Global Flow

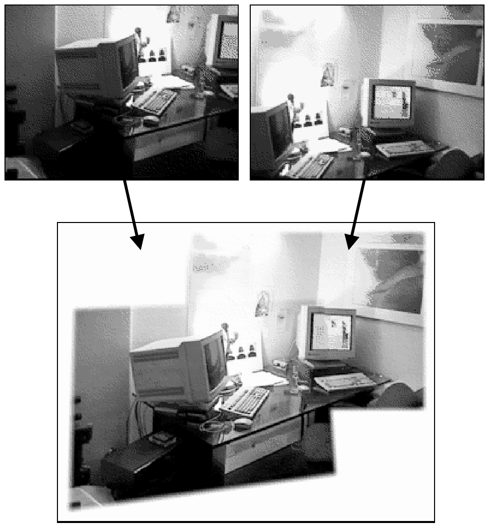
Global Flow

- Dominant Motion in the scene
  - Motion of all points in the scene
  - Motion of most of the points in the scene
  - A Component of motion of all points in the scene

- Global Motion is caused by
  - Motion of sensor (Ego Motion)
  - Motion of a rigid scene

- Estimation of Global Motion can be used to
  - Video Mosaics
  - Image Alignment (Registration)
  - Removing Camera Jitter
  - Tracking (By neglecting camera motion)
  - Video Segmentation etc.

Global Flow



Application: Image Alignment

Estimation of Global Flow

Assume Affine Flow:
$$\begin{bmatrix} u \\ v \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} + B$$

$$u = v_x = a_1 x + a_2 y + b_1$$
$$v = v_y = a_3 x + a_4 y + b_2$$

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

$$\mathbf{U} = \mathbf{Xa}$$

James R. Bergen, P. Anandan, Keith J. Hanna, Rajesh Hingorani: “Hierarchical Model-Based Motion Estimation," ECCV 1992: 237-252

Estimation of Global Flow

$$(\nabla I)^T \begin{bmatrix} u \\ v \end{bmatrix} + I_t = 0$$ (Optical Flow Constraint Equation)

$$(\nabla I)^T \mathbf{X}\mathbf{a} + I_t = 0$$

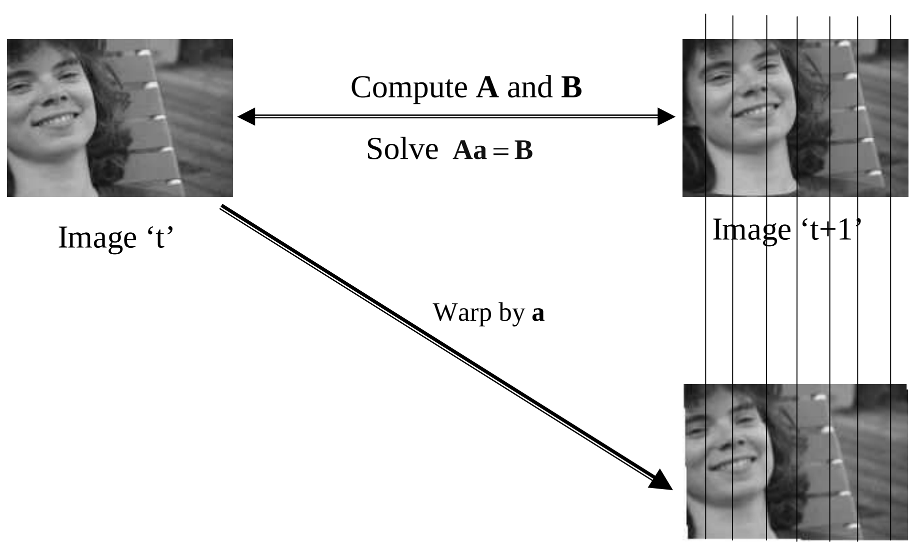
$$\sum_{\forall (x,y) \in I} \left[(\nabla I)^T \mathbf{X}\mathbf{a} + I_t \right]^2 = 0$$

$$\min \sum_{\forall (x,y) \in I} \left[(\nabla I)^T \mathbf{X}\mathbf{a} + I_t \right]^2 \Longrightarrow \left[\sum_{\forall (x,y) \in I} \mathbf{X}^T (\nabla I)(\nabla I)^T \mathbf{X} \right] \mathbf{a} = - \sum_{\forall (x,y) \in I} I_t \mathbf{X}^T \nabla I$$

