Lecture-17

Computing Optical Flow: Lucas & Kanade

Global Flow

Lucas & Kanade (Least Squares)

- Optical flow eq

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

- Consider 3 by 3 window

$$f_{x1} u + f_{y1} v = -f_{t1}$$
$$\vdots$$
$$f_{x9} u + f_{y9} v = -f_{t9}$$

$$\begin{bmatrix} f_{x1} & f_{y1} \\ \vdots & \vdots \\ f_{x9} & f_{y9} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_{t1} \\ \vdots \\ -f_{t9} \end{bmatrix}$$

$$\mathbf{Au} = \mathbf{f_t}$$

Lucas & Kanade

$$\mathbf{Au} = \mathbf{f_t}$$

$$\mathbf{A^T A u} = \mathbf{A^T f_t}$$

$$\mathbf{u} = (\mathbf{A^T A})^{-1} \mathbf{A^T f_t}$$

$$\Uparrow$$

$$\min \sum (f_{xi} u + f_{yi} v + f_t)^2$$

Lucas & Kanade

$$\min \sum (f_{xi} u + f_{yi} v + f_t)^2$$

$$\Downarrow$$

$$\sum (f_{xi} u + f_{yi} v + f_{ti}) f_{xi} = 0$$

$$\sum (f_{xi} u + f_{yi} v + f_{ti}) f_{yi} = 0$$

Lucas & Kanade

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum f_{xi}^2 & \sum f_{xi} f_{yi} \\ \sum f_{xi} f_{yi} & \sum f_{yi}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum f_{xi} f_{ti} \\ -\sum f_{yi} f_{ti} \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\sum f_{xi}^2 \sum f_{yi}^2 - \sum f_{xi} f_{yi} \sum f_{yi}^2} \begin{bmatrix} \sum f_{yi}^2 & -\sum f_{xi} f_{yi} \\ -\sum f_{xi} f_{yi} & \sum f_{xi}^2 \end{bmatrix} \begin{bmatrix} -\sum f_{xi} f_{ti} \\ -\sum f_{yi} f_{ti} \end{bmatrix}$$

$$u = \frac{-\sum f_{yi}^2 \sum f_{xi} f_{ti} + \sum f_{xi} f_{yi} \sum f_{yi} f_{ti}}{\sum f_{xi}^2 \sum f_{yi}^2 - \sum f_{xi} f_{yi} \sum f_{yi}^2}$$

$$v = \frac{\sum f_{xi} f_{ti} \sum f_{xi} f_{ti} - \sum f_{xi}^2 \sum f_{yi} f_{ti}}{\sum f_{xi}^2 \sum f_{yi}^2 - \sum f_{xi} f_{yi} \sum f_{yi}^2}$$

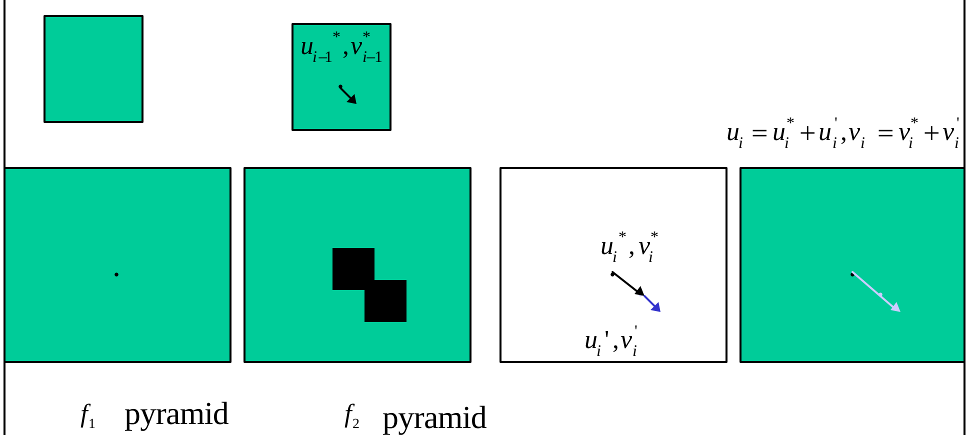
Comments

- Horn-Schunck and Lucas-Kanade optical method works only for small motion.
- If object moves faster, the brightness changes rapidly, 2x2 or 3x3 masks fail to estimate spatiotemporal derivatives.
- Pyramids can be used to compute large optical flow vectors.

