsymbol{p}$ of $\boldsymbol{p}^\top \boldsymbol{A}\boldsymbol{p}$ is non-negative:

    $\boldsymbol{p}^\top \boldsymbol{A}_i \boldsymbol{p} = \boldsymbol{p}^\top \boldsymbol{J}^\top\left(\boldsymbol{x}_i\right)\boldsymbol{J}\left(\boldsymbol{x}_i\right)\boldsymbol{p} = \left\|\boldsymbol{J}\left(\boldsymbol{x}_i\right)\boldsymbol{p}\right\|^2$
  - Is $\boldsymbol{A}$ positive definite?

    $\boldsymbol{p}^\top \boldsymbol{A}\boldsymbol{p} > 0$ as long as at least one term $\boldsymbol{p}^\top \boldsymbol{A}_i \boldsymbol{p} \neq 0 \leftrightarrow \boldsymbol{J}\left(\boldsymbol{x}_i\right)\boldsymbol{p} \neq \boldsymbol{0}$.

    Thus $\boldsymbol{A}$ is positive definite as long as $\boldsymbol{J}\left(\boldsymbol{x}_i\right)$ has full rank for at least one point $\boldsymbol{x}_i$.

Rank of the Jacobian

$\boldsymbol{A}$ is positive definite as long as $\boldsymbol{J}\left(\boldsymbol{x}_i\right)$ has full rank for at least one point $\boldsymbol{x}_i$.

$$\boldsymbol{J}(x_i) = \frac{\partial \boldsymbol{f}(\boldsymbol{x}_i)}{\partial \boldsymbol{p}} = \begin{bmatrix} \dfrac{\partial x_i'}{\partial p_1} & \dfrac{\partial x_i'}{\partial p_2} & \cdots & \dfrac{\partial x_i'}{\partial p_n} \\ \dfrac{\partial y_i'}{\partial p_1} & \dfrac{\partial y_i'}{\partial p_2} & \cdots & \dfrac{\partial y_i'}{\partial p_n} \end{bmatrix}$$

In other words, the influence of the parameters p_j on the point $\boldsymbol{x}_i$ must be linearly independent.

- This will generally be true if:
  - The parameters p_j are selected to control different aspects of the transformation
  - A diversity of points $\boldsymbol{x}_i$ are included

Linear Regression Framework: Solution

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b},$$

where

$$\boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad \boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\Delta\boldsymbol{x}_i$$

- A is symmetric and positive definite.

- Under these conditions, the best approach is usually **Cholesky decomposition**:

  $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^\top$

  where $\boldsymbol{L}$ is lower triangular with positive diagonal entries.

  MATLAB function chol(A)

  - ~twice as fast as LU decomposition

  - $O(n^3)$ to compute $\boldsymbol{L}$, where n is the size of $\boldsymbol{A}$.

Linear Regression Framework: Solution

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b}$$

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^\top$$

$$\longrightarrow \quad \boldsymbol{L}\boldsymbol{L}^\top \boldsymbol{p} = \boldsymbol{b} \quad \longrightarrow \quad \boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}, \text{ where } \boldsymbol{y} \triangleq \boldsymbol{L}^\top \boldsymbol{p}$$

where $\boldsymbol{L}$ is lower triangular with positive diagonal entries.

- First solve for $\boldsymbol{y}$ using forward substitution.
- Then solve for $\boldsymbol{p}$ using backward substitution.
- $O(n)$, where n is the size of $\boldsymbol{A}$.

Linear Regression Framework: MATLAB

$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b}$

MATLAB mldivide: $\boldsymbol{p} = \boldsymbol{A} \backslash \boldsymbol{b}$

- mldivide is very smart
  - It tests whether A is symmetric and positive definite.
  - If it is, it uses a Cholesky solver.

Example

Outline

- Linear Alignment Problems
- **Non-Linear Alignment Problems**

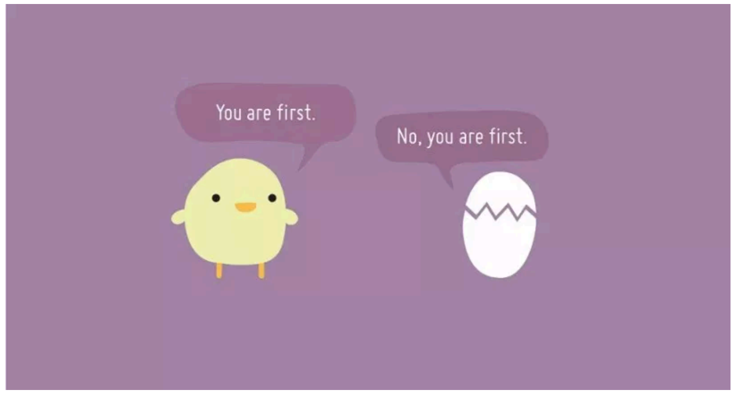
Non-Linear Alignment Problems

- Often the displacement is not in fact linear in the parameters.

