Optical Flow and Object Tracking

簡韶逸 Shao-Yi Chien

Department of Electrical Engineering

National Taiwan University

Slide Credits

- Lucas-Kanade algorithm from Prof. Yung-Yu Chuang's VFX course
- Feature tracker and optical flow from Prof. Jia-Bin Huang's Computer Vision
- Visual Tracking from Prof. Alexandre Alahi, Stanford Vision Lab
- Slides from members of Media IC and System Lab

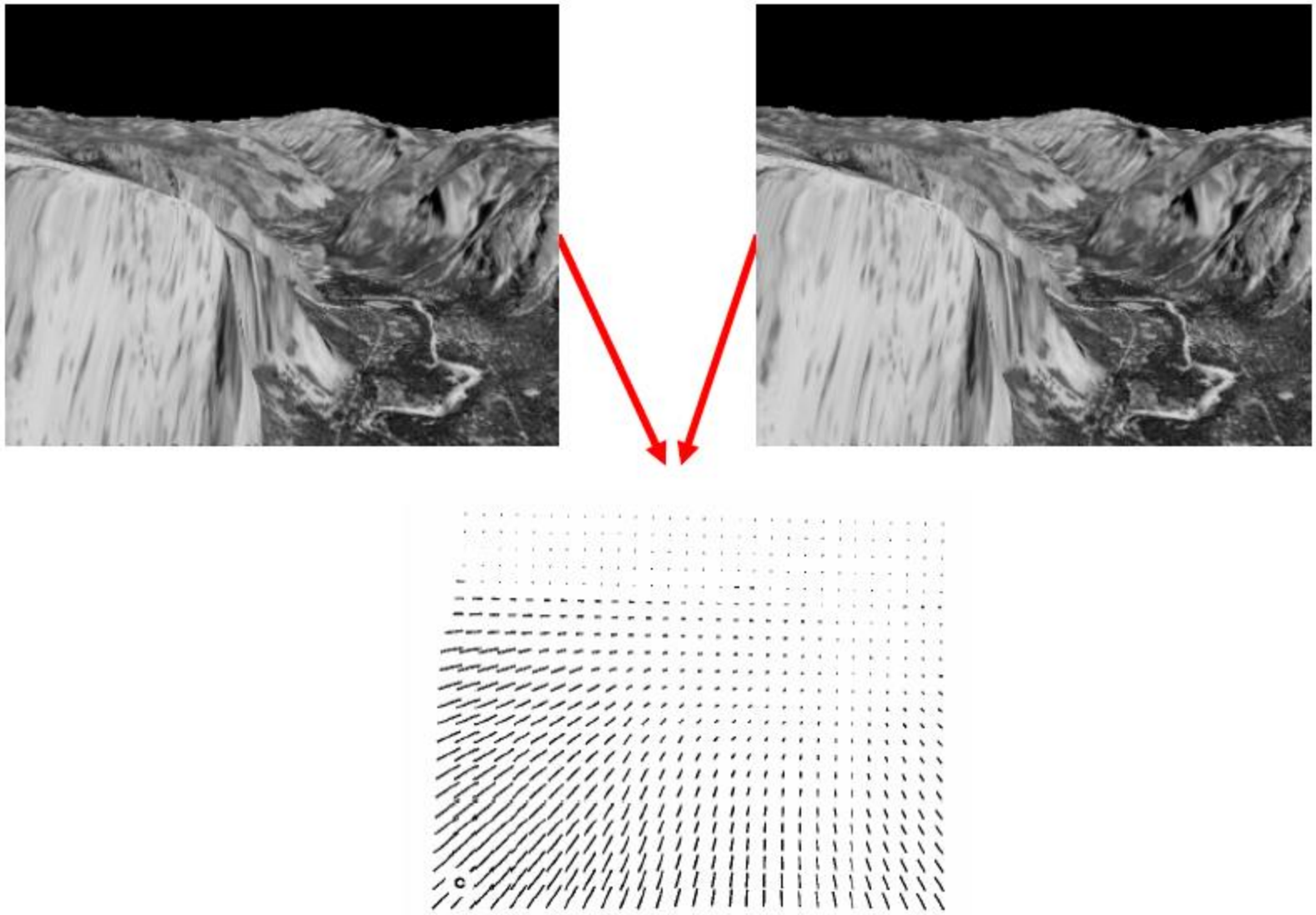
Outline

- Optical flow
  - Introduction
  - Lucas-Kanade algorithm
  - State-of-the-art optical flow
    - Epicflow
    - Flownet
    - PWC-Net

Introduction to Optical Flow

- Motion vector for each pixel
- Developed for 3-D motion estimation in the computer vision community
- Optical flow is caused by movement of intensity patterns in an image plane
- Optical flow is what we can only get from video frames

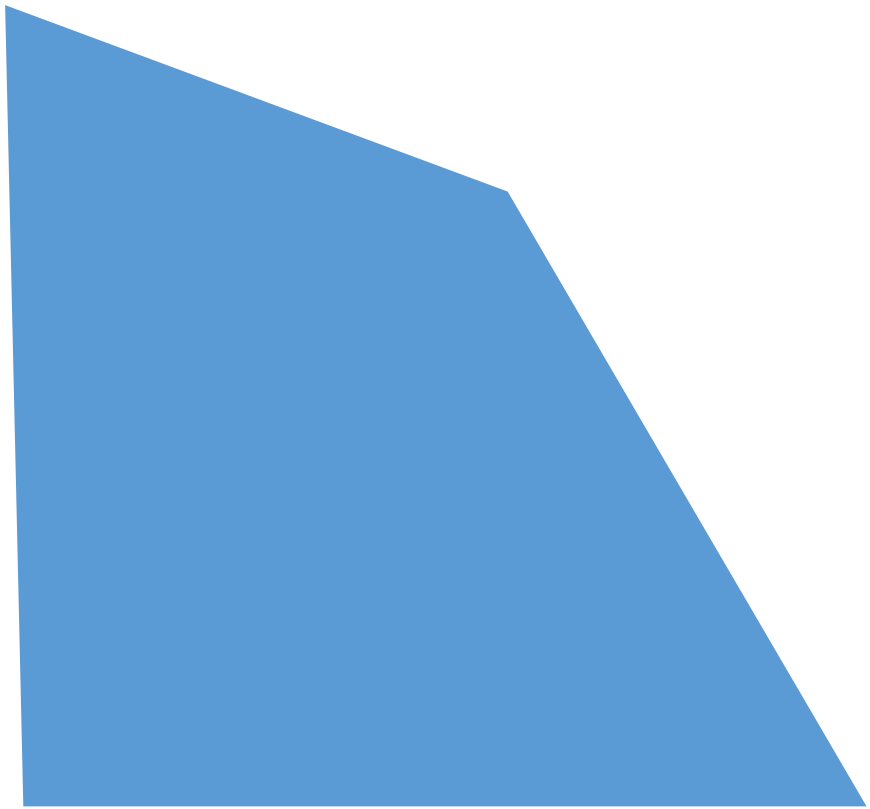
Introduction to Optical Flow



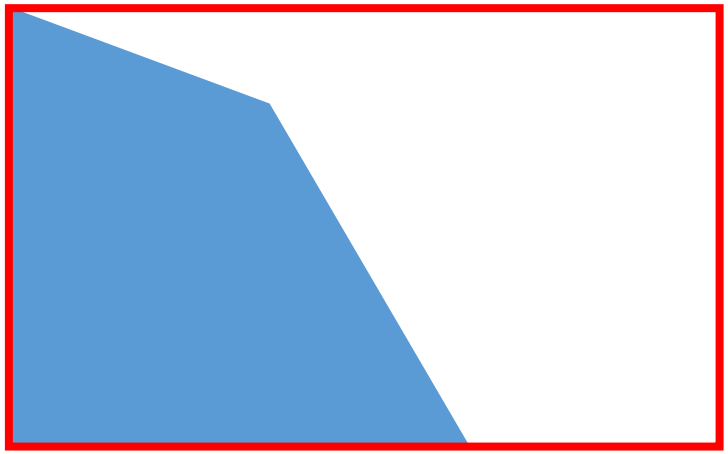
Introduction to Optical Flow

- Aperture problem
- Ill-pose inverse problem
  - Not well posed
  - Existance: No. Ex. rotating uniform sphere
  - Uniquness: No. Aperture problem
  - Continuity: No. Sensitive to noise

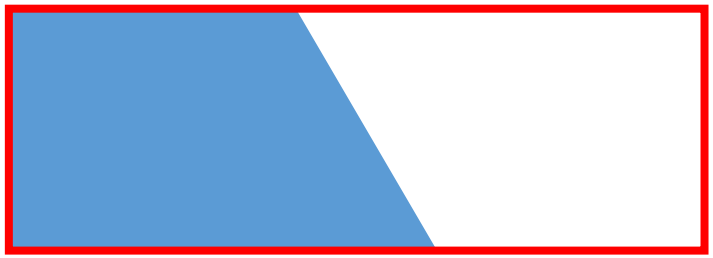
Aperture Problem



Aperture Problem



Aperture Problem



Basic Assumptions for Optical Flow

- Brightness consistency
- Spatial coherence
- Temporal persistence

Horn-Schunck Algorithm

- B. K. P. Horn and B. G. Schunck, “Determining optical flow,” Artificial Intelligence, vol. 17, pp. 185—203, 1981.

