Global Flow

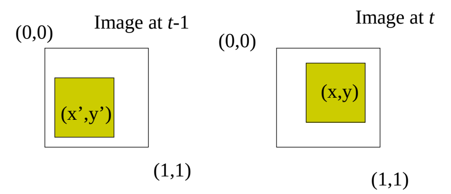
Anandan

Affine

Global Motion

- Estimate motion using all pixels in the image.
- Parametric flow gives an equation, which describes optical flow for each pixel.
  - Affine
  - Projective
- Global motion can be used to
  - generate mosaics
  - Object-based segmentation

Affine



$$u(x,y) = a_1 x + a_2 y + b_1$$
$$v(x,y) = a_3 x + a_4 y + b_2$$

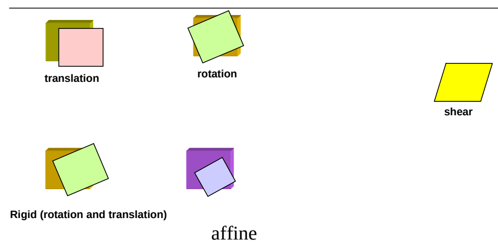
$$U = X - X'$$
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x' \\ y' \end{bmatrix}$$

Affine

$$u(x,y) = a_1 x + a_2 y + b_1$$
$$v(x,y) = a_3 x + a_4 y + b_2$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Spatial Transformations



Anandan

$$u(x,y) = a_1 x + a_2 y + b_1$$
$$v(x,y) = a_3 x + a_4 y + b_2$$

•Affine

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ a_3 \\ a_4 \\ b_2 \end{bmatrix}$$

$$\mathbf{u(x)} = \mathbf{X(x)a}$$

Anandan

$$\mathbf{u(x)} = \mathbf{X(x)a}$$

Optical flow constraint eq

$$f_x u + f_y v = -f_t$$

$$E(\mathbf{u}) = \sum_{\forall x \in f(x,y)} (f_t + f_{\mathbf{x}}^T \mathbf{u})^2 \qquad f_{\mathbf{x}} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

$$E(\mathbf{a}) = \sum_{\forall x \in f(x,y)} (f_t + f_{\mathbf{x}}^T \mathbf{X(x)a})^2$$

(a) Derive this Due

$$E(\delta a) = \sum_{\forall x \in f(x,y)} (f_t + f_{\mathbf{x}}^T \mathbf{X} \delta a\ )^2$$

Anandan

$$E(\delta a) = \sum_{\forall x \in f(x,y)} (f_t + f_{\mathbf{x}}^T \mathbf{X} \delta a\ )^2$$

min

$$\left[\sum X^T (f_X)(f_x)^T X\right] \delta a = -\sum X^T f_X f_t$$

(a) Derive this

$$Ax = b$$

Linear system

Basic Components

- Pyramid construction
- Motion estimation
- Image warping
- Coarse-to-fine refinement

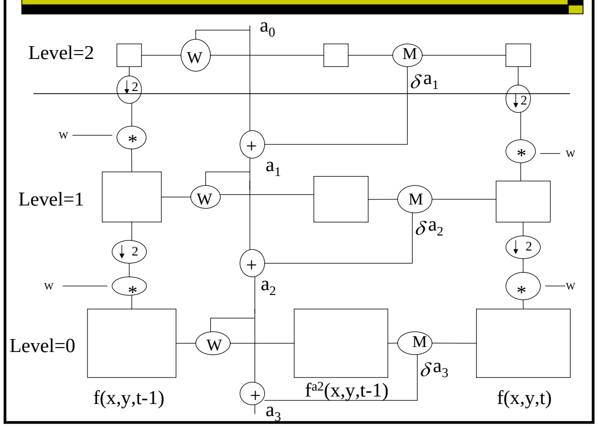


Image Warping

- Warping an image f into image h using some transformation g, involves mapping intensity at each pixel (x,y) in image f to a pixel $(g(x),g(y))$ image h such that