- Example: Rigid 2D transformation (translation + rotation):

  - Note that the Jacobian is itself a function of the rotation parameter θ

Transform	Matrix	Parameters p	Jacobian J
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$



Iterative Alignment

- Non-linear alignment problems can be solved iteratively.
- Suppose that we start with a guess at the parameters $\boldsymbol{p}$.
- We can now formulate an estimate of the error that would result if we took a step $\Delta\boldsymbol{p}$ from this initial guess:

$$
\begin{aligned}
E_{\mathrm{NLS}}(\Delta\boldsymbol{p}) &= \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{p}+\Delta\boldsymbol{p}) - \boldsymbol{x}'_i\|^2 \\
&\approx \sum_i \|\boldsymbol{J}(\boldsymbol{x}_i;\boldsymbol{p})\Delta\boldsymbol{p} - \boldsymbol{r}_i\|^2 \qquad \text{where } \boldsymbol{r}_i \triangleq \boldsymbol{x}'_i - f\left(\boldsymbol{x}_i;\boldsymbol{p}\right) \\
&= \Delta\boldsymbol{p}^T\left[\sum_i \boldsymbol{J}^T\boldsymbol{J}\right]\Delta\boldsymbol{p} - 2\Delta\boldsymbol{p}^T\left[\sum_i \boldsymbol{J}^T\boldsymbol{r}_i\right] + \sum_i \|\boldsymbol{r}_i\|^2 \\
&= \Delta\boldsymbol{p}^T\boldsymbol{A}\Delta\boldsymbol{p} - 2\Delta\boldsymbol{p}^T\boldsymbol{b} + c,
\end{aligned}
$$

where again

$$
\boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad\qquad \boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{r}_i
$$

Iterative Alignment - The Gauss-Newton Method

$$E_{\mathrm{NLS}}(\Delta \boldsymbol{p}) \quad = \quad \Delta \boldsymbol{p}^T \boldsymbol{A} \Delta \boldsymbol{p} - 2\Delta \boldsymbol{p}^T \boldsymbol{b} + c, \qquad \boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad \boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{r}_i$$

- Taking a first derivative with respect to $\Delta \boldsymbol{p}$ and setting it to zero, we obtain

  $\boldsymbol{A}\Delta \boldsymbol{p} = b$

- This can again be solved by Cholesky decomposition (MATLAB \).

- This is called the Gauss-Newton method.

- But since our linear approximation only applies locally, this $\Delta \boldsymbol{p}$ may step past the minimum and is thus not guaranteed to lower the error.

- Solution 1. reduce the step size

  $\boldsymbol{p} \leftarrow \boldsymbol{p} + \alpha \Delta \boldsymbol{p}, \quad 0 < \alpha \leq 1$

Iterative Alignment - Levenberg-Marquardt

$$E_{\mathrm{NLS}}(\Delta \boldsymbol{p}) \quad = \quad \Delta \boldsymbol{p}^T \boldsymbol{A} \Delta \boldsymbol{p} - 2\Delta \boldsymbol{p}^T \boldsymbol{b} + c, \qquad \boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad \boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{r}_i$$

- Solution 2. Levenberg-Marquardt (damped Gauss-Newton)
  - Add a diagonal damping term:

    $$\left(\boldsymbol{A} + \lambda \boldsymbol{I}\right)\Delta \boldsymbol{p} = b$$

  - L-M can be seen as a mixture of Gauss-Newton and gradient descent.
  - λ adjusted according to how fast error is decreasing
    - Slow: still far from minimum - increase λ (upweight gradient descent)
    - Fast: getting close to minimum - reduce λ (upweight Gauss-Newton)