For image function $f(x, y, t)$

Invariance of brightness $\frac{df}{dt} = 0$

$$\frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v + \frac{\partial f}{\partial t} = 0$$

$$u = \frac{\partial x}{\partial t}, v = \frac{\partial y}{\partial t}$$

Smoothness constraint : to minimize $\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2$

Horn-Schunck Algorithm

For image function $f(x, y, t)$

Invariance of brightness $\frac{df}{dt} = 0$

$$\varepsilon_b^2 = \left(\frac{\partial f}{\partial x} u + \frac{\partial f}{\partial y} v + \frac{\partial f}{\partial t} \right)^2$$

$$\varepsilon_s^2 = \left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2$$

To minimize $\varepsilon = \sum_x \sum_y \varepsilon_b^2 + \alpha^2 \varepsilon_s^2$

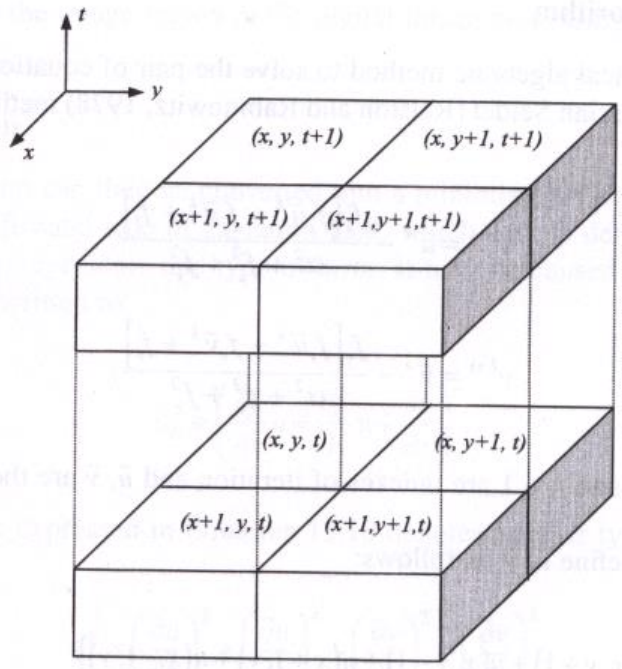
Iteration solution :

$$u^{k+1} = u^k - \frac{f_x \left[f_x u^k + f_y v^k + f_t \right]}{\alpha^2 + f_x^2 + f_y^2}$$

$$v^{k+1} = v^k - \frac{f_y \left[f_x u^k + f_y v^k + f_t \right]}{\alpha^2 + f_x^2 + f_y^2}$$

$$f_x = \frac{\partial f}{\partial x}, \; f_y = \frac{\partial f}{\partial y}, \; f_t = \frac{\partial f}{\partial t}$$

Horn-Schunck Algorithm

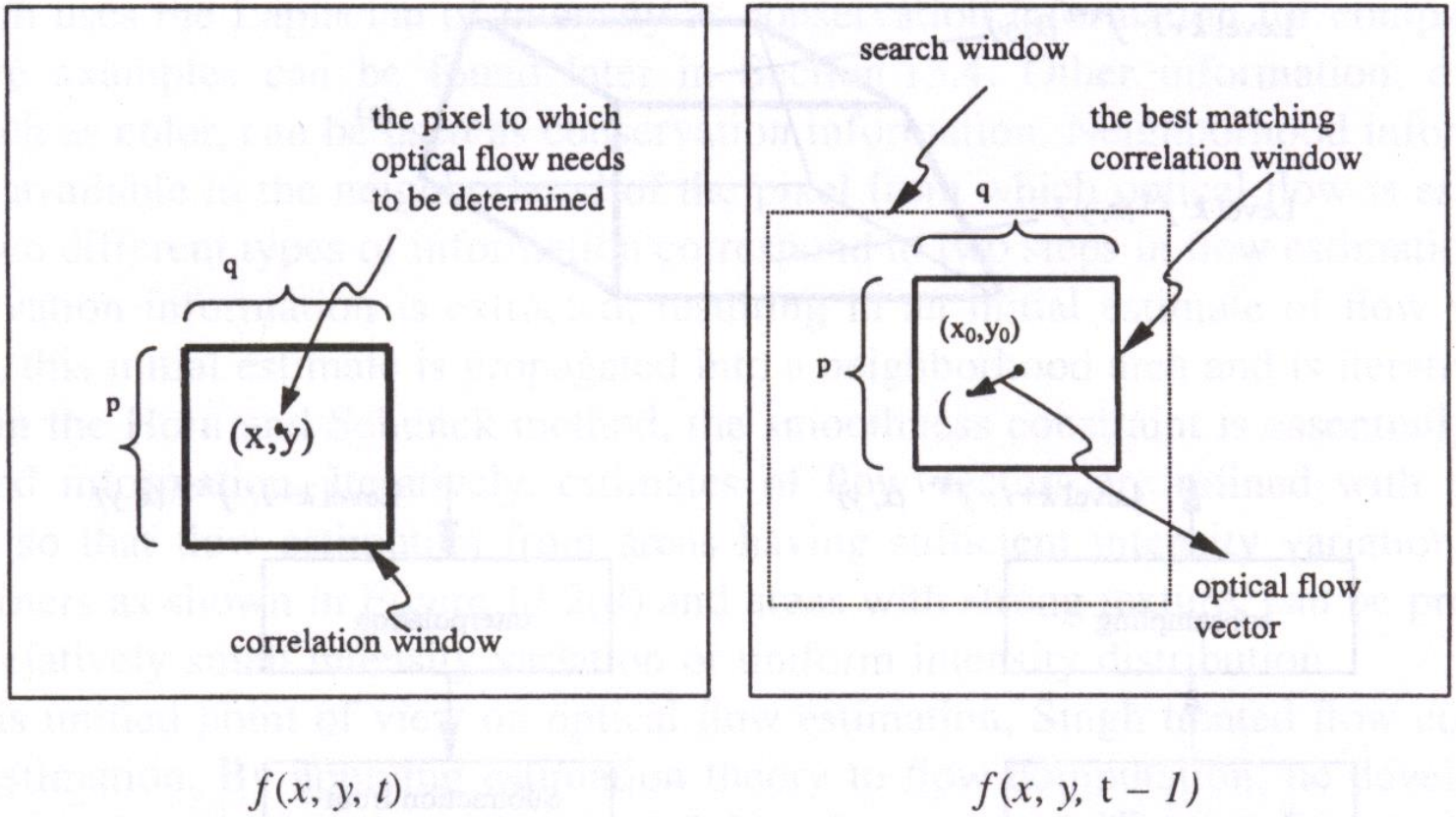


$$f_x = \frac{1}{4}\{[f(x+1,y,t)-f(x,y,t)]+[f(x+1,y,t+1)-f(x,y,t+1)]$$
$$+[f(x+1,y+1,t)-f(x,y,t)]+[f(x+1,y+1,t+1)-f(x,y+1,t+1)]\}$$

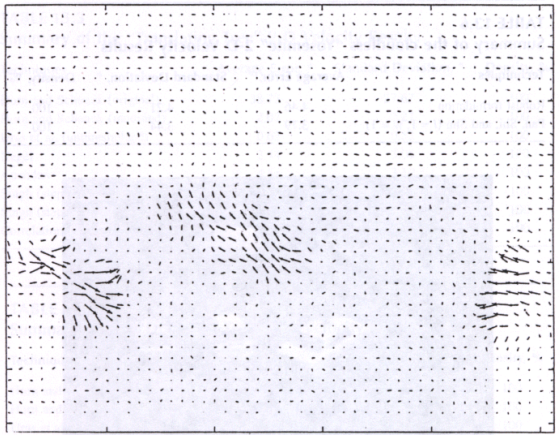
$$f_y = \frac{1}{4}\{[f(x,y+1,t)-f(x,y,t)]+[f(x+1,y+1,t)-f(x+1,y,t)]$$
$$+[f(x,y+1,t+1)-f(x,y,t+1)]+[f(x+1,y+1,t+1)-f(x+1,y,t+1)]\}$$