$$\mathbf{A}\mathbf{a} = \mathbf{B}$$

Estimation of Global Flow

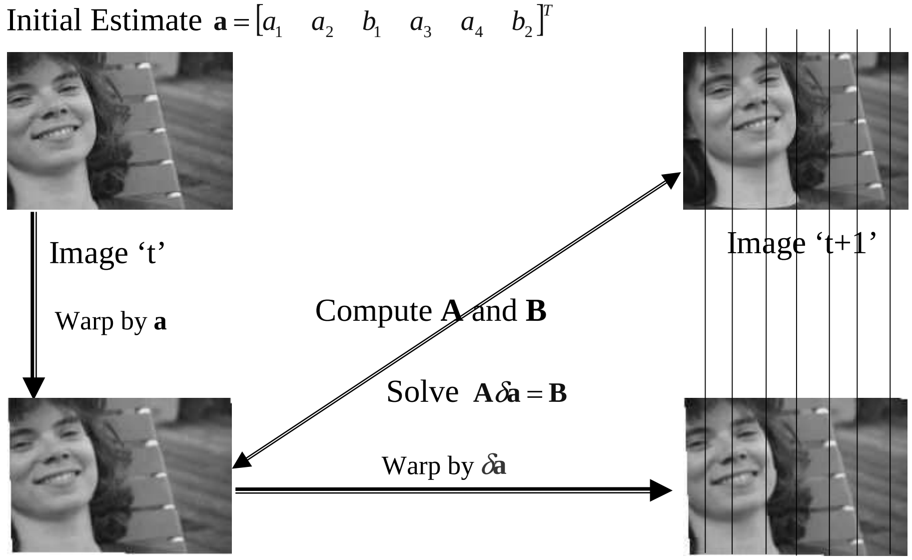
Single Iteration



Estimation of Global Flow

Iterative

Initial Estimate $\mathbf{a} = [a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$



Estimation of Global Flow

Parameters Update

Initial Estimate $\mathbf{a} = [a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$

Computed Parameters $\delta\mathbf{a} = [\delta a_1 \quad \delta a_2 \quad \delta b_1 \quad \delta a_3 \quad \delta a_4 \quad \delta b_2]^T$

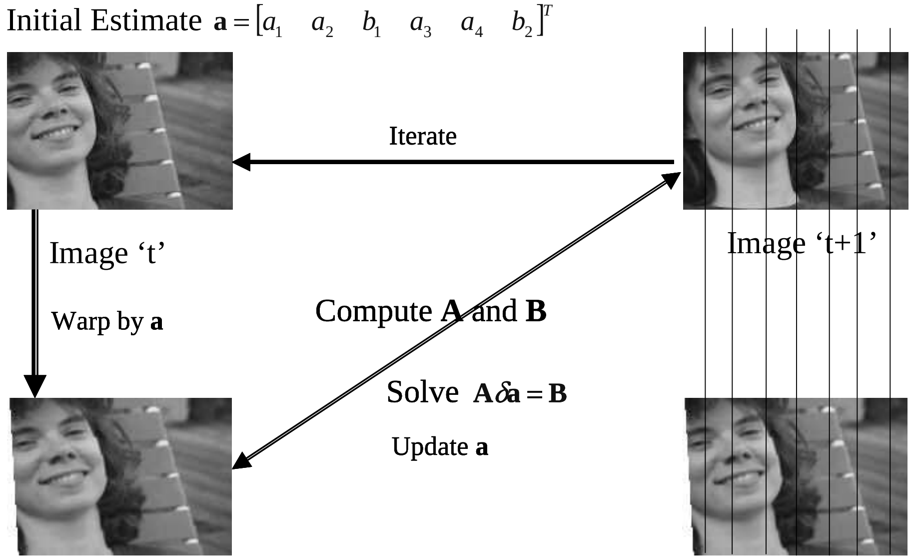
Update Equations

$$a_1 = a_1\delta a_1 + a_3\delta a_2 + a_1 + \delta a_1$$
$$a_2 = a_2\delta a_1 + a_4\delta a_2 + a_2 + \delta a_2$$
$$b_1 = b_1\delta a_1 + b_2\delta a_2 + b_1 + \delta b_1$$
$$a_3 = a_1\delta a_3 + a_3\delta a_4 + a_3 + \delta a_3$$
$$a_4 = a_2\delta a_3 + a_4\delta a_4 + a_4 + \delta a_4$$
$$b_2 = b_1\delta a_3 + b_2\delta a_4 + b_2 + \delta b_2$$

Estimation of Global Flow

Iterative

Initial Estimate $\mathbf{a} = [a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$