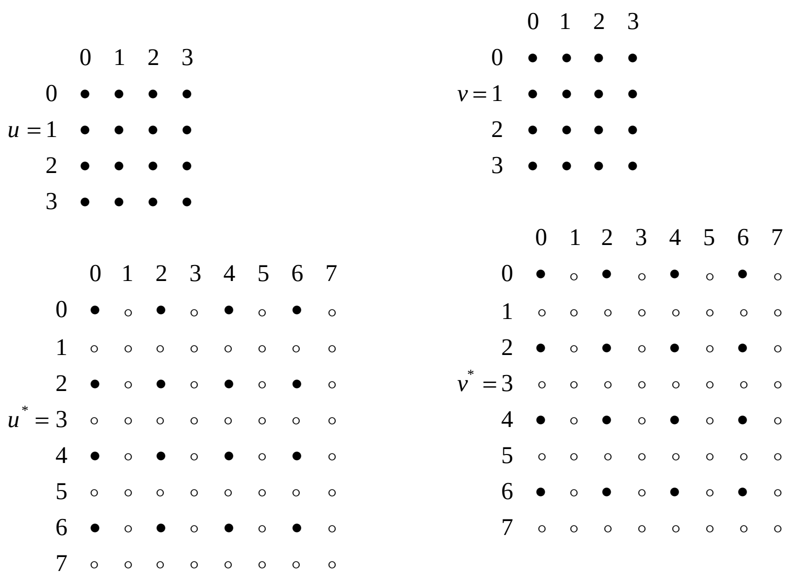
Lucas Kanade with Pyramids

- Compute 'simple' LK at highest level
- At level i
  - Take flow u_{i-1}, v_{i-1} from level i-1
  - bilinear interpolate it to create u_i^*, v_i^* matrices of twice resolution
  - multiply u_i^*, v_i^* by 2
  - compute f_t from a block displaced by $u_i^*(x,y)$, $v_i^*(x,y)$
  - Apply LK to get $u_i'(x, y)$, $v_i'(x, y)$ (the correction in flow)
  - Add corrections u_i' v_i', *i.e.* $u_i = u_i^* + u_i'$, $v_i = v_i^* + v_i'$.

Pyramids

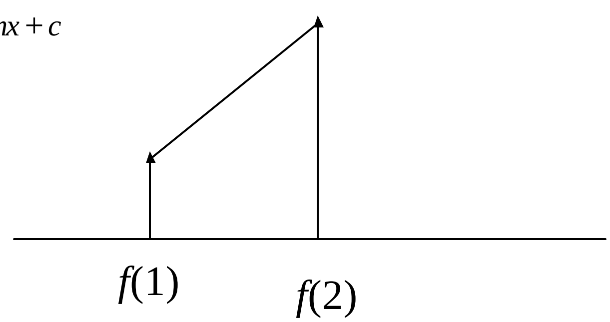


Interpolation



1-D Interpolation

$$y = mx + c$$
$$f(x) = mx + c$$



2-D Interpolation

$f(x,y) = a_1 + a_2 x + a_3 y + a_4 xy$ Bilinear

X X
O X

Bi-linear Interpolation

Four nearest points of (x,y) are:

$$(\underline{x}, \underline{y}), (\overline{x}, \underline{y}), (\underline{x}, \overline{y}), (\overline{x}, \overline{y})$$

$$(3,5), (4,5), (3,6), (4,6)$$

$\underline{x} = \text{int}(x)$ 3 (3.2,5.6)