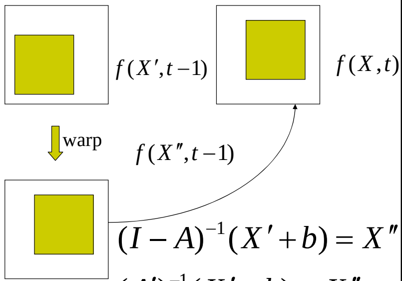
$$(x', y') = (g(x), g(y))$$

- In case of affine transformation, $\mathbf{x} = (x, y)$ is transformed to $\mathbf{x}' = (x', y')$ as:

$$\mathbf{x}' = \mathbf{Ax} + \mathbf{b}$$

Image Warping

$$X' = X - U = X - (AX + b)$$



$$(I - A)^{-1}(X' + b) = X''$$
$$(A')^{-1}(X' + b) = X''$$

Image Warping

$$X' = X - U = X - (AX + b)$$
$$X' = (I - A)X - b$$
$$X' = A'X - b$$
$$X' + b = A'X$$
$$(A')^{-1}(X' + b) = X$$

Image at time t: X
Image at time t-1: X'

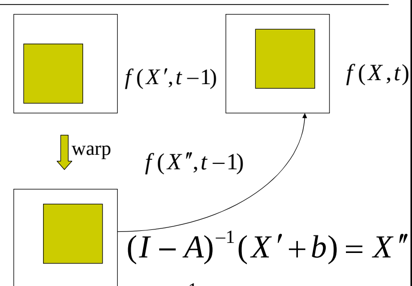
$$(A')^{-1}(X' + b) = X'' \qquad X' \rightarrow X''$$

Image Warping

- How about values in $X'' = (x'', y'')$ are not integer.
- But image is sampled only at integer rows and columns
  - Instead of converting X' to X'' and copying $f(X',t-1)$ at $f(X'',t-1)$ we can convert integer values X'' to X' and copy $f(X',t-1)$ at $f(X'',t-1)$

Image Warping

$$X' = X - U = X - (AX + b)$$



$$(I - A)^{-1}(X' + b) = X''$$
$$(A')^{-1}(X' + b) = X''$$

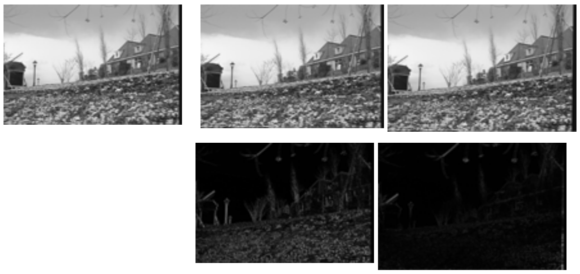
Image Warping

- But how about the values in X' are not integer.
- Perform bilinear interpolation to compute $f(X',t-1)$ at non-integer values.

Image Warping

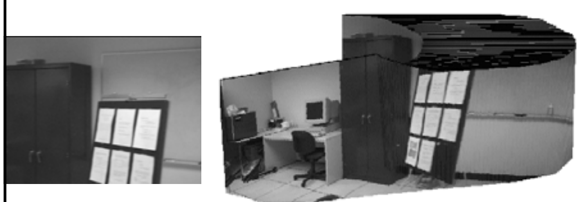
$$(A')^{-1}(X' + b) = X''$$
$$(X' + b) = (A')X''$$
$$X' = (A')X'' - b \qquad X'' \rightarrow X'$$