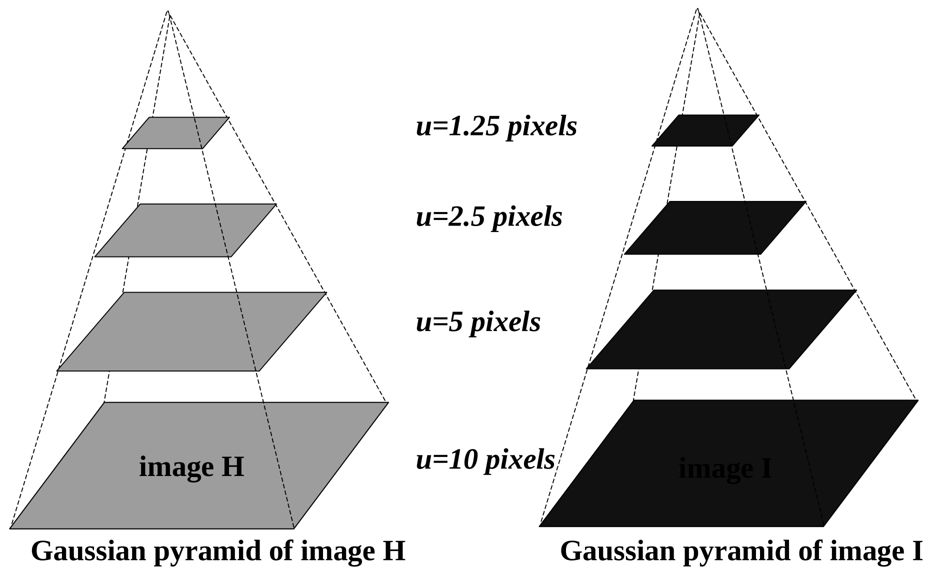


Iterative Refinement

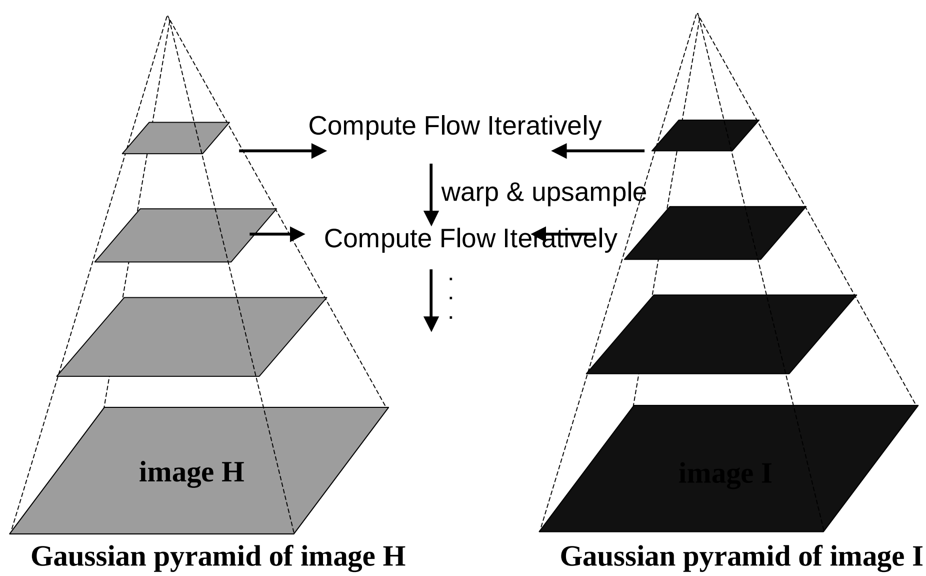
Iterative Algorithm

1. Given Im1, Im2 and Initial Estimate a.
2. Warp Im1 towards Im2 using the Estimate a and let I1 be the warped image
   - *Use image warping techniques (to be covered later)*
3. Compute A and B by using I1 and Im2
4. Estimate global flow by solving linear system Aδa=B
5. Update Parameters and let the updated parameters be a.
6. Repeat until convergence or a fixed number of iterations