$\underline{y} = \text{int}(y)$ 5

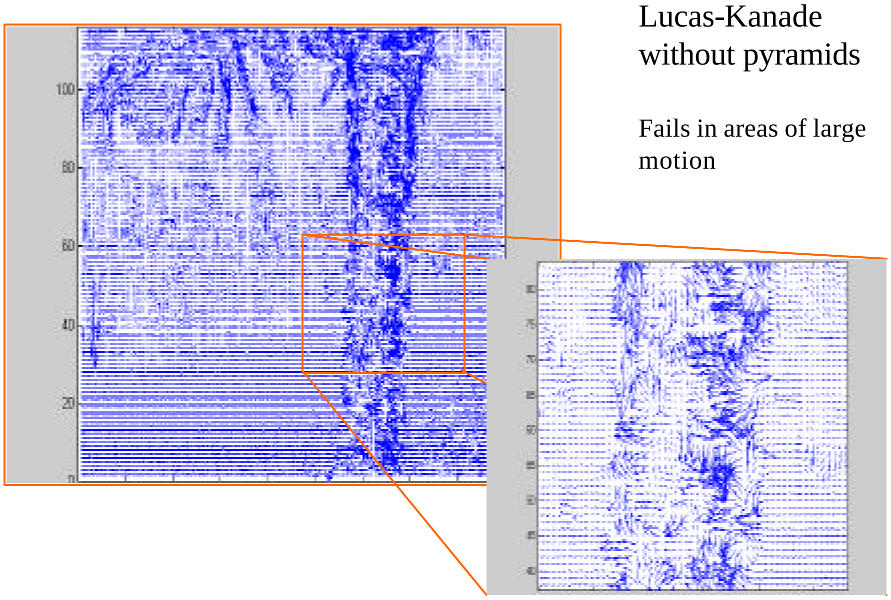
$\overline{x} = \underline{x} + 1$ 4

$\overline{y} = \underline{y} + 1$ 6

X (3,6) X (4,6)
X(3,5) X (4,5)

$$f'(x,y) = \overline{e}_x \overline{e}_y f(\underline{x},\underline{y}) + \underline{e}_x \overline{e}_y f(\overline{x},\underline{y}) +$$
$$\overline{e}_x \underline{e}_y f(\underline{x},\overline{y}) + \underline{e}_x \underline{e}_y f(\overline{x},\overline{y})$$

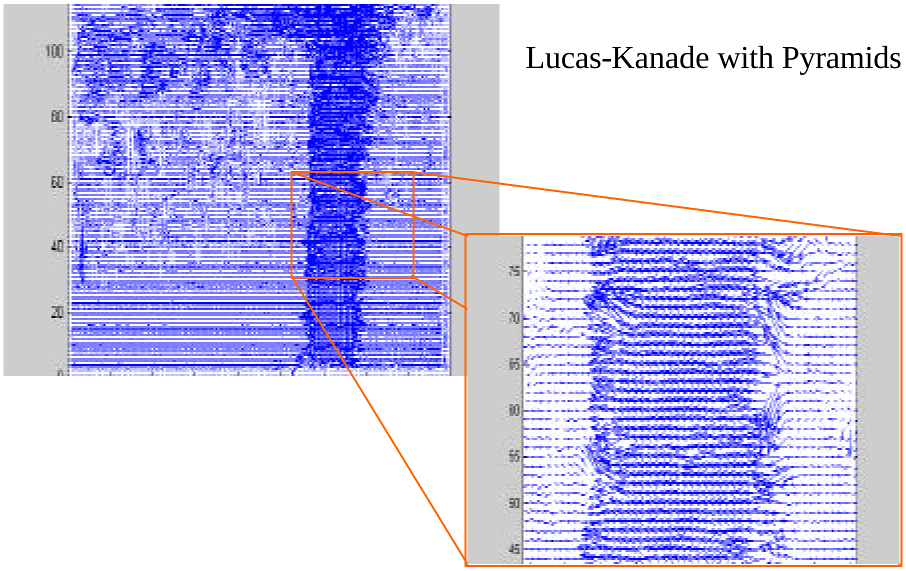
$$\overline{e}_x = \overline{x} - x \qquad \overline{e}_x = \overline{x} - x = 4 - 3.2 = .8$$
$$\overline{e}_y = \overline{y} - y \qquad \overline{e}_y = \overline{y} - y = 6 - 5.6 = .4$$
$$\underline{e}_x = x - \underline{x} \qquad \underline{e}_x = x - \underline{x} = 3.2 - 2 = .2$$
$$\underline{e}_y = y - \underline{y} \qquad \underline{e}_y = y - \underline{y} = 5.6 - 5 = .6$$