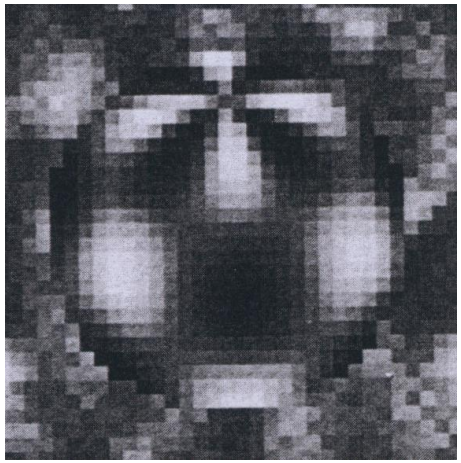
$$f_x = \frac{1}{4}\{[f(x,y,t+1)-f(x,y,t)]+[f(x+1,y,t+1)-f(x+1,y,t)]$$
$$+[f(x,y+1,t+1)-f(x,y+1,t)]+[f(x+1,y+1,t+1)-f(x+1,y+1,t)]\}$$

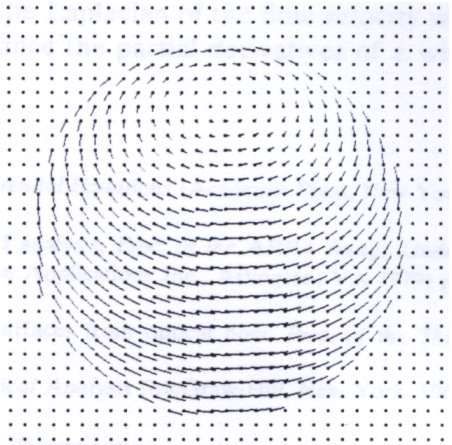
Correlation-Based Approach



Some Example

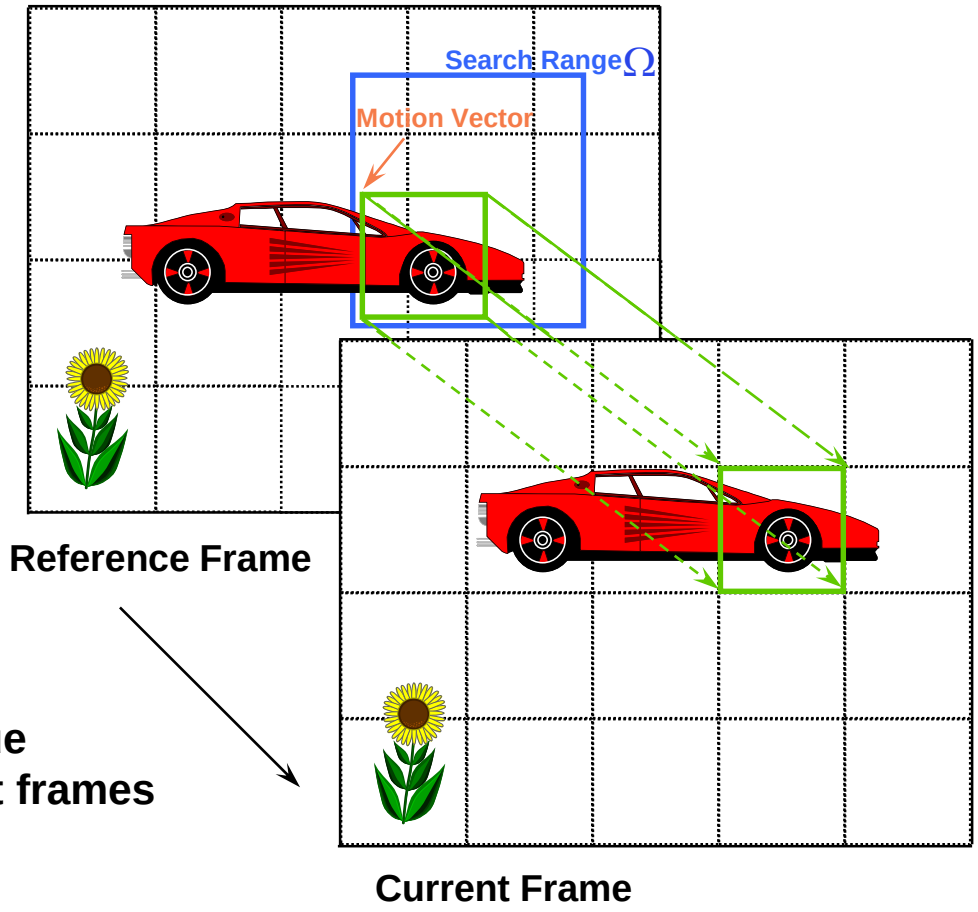






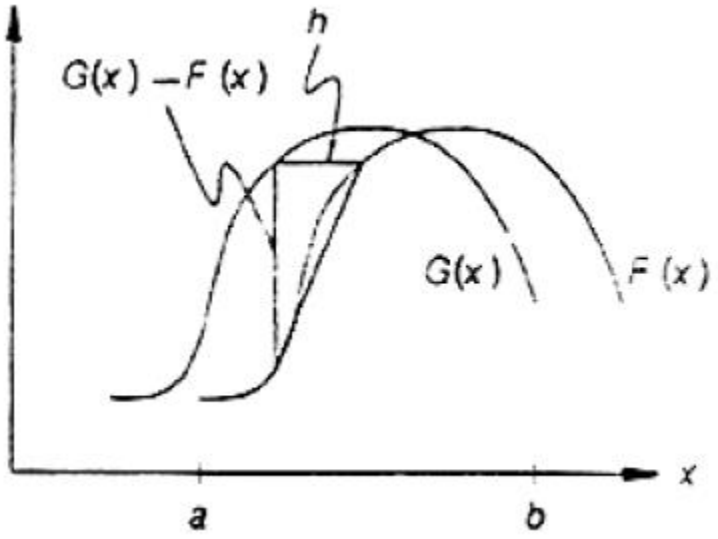
Needle diagram

Block-Matching Motion Estimation



Motion Vector
$V_t(p.q) = (Veci, Vecj)$
the location in the search range Ω
that has the maximum correlation value
between blocks in temporally adjacent frames

Lucas-Kanade Optical Flow



To minimize

$$E = \sum_x [F(x+h) - G(x)]^2 .$$

$$F(x+h) \approx F(x) + hF(x),$$

$$\begin{aligned} 0 &= \frac{\partial E}{\partial h} \\ &\approx \frac{\partial}{\partial h} \sum_x [F(x) + hF'(x) - G(x)]^2 \\ &= \sum_x 2F'(x)[F(x) + hF'(x) - G(x)] \end{aligned}$$

from which

$$h \approx \frac{\sum_x F'(x)[G(x) - F(x)]}{\sum_x F'(x)^2} .$$

Ref: Lucas and Kanade, "An iterative image registration technique with an application to stereo vision," IJCAI, 1981.

Lucas-Kanade Optical Flow

- Iterative solution

Add weights

$$w(x) = \frac{1}{|G(x) - F(x)|}.$$

$h_0 = 0,$

$$h_{k+1} = h_k + \frac{\sum_x w(x) F'(x + h_k)[G(x) - F(x + h_k)]}{\sum_x w(x) F'(x + h_k)^2},$$

- More dimensions

$$h = \left[\sum_x \left(\frac{\partial F}{\partial x}\right)^T [G(x) - F(x)]\right]\left[\sum_x \left(\frac{\partial F}{\partial x}\right)^T \left(\frac{\partial F}{\partial x}\right)\right]^{-1}$$

Ref: Simon Baker and Iain Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," IJCV2004.