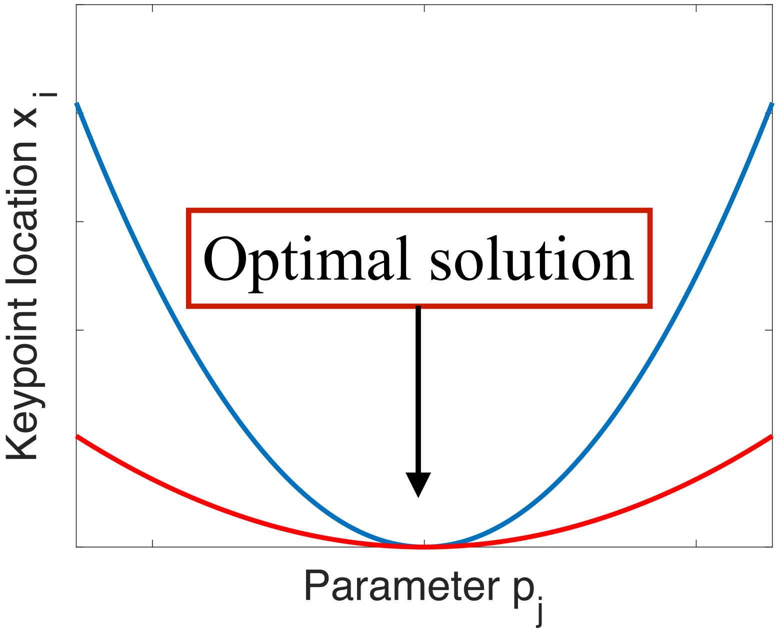
Iterative Alignment - Levenberg-Marquardt

$$E_{\mathrm{NLS}}(\Delta \boldsymbol{p}) \quad = \quad \Delta \boldsymbol{p}^T \boldsymbol{A} \Delta \boldsymbol{p} - 2\Delta \boldsymbol{p}^T \boldsymbol{b} + c, \qquad \boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad \boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{r}_i$$

- Solution 2. Levenberg-Marquardt (damped Gauss-Newton)

  $$(\boldsymbol{A} + \lambda \boldsymbol{I})\Delta \boldsymbol{p} = b$$
- Consider the gradient descent term:

$$\Delta \boldsymbol{p} = \frac{1}{\lambda}\sum_i \boldsymbol{J}^\top(\boldsymbol{x}_i)\boldsymbol{r}_i$$



- This will shift the parameters in the direction that reduces the residual $\boldsymbol{r}_i$.
- But the size of the shift depends on the magnitude of the gradient and the residual:
  - A larger residual $\boldsymbol{r}_i$ will result in a larger shift in the parameters $\boldsymbol{p}$.
  - A larger gradient $||\boldsymbol{J}||$ will result in a larger shift in the parameters $\boldsymbol{p}$.
- This is not necessarily what we want.

Iterative Alignment - Levenberg-Marquardt

$$E_{\mathrm{NLS}}(\Delta\boldsymbol{p}) \quad = \quad \Delta\boldsymbol{p}^T\boldsymbol{A}\Delta\boldsymbol{p} - 2\Delta\boldsymbol{p}^T\boldsymbol{b} + c, \qquad \boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad \boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{r}_i$$

- Gradient descent term:

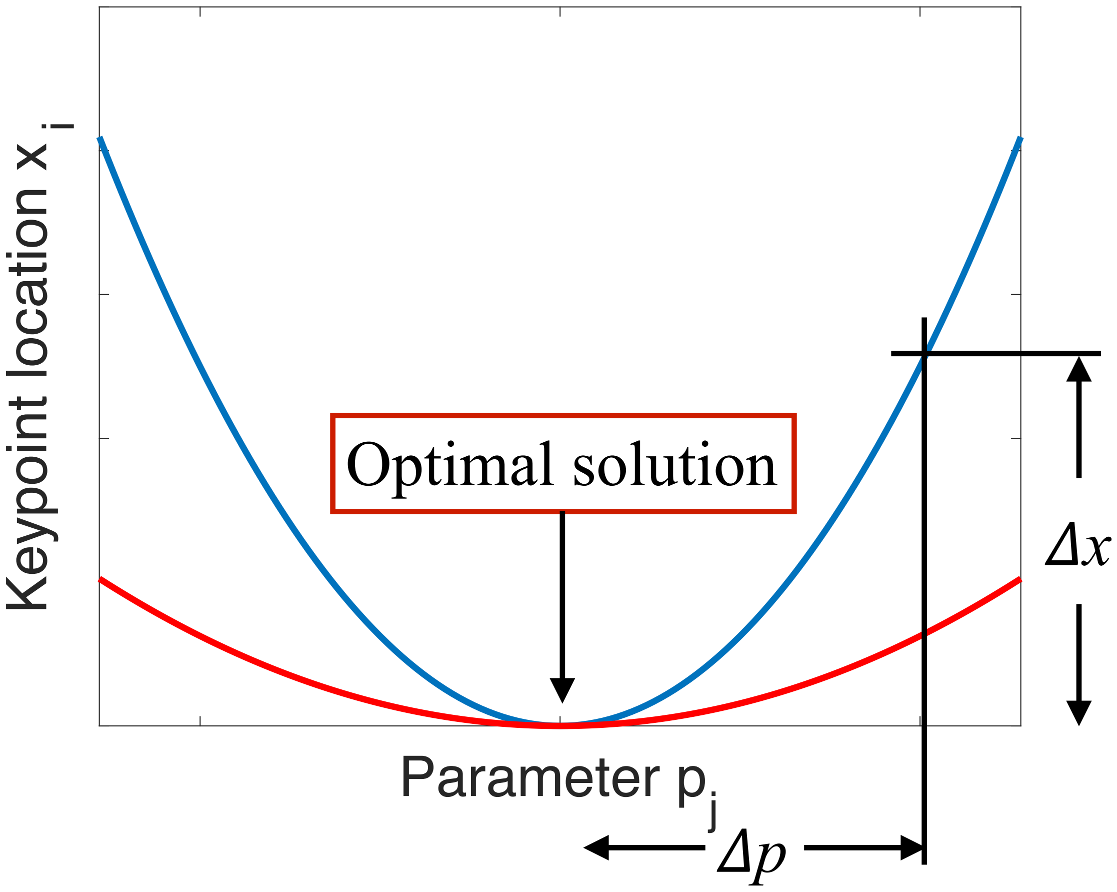
$$\Delta\boldsymbol{p} = \frac{1}{\lambda}\sum_i \boldsymbol{J}^\top(\boldsymbol{x}_i)\boldsymbol{r}_i$$

- Consider a simple transformation of the x coordinate with only one parameter p:

$$\Delta p = \frac{1}{\lambda}\sum_i \frac{dx_i}{dp}\Delta x$$

- We wish to select a value for λ that we predict will close the gap Δx:

$$\lambda \propto \sum_i \frac{dx_i \Delta x}{dp\Delta p} \cong \sum_i \left(\frac{dx_i}{dp}\right)^2$$



Iterative Alignment - Levenberg-Marquardt

$$E_{\mathrm{NLS}}(\Delta\boldsymbol{p}) = \Delta\boldsymbol{p}^T\boldsymbol{A}\Delta\boldsymbol{p} - 2\Delta\boldsymbol{p}^T\boldsymbol{b} + c, \qquad \boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad \boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{r}_i$$

$$\Delta p = \frac{1}{\lambda}\sum_i \frac{dx_i}{dp}\Delta x$$

$$\lambda \propto \sum_i \frac{dx_i \Delta x}{dp\Delta p} \cong \sum_i \left(\frac{dx_i}{dp}\right)^2$$

- Generalizing to multiple dimensions:

$$\lambda \propto \mathrm{diag}(\boldsymbol{A}) = \begin{bmatrix} \left(\frac{\partial x}{\partial p_1}\right)^2 + \left(\frac{\partial y}{\partial p_1}\right)^2 & 0 & \cdots & 0 \\ 0 & \left(\frac{\partial x}{\partial p_2}\right)^2 + \left(\frac{\partial y}{\partial p_2}\right)^2 & 0 & 0 \\ \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \cdots & \left(\frac{\partial x}{\partial p_n}\right)^2 + \left(\frac{\partial y}{\partial p_n}\right)^2 \end{bmatrix}$$

Iterative Alignment - Levenberg-Marquardt

$$E_{\mathrm{NLS}}(\Delta \boldsymbol{p}) \quad = \quad \Delta \boldsymbol{p}^T \boldsymbol{A} \Delta \boldsymbol{p} - 2\Delta \boldsymbol{p}^T \boldsymbol{b} + c, \qquad \boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad \boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{r}_i$$

- This reasoning led Marquardt to replace the identity matrix with diag(A):

$$(\boldsymbol{A} + \lambda \boldsymbol{I})\Delta \boldsymbol{p} = b \longrightarrow (\boldsymbol{A} + \lambda \mathrm{diag}(\boldsymbol{A}))\Delta \boldsymbol{p} = b$$

- The diag(A) term serves to scale the gradient descent step appropriately given the observed residual.