Warping



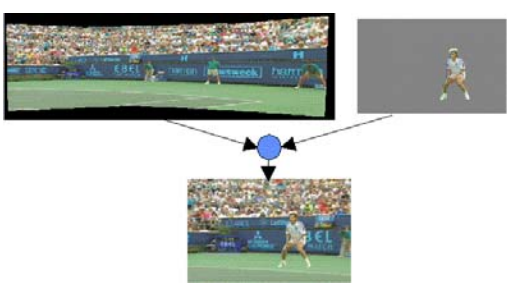
Video Mosaic

Video Mosaic



Video Mosaic

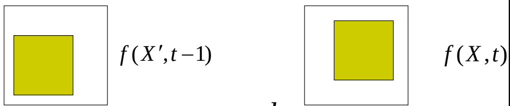
Sprite



Szeliski

Projective

Projective



$$x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}$$

$$y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}$$

Szeliski

$$x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1} \qquad \text{Projective}$$

$$y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}$$

$$E = \sum [f(x', y') - f(x, y)]^2 = \sum e^2$$

min

Szeliski

Motion Vector:

$$\mathbf{m} = [a_1 \quad a_2 \quad a_3 \quad a_4 \quad b_1 \quad b_2 \quad c_1 \quad c_2]^T$$

Szeliski (Levenberg-Marquadet)

$$\alpha_{kl} = \sum_n \frac{\partial e_n}{\partial m_k} \frac{\partial e_n}{\partial m_l} \qquad b_k = -\sum e \frac{\partial e_n}{\partial m_k}$$

gradient

$$\Delta m = (A + \lambda I)^{-1} b$$

Approximation of Hessian ($J^T J$, *Jacobian*)

$$\frac{\partial e}{\partial a_1} = \frac{\partial e}{\partial x'} \frac{\partial x'}{\partial a_1} + \frac{\partial e}{\partial y'} \frac{\partial y'}{\partial a_1}$$

$$E = \sum [f(x', y') - f(x, y)]^2 = \sum e^2$$

$$x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}$$

$$y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}$$

Approximation of Hessian

$$J^T = \begin{bmatrix} \frac{\partial e_1}{\partial m_1} & \frac{\partial e_2}{\partial m_1} & \cdots & \frac{\partial e_n}{\partial m_1} \\ \frac{\partial e_1}{\partial m_2} & \frac{\partial e_2}{\partial m_2} & \cdots & \frac{\partial e_n}{\partial m_2} \\ \frac{\partial e_1}{\partial m_3} & \frac{\partial e_2}{\partial m_3} & \cdots & \frac{\partial e_n}{\partial m_3} \\ \frac{\partial e_1}{\partial m_4} & \frac{\partial e_2}{\partial m_4} & \cdots & \frac{\partial e_n}{\partial m_4} \\ \frac{\partial e_1}{\partial m_5} & \frac{\partial e_2}{\partial m_5} & \cdots & \frac{\partial e_n}{\partial m_5} \\ \frac{\partial e_1}{\partial m_6} & \frac{\partial e_2}{\partial m_6} & \cdots & \frac{\partial e_n}{\partial m_6} \\ \frac{\partial e_1}{\partial m_7} & \frac{\partial e_2}{\partial m_7} & \cdots & \frac{\partial e_n}{\partial m_7} \\ \frac{\partial e_1}{\partial m_8} & \frac{\partial e_2}{\partial m_8} & \cdots & \frac{\partial e_n}{\partial m_8} \end{bmatrix} \quad J = \begin{bmatrix} \frac{\partial e_1}{\partial m_1} & \frac{\partial e_1}{\partial m_2} & \frac{\partial e_1}{\partial m_3} & \frac{\partial e_1}{\partial m_4} & \frac{\partial e_1}{\partial m_5} & \frac{\partial e_1}{\partial m_6} & \frac{\partial e_1}{\partial m_7} & \frac{\partial e_1}{\partial m_8} \\ \frac{\partial e_2}{\partial m_1} & \frac{\partial e_2}{\partial m_2} & \frac{\partial e_2}{\partial m_3} & \frac{\partial e_2}{\partial m_4} & \frac{\partial e_2}{\partial m_5} & \frac{\partial e_2}{\partial m_6} & \frac{\partial e_2}{\partial m_7} & \frac{\partial e_2}{\partial m_8} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_n}{\partial m_1} & \frac{\partial e_n}{\partial m_2} & \frac{\partial e_n}{\partial m_3} & \frac{\partial e_n}{\partial m_4} & \frac{\partial e_n}{\partial m_5} & \frac{\partial e_n}{\partial m_6} & \frac{\partial e_n}{\partial m_7} & \frac{\partial e_n}{\partial m_8} \end{bmatrix}$$