Lucas-Kanade Algorithm

$$E(u,v) = \sum_{x,y} \left(I(x+u, y+v) - T(x,y) \right)^2$$

$$I(x+u, y+v) \approx I(x,y) + uI_x + vI_y$$

$$= \sum_{x,y} \left(I(x,y) - T(x,y) + uI_x + vI_y \right)^2$$

$$0 = \frac{\partial E}{\partial u} = \sum_{x,y} 2I_x \left(I(x,y) - T(x,y) + uI_x + vI_y \right)$$

$$0 = \frac{\partial E}{\partial v} = \sum_{x,y} 2I_y \left(I(x,y) - T(x,y) + uI_x + vI_y \right)$$

Lucas Kanade Algorithm

$$0 = \frac{\partial E}{\partial u} = \sum_{x,y} 2I_x \left(I(x,y) - T(x,y) + uI_x + vI_y \right)$$

$$0 = \frac{\partial E}{\partial v} = \sum_{x,y} 2I_y \left(I(x,y) - T(x,y) + uI_x + vI_y \right)$$

$$\begin{cases} \sum_{x,y} I_x^2 u & + \sum_{x,y} I_x I_y v = \sum_{x,y} I_x \left(T(x,y) - I(x,y) \right) \\ \sum_{x,y} I_x I_y u & + \sum_{x,y} I_y^2 v = \sum_{x,y} I_y \left(T(x,y) - I(x,y) \right) \end{cases}$$

$$\begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_{x,y} I_x \left(T(x,y) - I(x,y) \right) \\ \sum_{x,y} I_y \left(T(x,y) - I(x,y) \right) \end{bmatrix}$$

Lucas Kanade Algorithm

iterate

shift I(x,y) with (u,v)

compute gradient image I_x, I_y

compute error image T(x,y)-I(x,y)

compute Hessian matrix

solve the linear system

(u,v)=(u,v)+(Δu,Δv)

until converge

$$\begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_{x,y} I_x \big(T(x,y) - I(x,y)\big) \\ \sum_{x,y} I_y \big(T(x,y) - I(x,y)\big) \end{bmatrix}$$

Lucas Kanade Algorithm: Parametric

$$E(u,v) = \sum_{x,y} \left(I(x+u, y+v) - T(x,y) \right)^2$$

$$\Rightarrow \quad E(\mathbf{p}) = \sum_{\mathbf{x}} \left(I(\mathbf{W(x;p)}) - T(\mathbf{x}) \right)^2$$ ← Our goal is to find $\mathbf{p}$ to minimize $\mathbf{E(p)}$

for all $\mathbf{x}$ in T's domain

translation $\mathbf{W(x;p)} = \begin{pmatrix} x + d_x \\ y + d_y \end{pmatrix}, p = (d_x, d_y)^T$

affine $\mathbf{W(x;p)} = \mathbf{Ax + d} = \begin{pmatrix} 1+d_{xx} & d_{xy} & d_x \\ d_{yx} & 1+d_{yy} & d_y \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix},$

$$p = (d_{xx}, d_{xy}, d_{yx}, d_{yy}, d_x, d_y)^T$$

Lucas Kanade Algorithm: Parametric

minimize $\sum_{\mathbf{x}} \left(I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x}) \right)^2$

with respect to $\Delta\mathbf{p}$

$$\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p}) \approx \mathbf{W}(\mathbf{x};\mathbf{p}) + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p}$$

$$I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) \approx I(\mathbf{W}(\mathbf{x};\mathbf{p}) + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p})$$

$$\approx I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \frac{\partial I}{\partial \mathbf{x}} \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p}$$

→ minimize $\sum_{\mathbf{x}} \left(I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right)^2$

Lucas Kanade Algorithm: Parametric

warped image

target image

image gradient

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right]^2$$

Jacobian of the warp

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}} \\ \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_n} \\ \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_n} \end{pmatrix}$$

Lucas Kanade Algorithm: Parametric

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}} \\ \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_n} \\ \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_n} \end{pmatrix}$$

For example, for affine

$$\mathbf{W(x;p)} = \begin{pmatrix} 1+d_{xx} & d_{xy} & d_x \\ d_{yx} & 1+d_{yy} & d_y \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} (1+d_{xx})x + d_{xy}y + d_x \\ d_{yx}x + (1+d_{yy})y + d_y \end{pmatrix}$$

$$\rightarrow \quad \frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix}$$

$$d_{xx}\ d_{yx}\ d_{xy}\ d_{yy}\ d_x\ d_y$$

Lucas Kanade Algorithm: Parametric

$$\arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}} \left(I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right)^2$$

$$\rightarrow \quad 0 = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right]$$

$$\Delta\mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$

(Approximated) Hessian $\quad \mathbf{H} = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$

Lucas Kanade Algorithm

iterate

1) warp I with W(x;p)
2) compute error image T(x,y)-I(W(x,p))
3) compute gradient image ∇I with W(x,p)
4) evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at (x;p)
5) compute $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
6) compute Hessian
7) compute $\sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$
8) solve $\Delta \mathbf{p}$
9) update p by p+ $\Delta \mathbf{p}$