MATLAB:
```
options.Algorithm = 'levenberg-marquardt';
p = lsqnonlin(fun,p0,[],[],options);
```

End of Lecture
Nov 14, 2018

Example 1: Rigid 2D Transformation

Transformation	Parameters $\boldsymbol{p}$	Jacobian $\boldsymbol{J}(\boldsymbol{x};\theta)$
$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$

❖ Initial guess - use linear similarity transform

$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$

❖ and now set $\theta = \arctan \dfrac{b}{1+a}$

Example 2. Projective 2D Transformation

- Consider two images taken of the same planar scene, but from different vantages

- A 3x4 camera projection matrix relates the image points to the scene points for each of the images.

$$\tilde{\boldsymbol{x}}_s = \boldsymbol{K} \left[\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{t} \end{array} \right] \bar{\boldsymbol{p}}_w = \boldsymbol{P}\bar{\boldsymbol{p}}_w$$

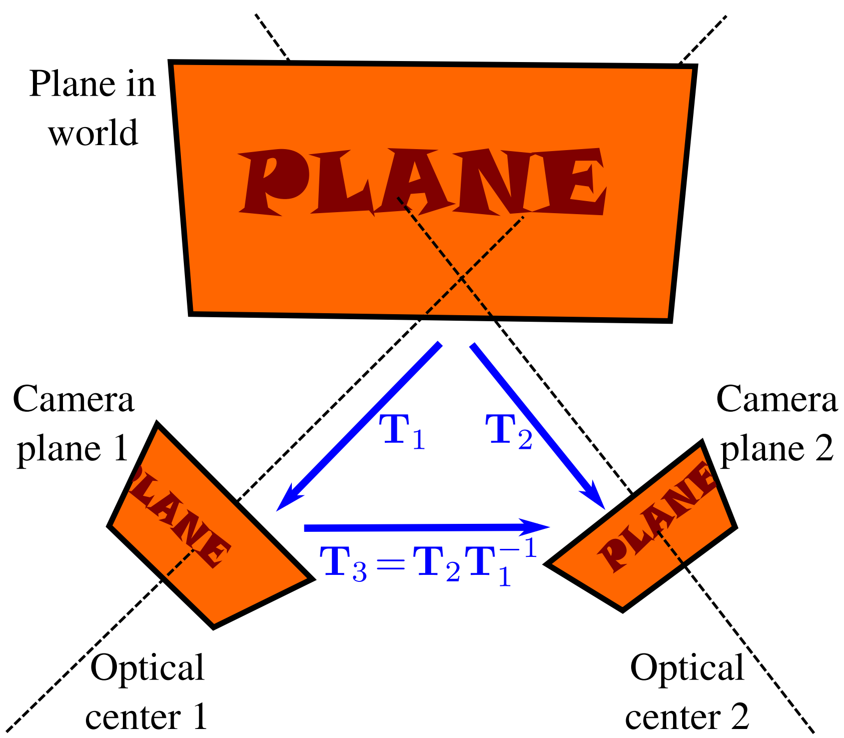
2D image projection

Intrinsic (calibration) matrix

Extrinsic (rotation + translation) matrix

Projection matrix

3D world point



Example 2. Projective 2D Transformation

$$\tilde{\boldsymbol{x}}_s = \boldsymbol{K}\left[\begin{array}{c|c}\boldsymbol{R} & \boldsymbol{t}\end{array}\right]\boldsymbol{p}_w = \boldsymbol{P}\boldsymbol{p}_w$$

- For convenience, we can align the 3D world coordinate frame with the scene plane, so that Z = 0 for all scene points.