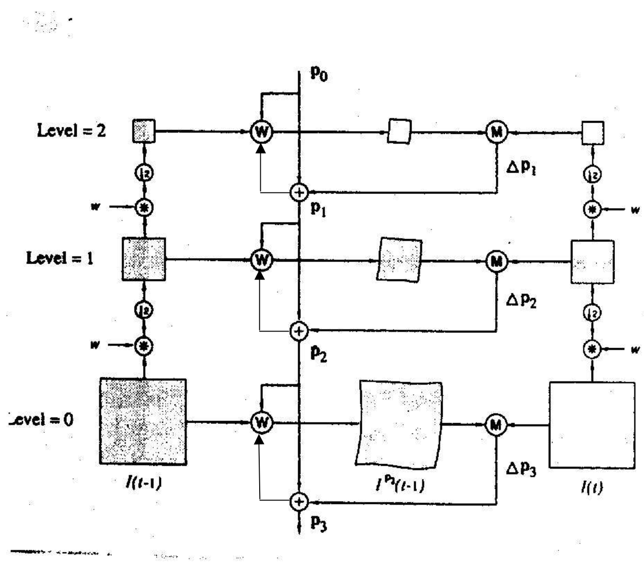
Coarse-to-fine global flow estimation



Coarse-to-fine global flow estimation



Coarse-to-fine global motion estimation



Basic Components

- Pyramid Construction
- Motion Estimation
- Image Warping
- Coarse to Fine Refinement

Result of Global Motion Estimation

Image
'‘t’

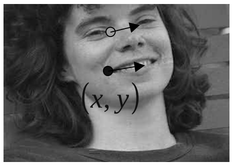
Image ‘t+1’

output

4 Pyramids Level
5 Iterations/Pyramid Level

Video Mosaic

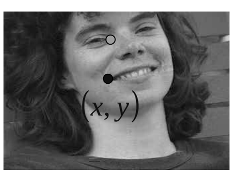
Image Warping

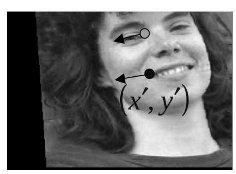


$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x' - x \\ y' - y \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 + a_1 & a_2 \\ a_3 & 1 + a_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Image Warping





$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 + a_1 & a_2 \\ a_3 & 1 + a_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

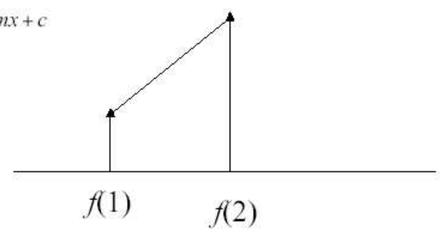
$$\Rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 + a_1 & a_2 \\ a_3 & 1 + a_4 \end{bmatrix}^{-1} \left(\begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right)$$

Interpolation

1-D Interpolation

$y = mx + c$

$f(x) = mx + c$



Interpolation

2-D Interpolation

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

X X

O X

Interpolation

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	

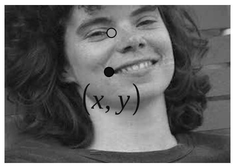
X (3,6) X (4,6)
X(3,5) X (4,5)

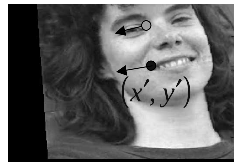
Interpolation

$$f'(x,y) = \overline{\varepsilon_x \varepsilon_y} f(\underline{x},\underline{y}) + \underline{\varepsilon_x}\,\overline{\varepsilon_y} f(\overline{x},\underline{y}) +$$
$$\overline{\varepsilon_x}\,\underline{\varepsilon_y} f(\underline{x},\overline{y}) + \underline{\varepsilon_x}\,\underline{\varepsilon_y} f(\overline{x},\overline{y})$$

$\overline{\varepsilon_x} = \overline{x} - x$	$\overline{\varepsilon_x} = \overline{x} - x = 4 - 3.2 = .8$
$\overline{\varepsilon_y} = \overline{y} - y$	$\overline{\varepsilon_y} = \overline{y} - y = 6 - 5.6 = .4$
$\underline{\varepsilon_x} = x - \underline{x}$	$\underline{\varepsilon_x} = x - \underline{x} = 3.2 - 2 = .2$
$\underline{\varepsilon_y} = y - \underline{y}$	$\underline{\varepsilon_y} = y - \underline{y} = 5.6 - 5 = .6$

Image Warping

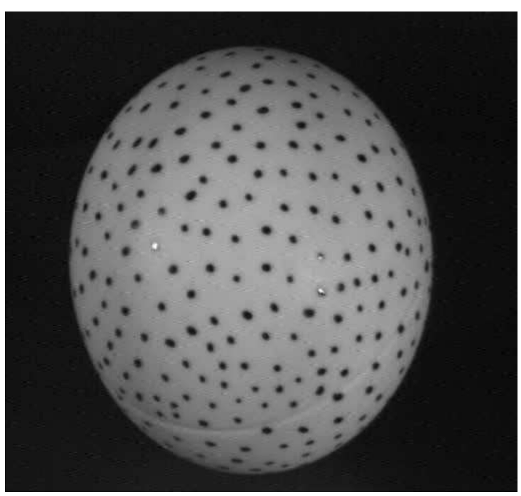




$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1+a_1 & a_2 \\ a_3 & 1+a_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1+a_1 & a_2 \\ a_3 & 1+a_4 \end{bmatrix}^{-1} \left(\begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right)$$