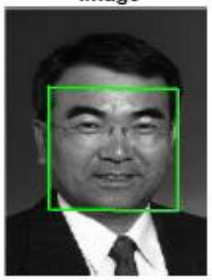
until converge

$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$

Template

$T(\mathbf{x})$

Image



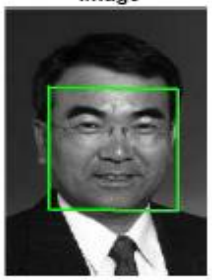
$I(\mathbf{x})$

$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$

Template

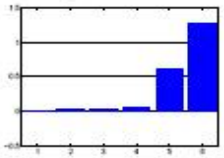
$T(\mathbf{x})$

Image



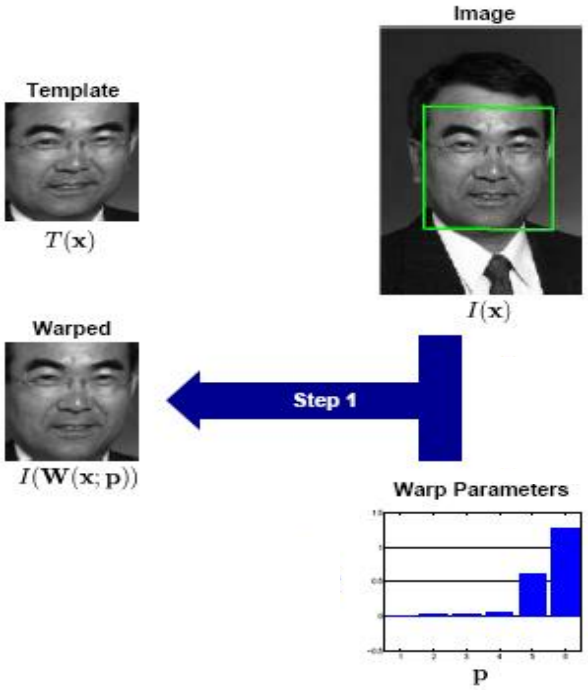
$I(\mathbf{x})$

Warp Parameters

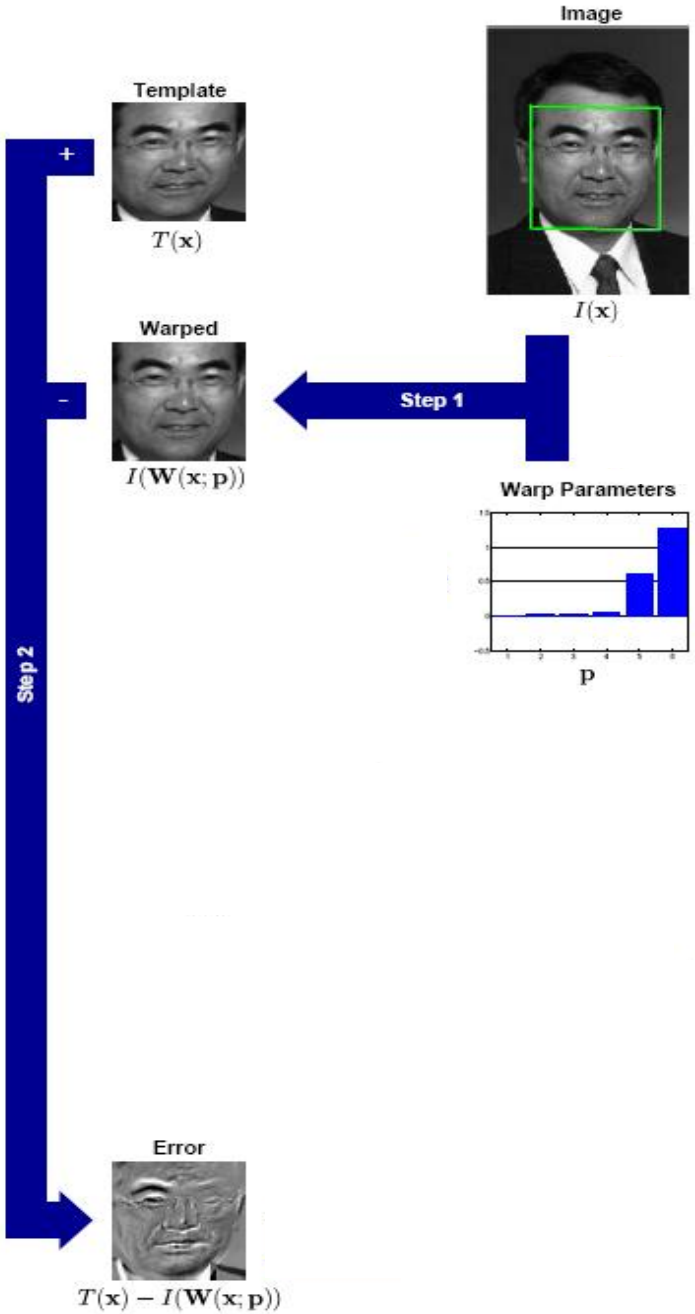


$\mathbf{p}$

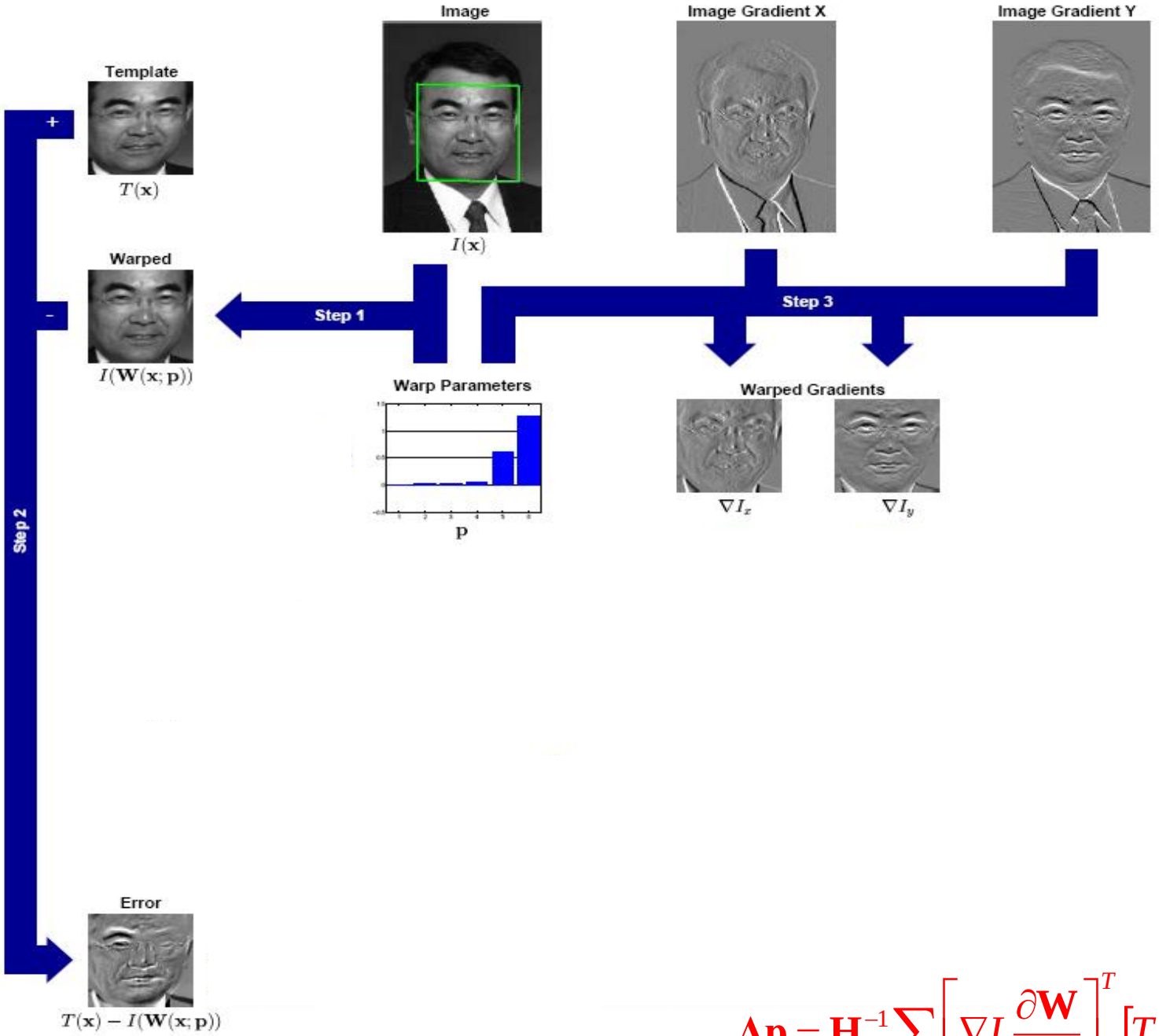
$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$



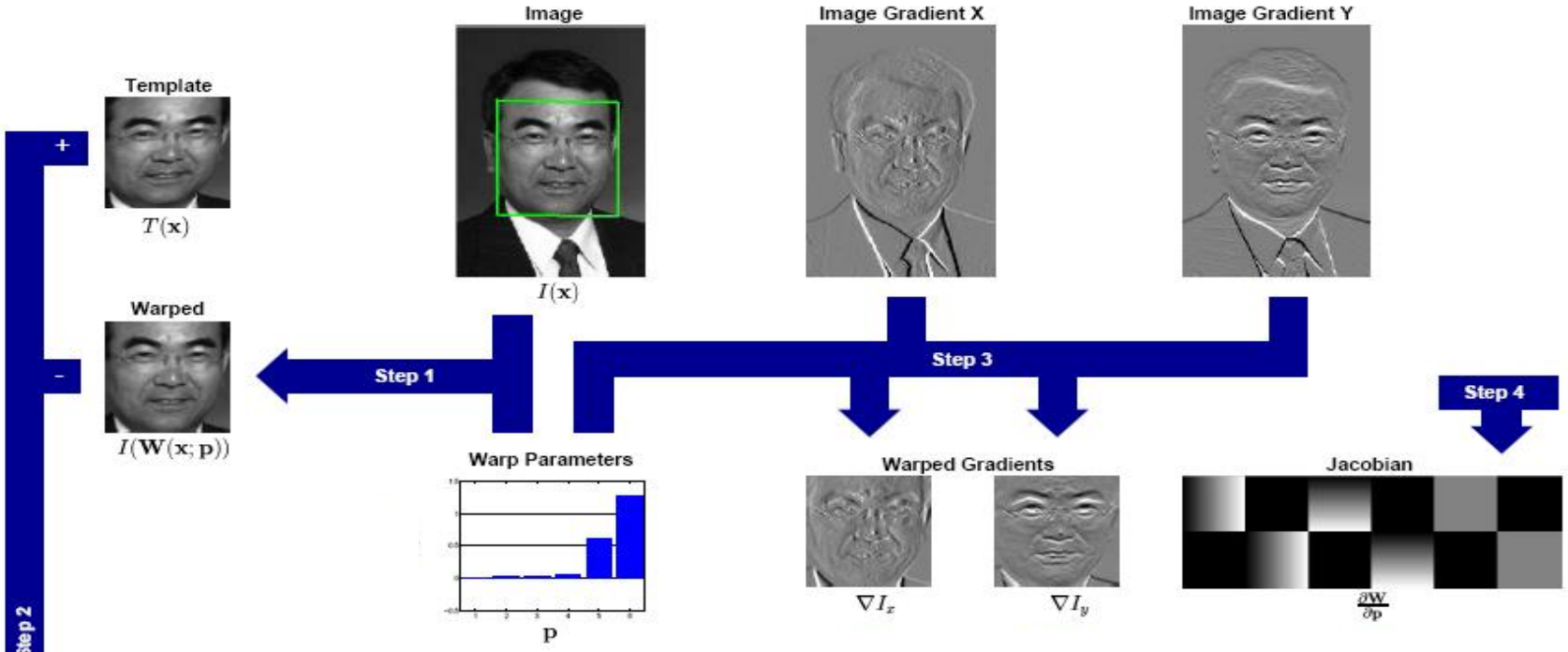
$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$