- Under these conditions, projection to the image can be modelled by a 3 x 3 matrix $\tilde{\boldsymbol{H}}$ known as a homography:

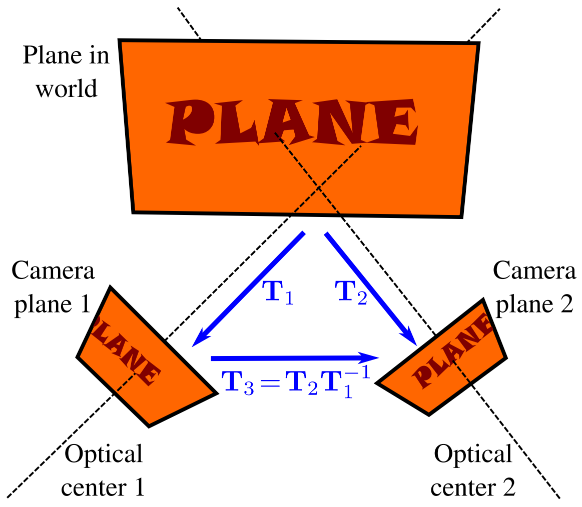
$$\tilde{\boldsymbol{x}} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \tilde{\boldsymbol{H}} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

- This means that the transformation of points between the two images is also a homography.

In particular, if

$\tilde{\boldsymbol{H}}_1$ and $\tilde{\boldsymbol{H}}_2$ model projection of the scene plane to Image 1 and 2, then

$\tilde{\boldsymbol{H}}_{21} = \tilde{\boldsymbol{H}}_2\tilde{\boldsymbol{H}}_1^{-1}$ models projection from Image 1 to Image 2.



Example 2. Projective 2D Transformation

$$\tilde{\boldsymbol{x}}' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \tilde{\boldsymbol{H}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Since this homography is a 3 x 3 matrix relating 2D image points in homogenous coordinates, it has 8 degrees of freedom.

$$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$$

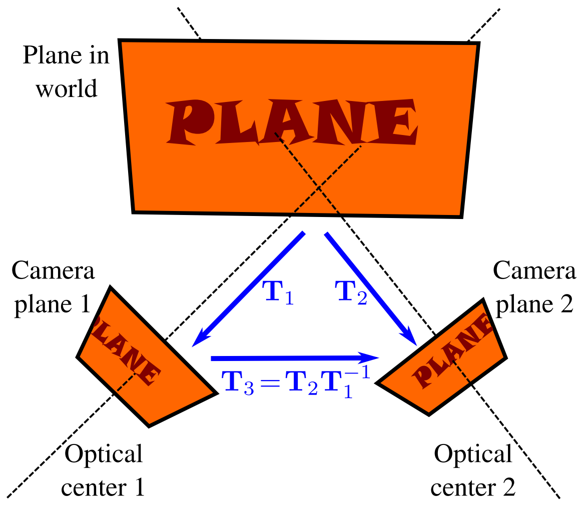
- While linear in projective space, this transformation is nonlinear in Euclidean space.

$$x' = \frac{(1+h_{00})x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1} \text{ and } y' = \frac{h_{10}x + (1+h_{11})y + h_{12}}{h_{20}x + h_{21}y + 1}.$$

- The Jacobian is

$$\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} = \frac{1}{D} \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{bmatrix}$$

where $D = h_{20}x + h_{21}y + 1$



Example 2. Projective 2D Transformation

$$x' = \frac{(1+h_{00})x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1} \quad \text{and} \quad y' = \frac{h_{10}x + (1+h_{11})y + h_{12}}{h_{20}x + h_{21}y + 1}.$$

- If we multiply through by the denominators we get a pair of equations that are linear in the parameters h_{ij}:

$$\begin{bmatrix} \hat{x}' - x \\ \hat{y}' - y \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -\hat{x}'x & -\hat{x}'y \\ 0 & 0 & 0 & x & y & 1 & -\hat{y}'x & -\hat{y}'y \end{bmatrix} \begin{bmatrix} h_{00} \\ \vdots \\ h_{21} \end{bmatrix} \quad \Leftrightarrow \boldsymbol{b} = \boldsymbol{A}\boldsymbol{h}$$

- If we have 4 pairs of matched points, a unique solution for the $\boldsymbol{h}$ is defined.
  - MATLAB \ will use LU solver
- If we have > 4 pairs of matched points, will generally be no exact solution, but we can find the $\boldsymbol{h}$ minimizing $\|\boldsymbol{A}\boldsymbol{h} - \boldsymbol{b}\|$ using the Moore-Penrose pseudo-inverse:

$$\boldsymbol{h} = \boldsymbol{A}^{\dagger}\boldsymbol{b} \quad \text{where } \boldsymbol{A}^{\dagger} \triangleq \left(\boldsymbol{A}^{\top}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{\top}$$