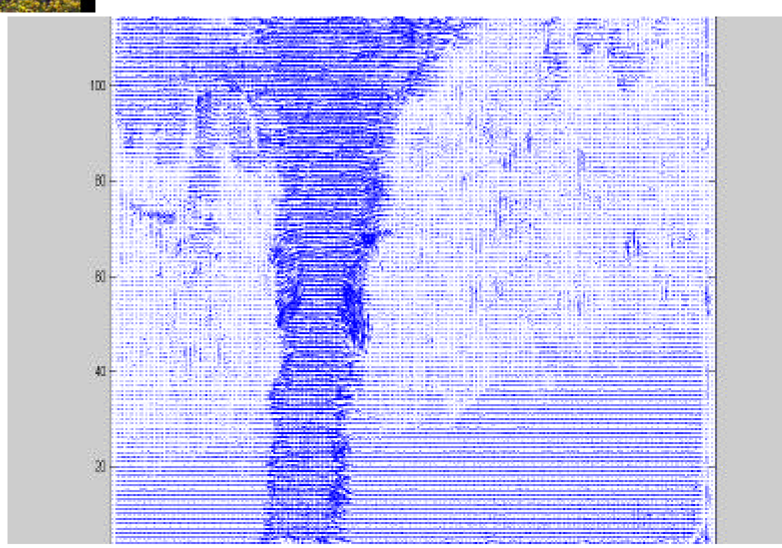


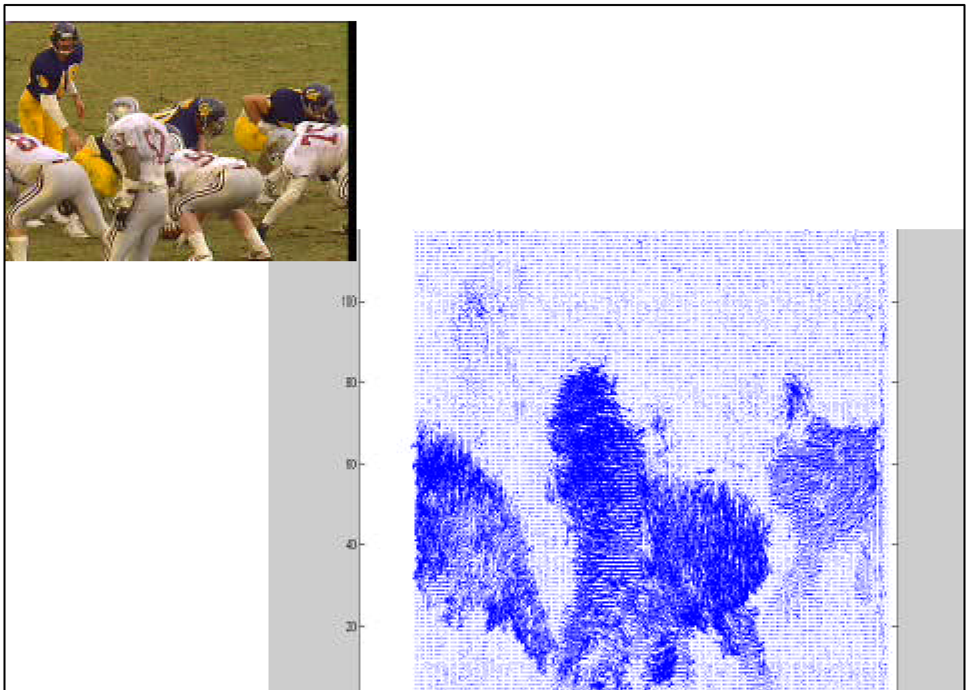
Lucas-Kanade without pyramids

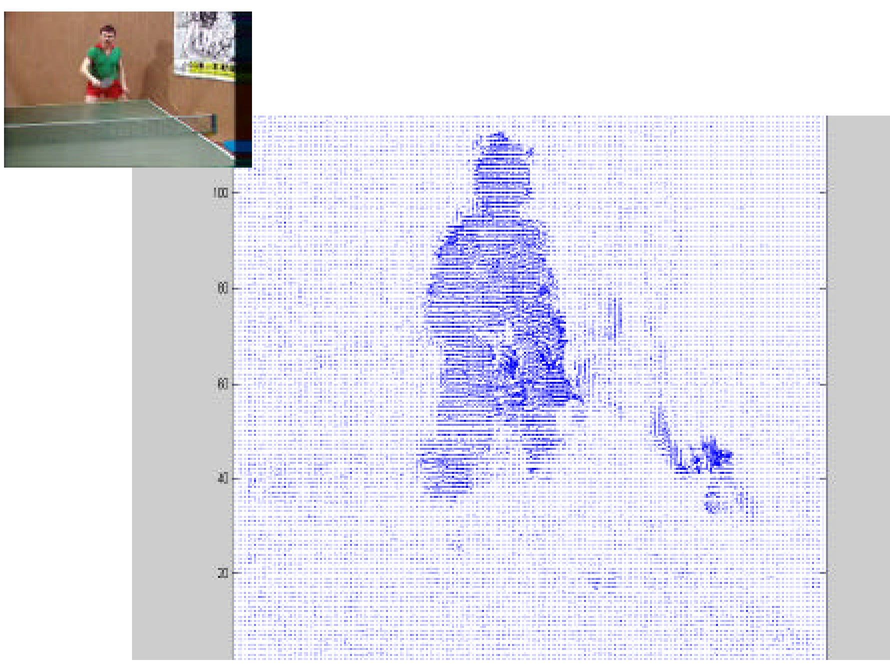
Fails in areas of large motion

Lucas-Kanade with Pyramids






100
80
60
40
20


100
80
60
40
20

Global Flow

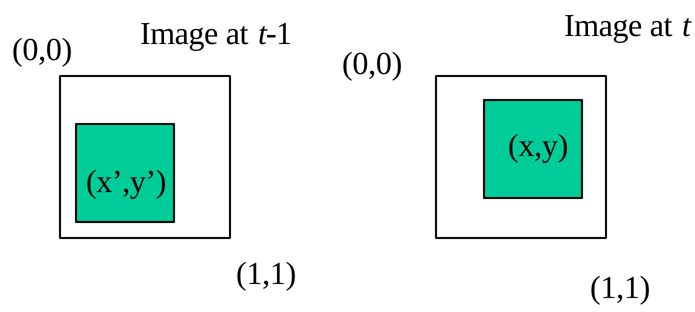
Anandan

Affine

Global Motion

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

Affine



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

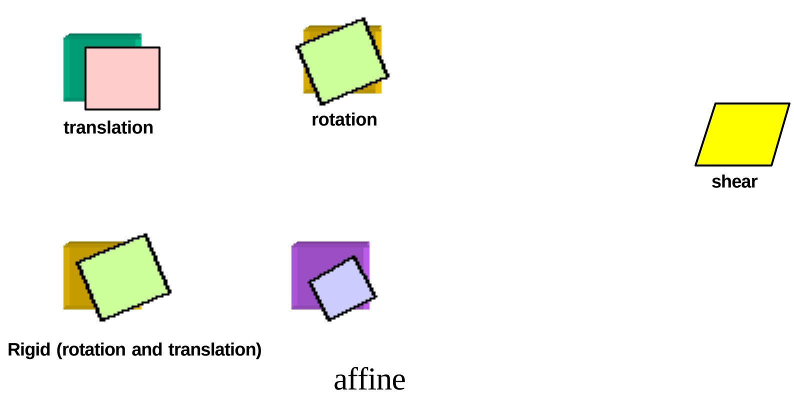
$$X' = X - U$$

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

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