Motion Tracking



Motion tracking

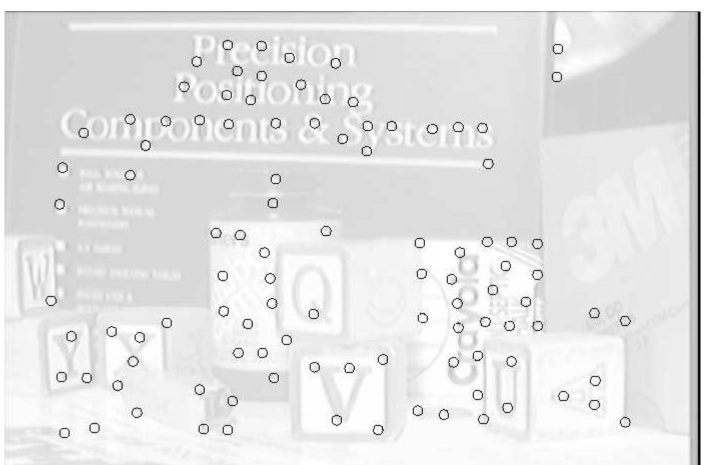
Suppose we have more than two images

- How to track a point through all of the images?
  - In principle, we could estimate motion between each pair of consecutive frames
  - Given point in first frame, follow arrows to trace out it's path
  - Problem: DRIFT
    - » small errors will tend to grow and grow over time—the point will drift way off course

Feature Tracking

- Choose only the points ("features") that are easily tracked
- How to find these features?
  - windows where $\sum \nabla I (\nabla I)^T$ has two large eigenvalues
- Called the Harris Corner Detector

Feature Detection



Tracking features

Feature tracking

- Compute optical flow for that feature for each consecutive H, I

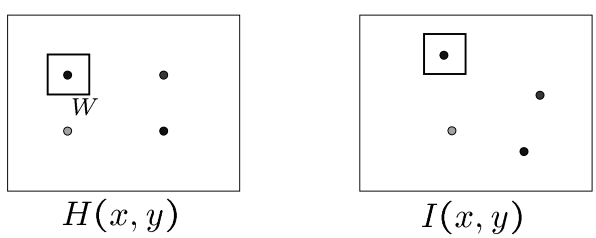
When will this go wrong?

- Occlusions—feature may disappear
  - need mechanism for deleting, adding new features
- Changes in shape, orientation
  - allow the feature to deform
- Changes in color
- Large motions
  - will pyramid techniques work for feature tracking?

Handling large motions

L-K requires small motion

- If the motion is much more than a pixel, use discrete **search** instead



- Given feature window W in H, find best matching window in I
- Minimize sum squared difference (SSD) of pixels in window

$$min_{(u,v)} \left\{ \sum_{(x,y)\in W} |I(x+u, y+v) - H(x,y)|^2 \right\}$$

- Solve by doing a search over a specified range of (u,v) values
  - this (u,v) range defines the **search window**

Tracking Over Many Frames

Feature tracking with m frames

1. Select features in first frame
2. Given feature in frame i, compute position in i+1
3. Select more features if needed
4. i = i + 1
5. If i < m, go to step 2

Issues

- Discrete search vs. Lucas Kanade?
  - depends on expected magnitude of motion
  - discrete search is more flexible
- Compare feature in frame i to i+1 or frame 1 to i+1?
  - affects tendency to drift..
- How big should search window be?
  - too small: lost features. Too large: slow

Incorporating Dynamics

Idea

- Can get better performance if we know something about the way points move
- Most approaches assume constant velocity

$$\begin{aligned} \dot{\mathbf{x}}_{i+1} &= \dot{\mathbf{x}}_i \\ \mathbf{x}_{i+1} &= 2\mathbf{x}_i - \mathbf{x}_{i-1} \end{aligned}$$

  or constant acceleration

$$\begin{aligned} \ddot{\mathbf{x}}_{i+1} &= \ddot{\mathbf{x}}_i \\ \mathbf{x}_{i+1} &= 3\mathbf{x}_i - 3\mathbf{x}_{i-1} + \mathbf{x}_{i-2} \end{aligned}$$

- Use above to predict position in next frame, initialize search