MATLAB function lsqminnorm

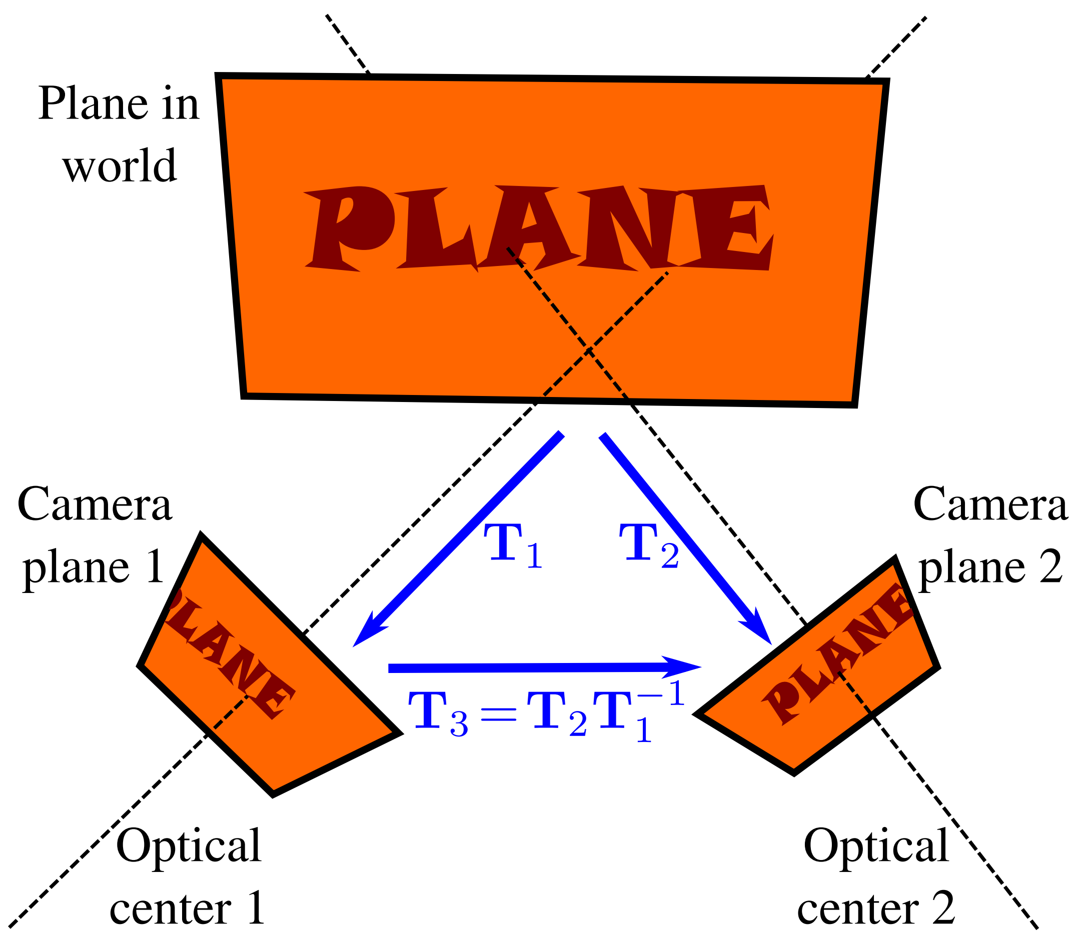
- This is called the Direct Linear Transform (DLT) method.
- This solution does not minimize the squared error $E_{\mathrm{NLS}}(\Delta \boldsymbol{p}) = \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p} + \Delta \boldsymbol{p}) - \boldsymbol{x}'_i\|^2$
- But it can be used to generate an initial guess at the parameters $\boldsymbol{p} = \{h_{ij}\}$.

Example 2. Projective 2D Transformation

- Given this initial guess, we can now solve for the homography iteratively as we did for the 2D rigid transformation, using Levenberg-Marquardt:

$$\left(\boldsymbol{A} + \lambda \text{diag}(\boldsymbol{A})\right)\Delta \boldsymbol{p} = b \quad \text{where} \quad \boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \quad \text{and} \quad \boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{r}_i$$

$$\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} = \frac{1}{D}\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{bmatrix} \quad \text{where } D = h_{20}x + h_{21}y + 1$$



Outline

- Linear Alignment Problems
- Non-Linear Alignment Problems