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

Anandan

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

Optical flow constraint eq $f_x u + f_y v = -f_t$

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

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

min

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

$$Ax = b$$

Linear system

Basic Components

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

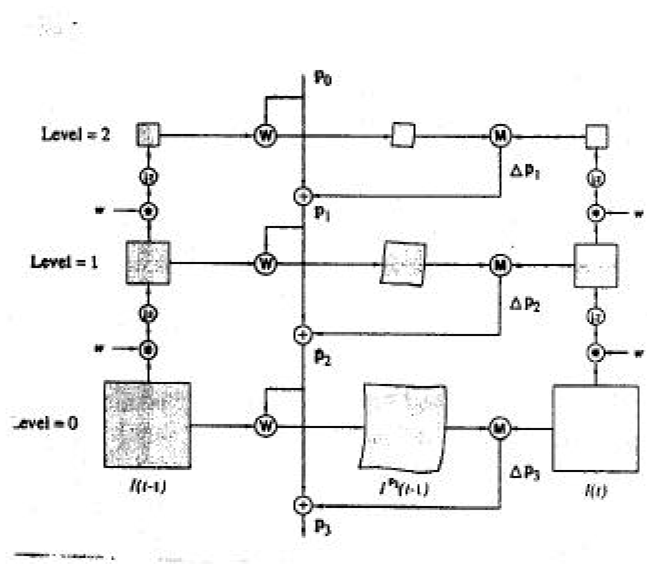


Image Warping

- 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

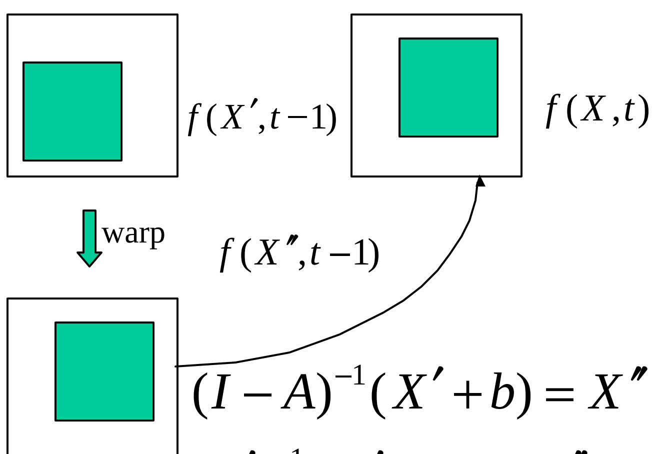
$$(x', y') = (g(x), g(y))$$

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

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

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'

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

$$X' \rightarrow X''$$

Image Warping

- How about values in are not integer.
- But image is sampled only at integer rows and columns
  - Instead of converting to and copying at we can convert integer values to and copy at

Image Warping

• But how about the values in are not integer.

• Perform bilinear interpolation to compute at non-integer values.