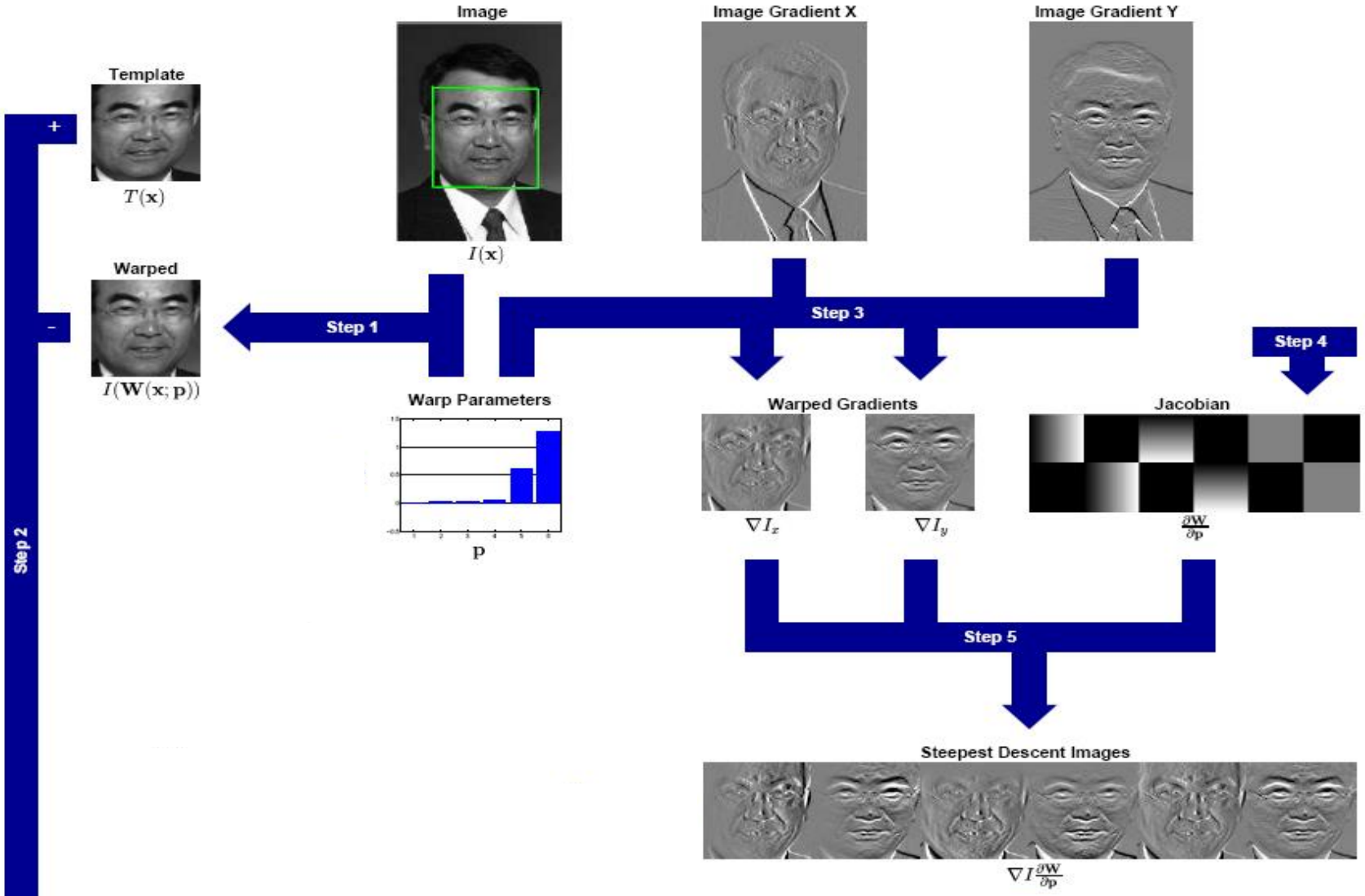
$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$



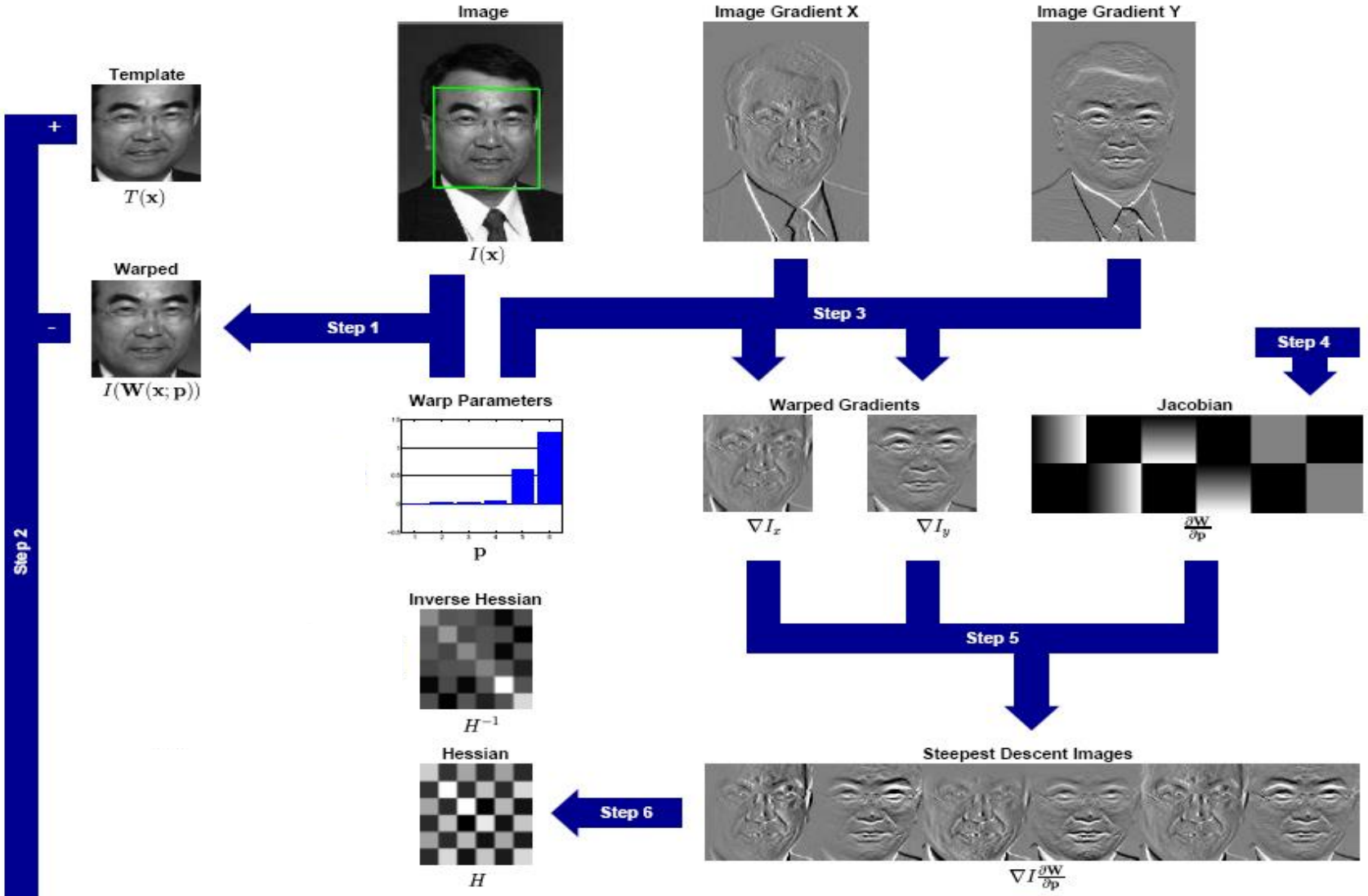
$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$