$$A = J^T J$$

$$\alpha_{kl} = \sum_n \frac{\partial e_n}{\partial m_k} \frac{\partial e_n}{\partial m_l} \qquad \textbf{A} \text{ Matrix}$$

Gradient Vector

$$b=\begin{bmatrix} -\sum_n e_n \frac{\partial e_n}{\partial a_1} \\ -\sum_n e_n \frac{\partial e_n}{\partial a_2} \\ -\sum_n e_n \frac{\partial e_n}{\partial a_3} \\ -\sum_n e_n \frac{\partial e_n}{\partial a_4} \\ -\sum_n e \frac{\partial e_n}{\partial b_1} \\ -\sum_n e_n \frac{\partial e_n}{\partial b_2} \\ -\sum_n e_n \frac{\partial e_n}{\partial c_1} \\ -\sum_n e_n \frac{\partial e_n}{\partial c_2} \end{bmatrix}$$

Partial Derivatives wrt motion parameters

$$\frac{\partial e}{\partial a_1}=\frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_1}+\frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_1}$$

$$x'=\frac{a_1x+a_2y+b_1}{c_1x+c_2y+1},\quad y'=\frac{a_3x+a_4y+b_2}{c_1x+c_2y+1}$$

$$\frac{\partial x'}{\partial a_1}=\frac{x}{c_1x+c_2y+1} \qquad \frac{\partial y'}{\partial a_1}=0$$

$$\frac{\partial x'}{\partial a_2}=\frac{y}{c_1x+c_2y+1} \qquad \frac{\partial y'}{\partial a_2}=0$$

$$\frac{\partial x'}{\partial a_3}=0 \qquad \frac{\partial y'}{\partial a_3}=\frac{x}{c_1x+c_2y+1}$$

$$\frac{\partial x'}{\partial a_4}=0 \qquad \frac{\partial y'}{\partial a_4}=\frac{y}{c_1x+c_2y+1}$$

$$\frac{\partial x'}{\partial b_1}=\frac{1}{c_1x+c_2y+1} \qquad \frac{\partial y'}{\partial b_1}=0$$

$$\frac{\partial x'}{\partial b_2}=0 \qquad \frac{\partial y'}{\partial b_2}=\frac{1}{c_1x+c_2y+1}$$

$$\frac{\partial x'}{\partial c_1}=\frac{-x(a_1x+a_2y+b_1)}{(c_1x+c_2y+1)^2} \qquad \frac{\partial y'}{\partial c_1}=\frac{-x(a_3x+a_4y+b_2)}{(c_1x+c_2y+1)^2}$$

$$\frac{\partial x'}{\partial c_2}=\frac{-y(a_1x+a_2y+b_1)}{(c_1x+c_2y+1)^2} \qquad \frac{\partial y'}{\partial c_2}=\frac{-y(a_3x+a_4y+b_2)}{(c_1x+c_2y+1)^2}$$

Partial derivatives wrt image coordinates

$$E=\sum[f(x',y')-f(x,y)]^2=\sum e^2$$

$$\frac{\partial e}{\partial x'}=f_{x'}$$

$$\frac{\partial e}{\partial y'}=f_{y'}$$

Partial derivatives

$$\frac{\partial e}{\partial a_1}=\frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_1}+\frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_1}=f_{x'}\frac{x}{c_1x+c_2y+1}$$

$$\frac{\partial e}{\partial a_2}=\frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_2}+\frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_2}=f_{x'}\frac{y}{c_1x+c_2y+1}$$