Image Warping

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

$$(X' + b) = (A')X''$$

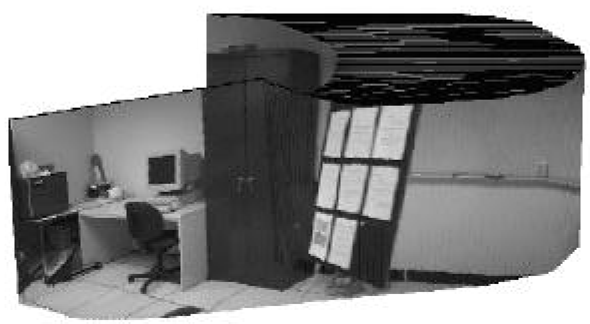
$$X' = (A')X'' - b \qquad X'' \rightarrow X'$$

Warping

Show Demos

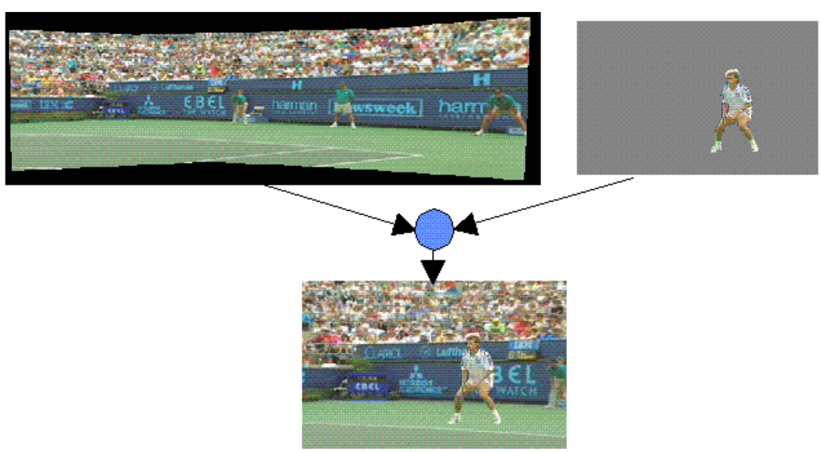
Video Mosaic

Video Mosaic

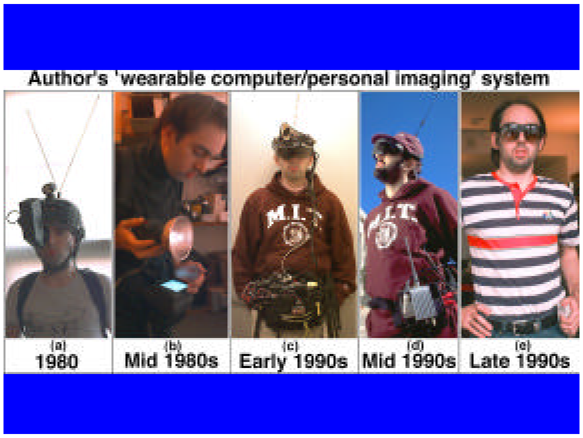


Video Mosaic

Sprite

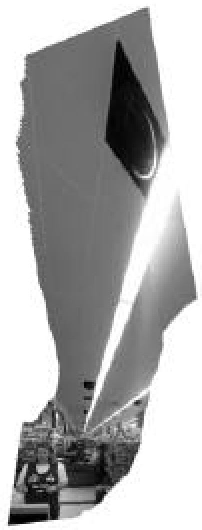


Steve Mann



Building

Wal-Mart

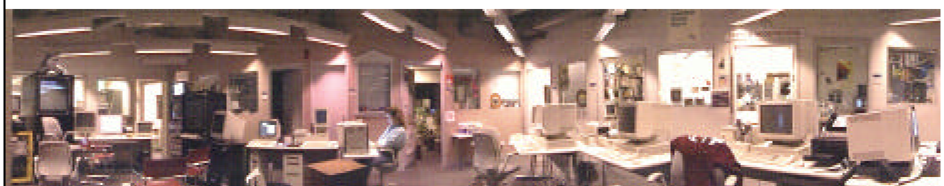


Scientific American Frontiers

Scientific American Frontiers

Head-mounted Camera at Restaurant

MIT Media Lab



Webpages

- http://n1nlf1.eecg.toronto.edu/tip.ps.gz
  Video Orbits of the projective group, S. Mann and R. Picard.
- http://wearcam.org/pencigraphy (C code for generating mosaics)