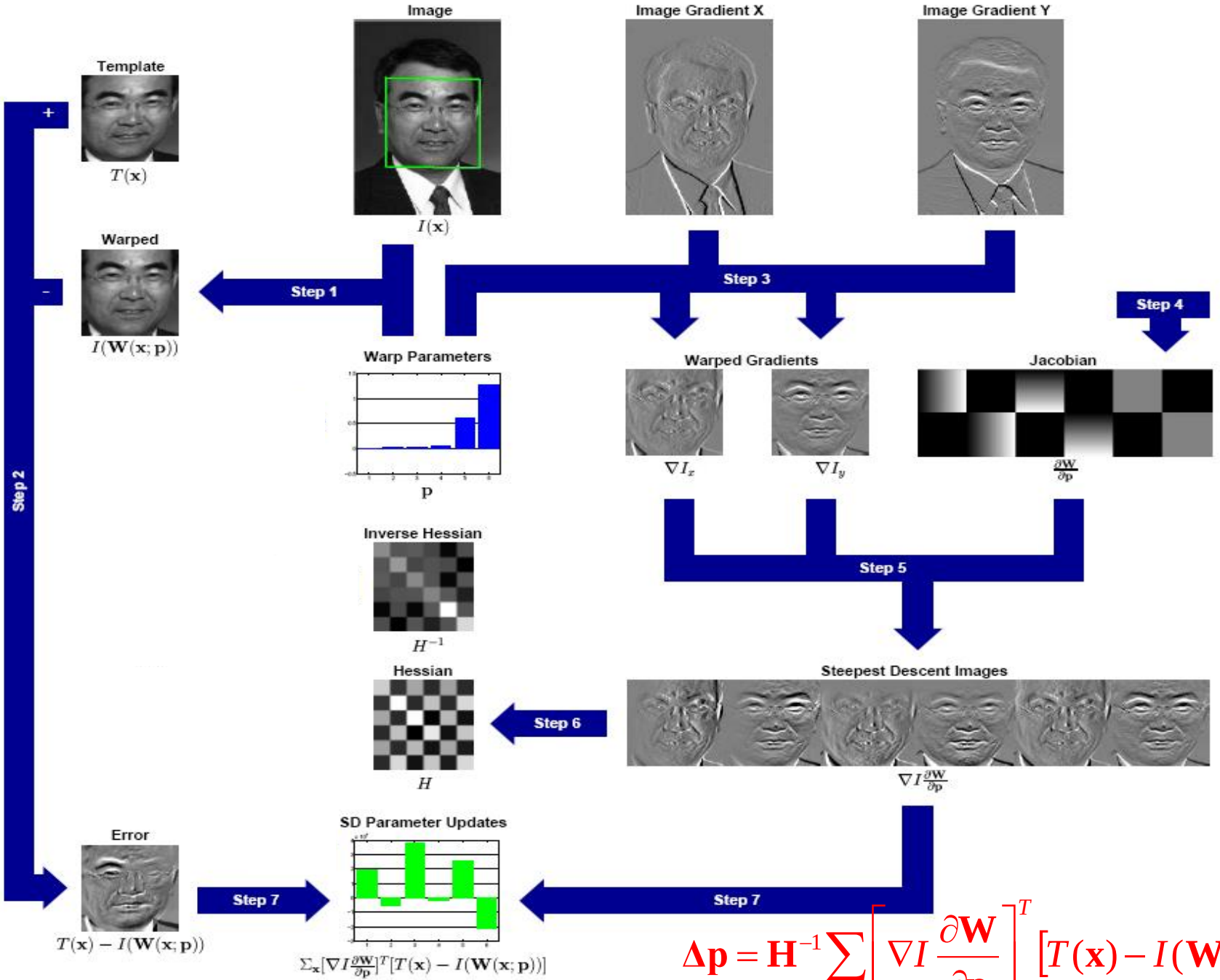
$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$



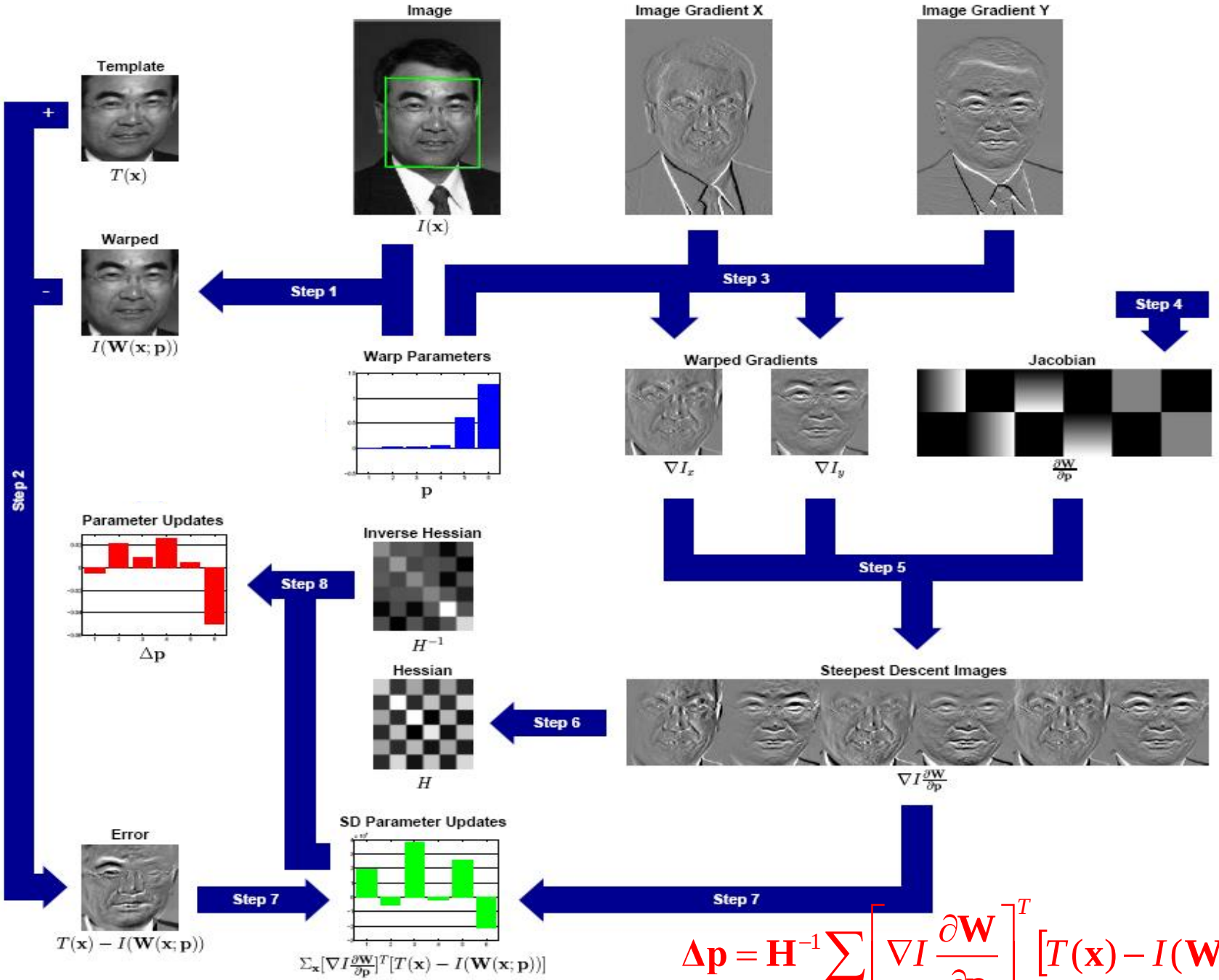
$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$