$$\frac{\partial e}{\partial a_3}=\frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_3}+\frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_3}=f_{y'}\frac{x}{c_1x+c_2y+1}$$

$$\frac{\partial e}{\partial a_4}=\frac{\partial e}{\partial x'}\frac{\partial x'}{\partial a_4}+\frac{\partial e}{\partial y'}\frac{\partial y'}{\partial a_4}=f_{y'}\frac{y}{c_1x+c_2y+1}$$

$$\frac{\partial e}{\partial b_1}=\frac{\partial e}{\partial x'}\frac{\partial x'}{\partial b_1}+\frac{\partial e}{\partial y'}\frac{\partial y'}{\partial b_1}=f_{x'}\frac{1}{c_1x+c_2y+1}$$

$$\frac{\partial e}{\partial b_2}=\frac{\partial e}{\partial x'}\frac{\partial x'}{\partial b_2}+\frac{\partial e}{\partial y'}\frac{\partial y'}{\partial b_2}=f_{y'}\frac{1}{c_1x+c_2y+1}$$

$$\frac{\partial e}{\partial c_1}=\frac{\partial e}{\partial x'}\frac{\partial x'}{\partial c_1}+\frac{\partial e}{\partial y'}\frac{\partial y'}{\partial c_1}=f_{x'}\frac{-x(a_1x+a_2y+b_1)}{(c_1x+c_2y+1)^2}+f_{y'}\frac{-x(a_3x+a_4y+b_2)}{(c_1x+c_2y+1)^2}$$

$$\frac{\partial e}{\partial c_2}=\frac{\partial e}{\partial x'}\frac{\partial x'}{\partial c_2}+\frac{\partial e}{\partial y'}\frac{\partial y'}{\partial c_2}=f_{x'}\frac{-y(a_1x+a_2y+b_1)}{(c_1x+c_2y+1)^2}+f_{y'}\frac{-y(a_3x+a_4y+b_2)}{(c_1x+c_2y+1)^2}$$

Gradient Vector

$$\mathbf{b}=\begin{bmatrix} -\sum e f_{x'}\frac{x}{c_1x+c_2y+1} \\ -\sum e f_{x'}\frac{y}{c_1x+c_2y+1} \\ -\sum e f_{y'}\frac{x}{c_1x+c_2y+1} \\ -\sum e f_{y'}\frac{y}{c_1x+c_2y+1} \\ -\sum e f_{x'}\frac{1}{c_1x+c_2y+1} \\ -\sum e f_{y'}\frac{1}{c_1x+c_2y+1} \\ \sum ex\left[\frac{f_{x'}(a_1x+a_2y+b_1)+f_{y'}(a_3x+a_4y+b_2)}{(c_1x+c_2y+1)^2}\right] \\ \sum ey\left[\frac{f_{x'}(a_1x+a_2y+b_1)+f_{y'}(a_3x+a_4y+b_2)}{(c_1x+c_2y+1)^2}\right] \end{bmatrix}$$

Szeliski (Levenberg-Marquadet)

- Start with some initial value of m, and λ=.001
  - For each pixel I at (x_i, y_i)
    - Compute (x', y') using projective transform.
    - Compute $e = f(x', y') - f(x, y)$
    - Compute $\frac{\partial e}{\partial m_k}=\frac{\partial e}{\partial x'}\frac{\partial x'}{\partial m_k}+\frac{\partial e}{\partial y'}\frac{\partial y'}{\partial m_k}$

Szeliski (Levenberg-Marquadet)

-Compute A **and** b

-Solve system

$$(A - \lambda I)\Delta m = b$$

-Update

$$m^{t+1} = m^t + \Delta m$$

Szeliski (Levenberg-Marquadet)

- check if error has decreased, if not increase λ by a factor of 10 and compute a new Δm
- If error has decreased, decrease λ by a factor of 10 and compute a new Δm
- Continue iteration until error is below threshold.