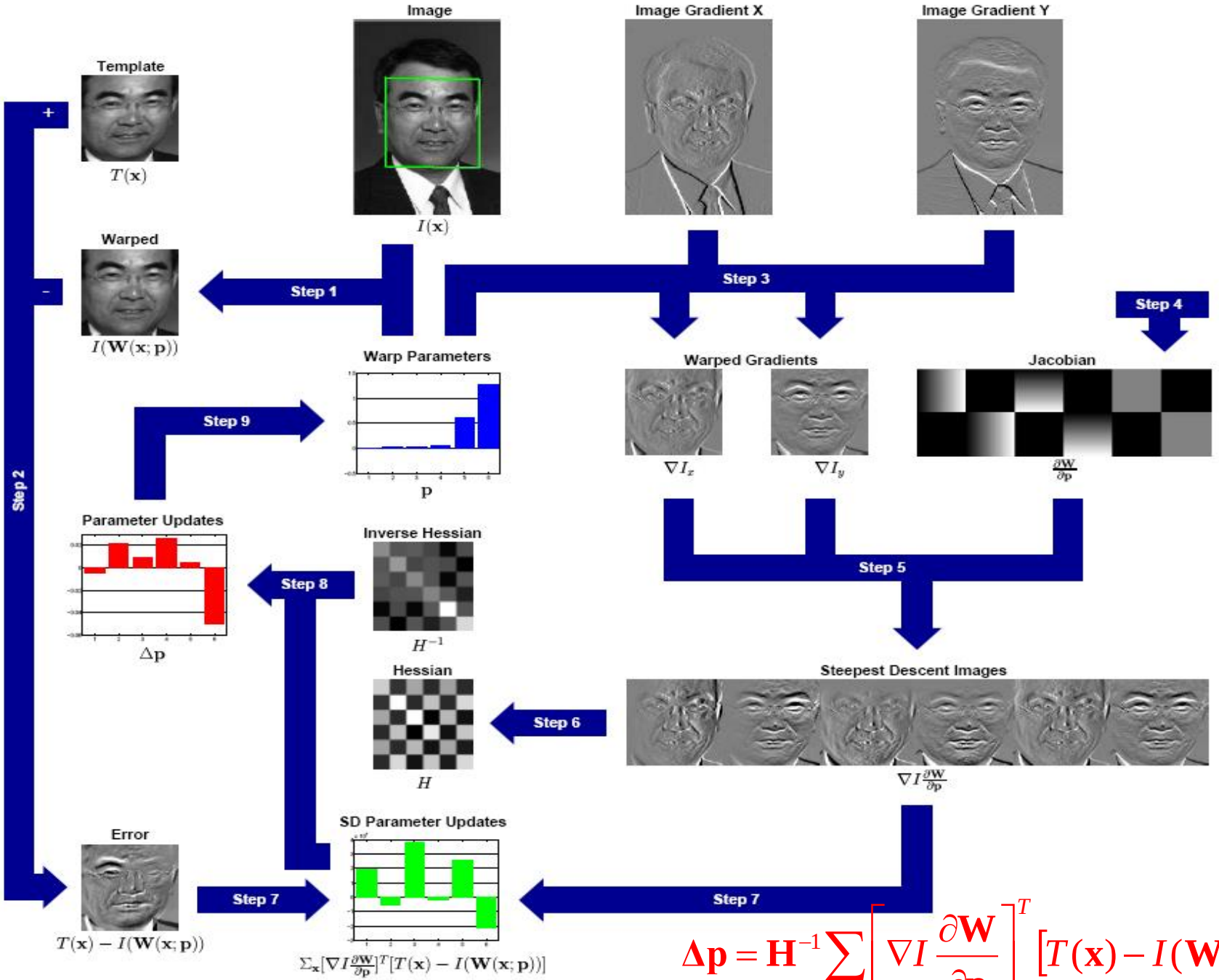
$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$



$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$

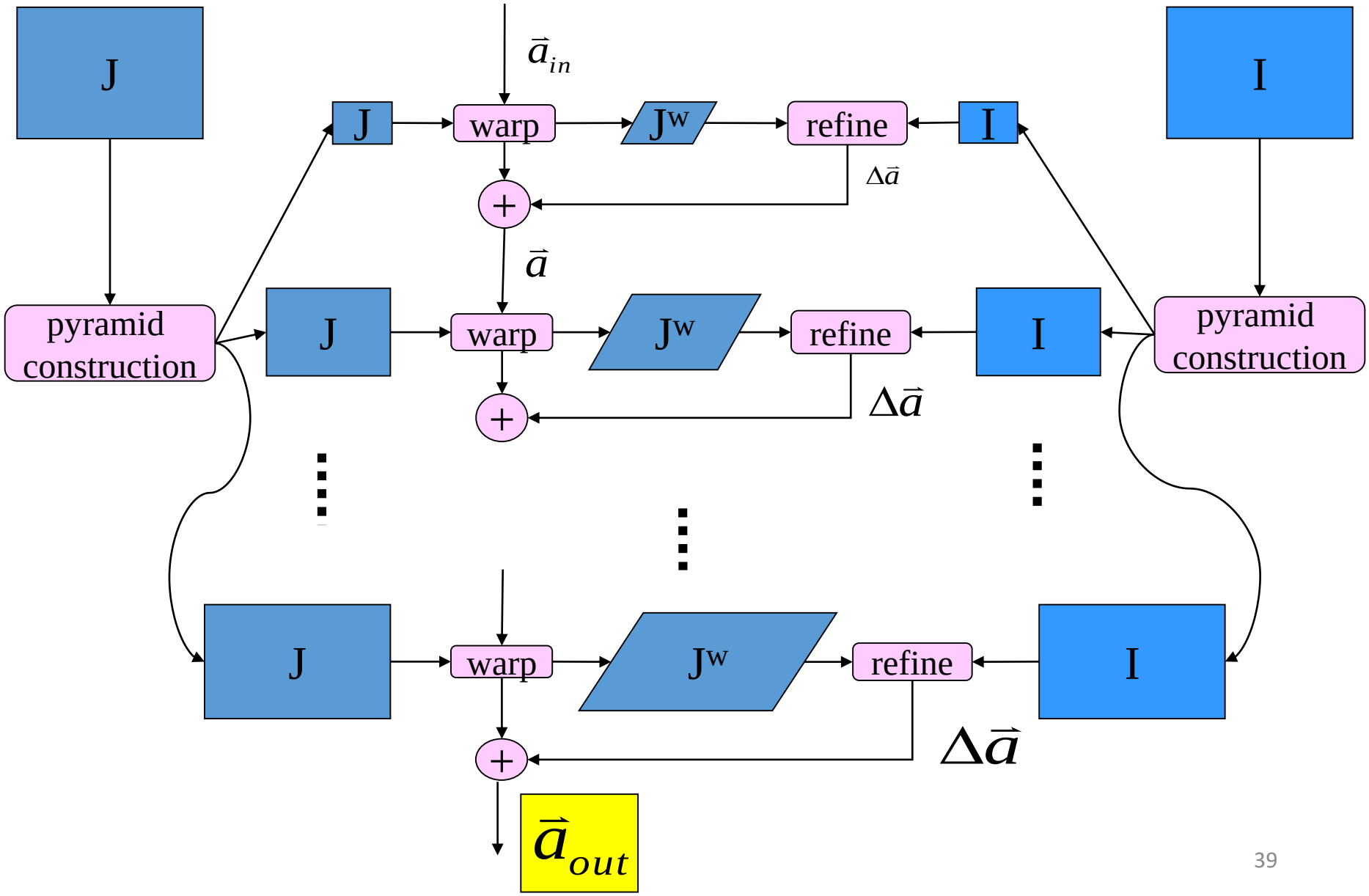


$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$



$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$

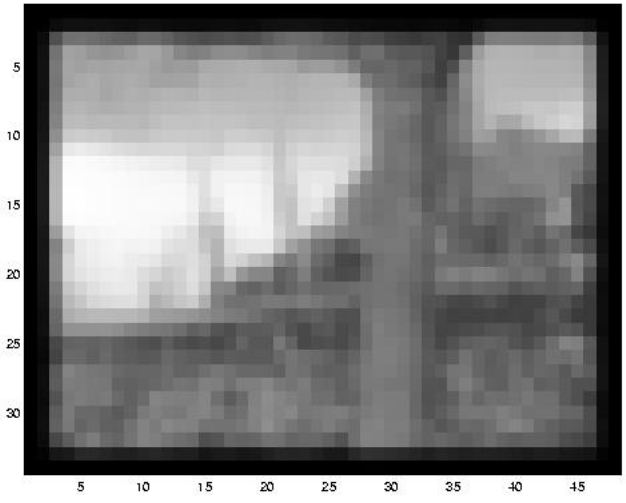
Coarse-to-Fine Strategy

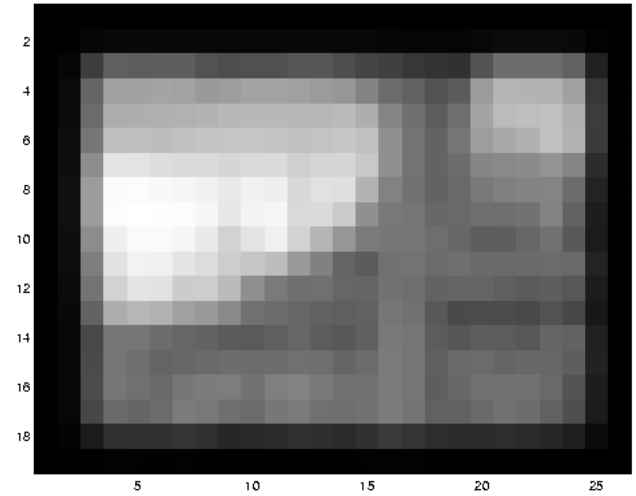


Example

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

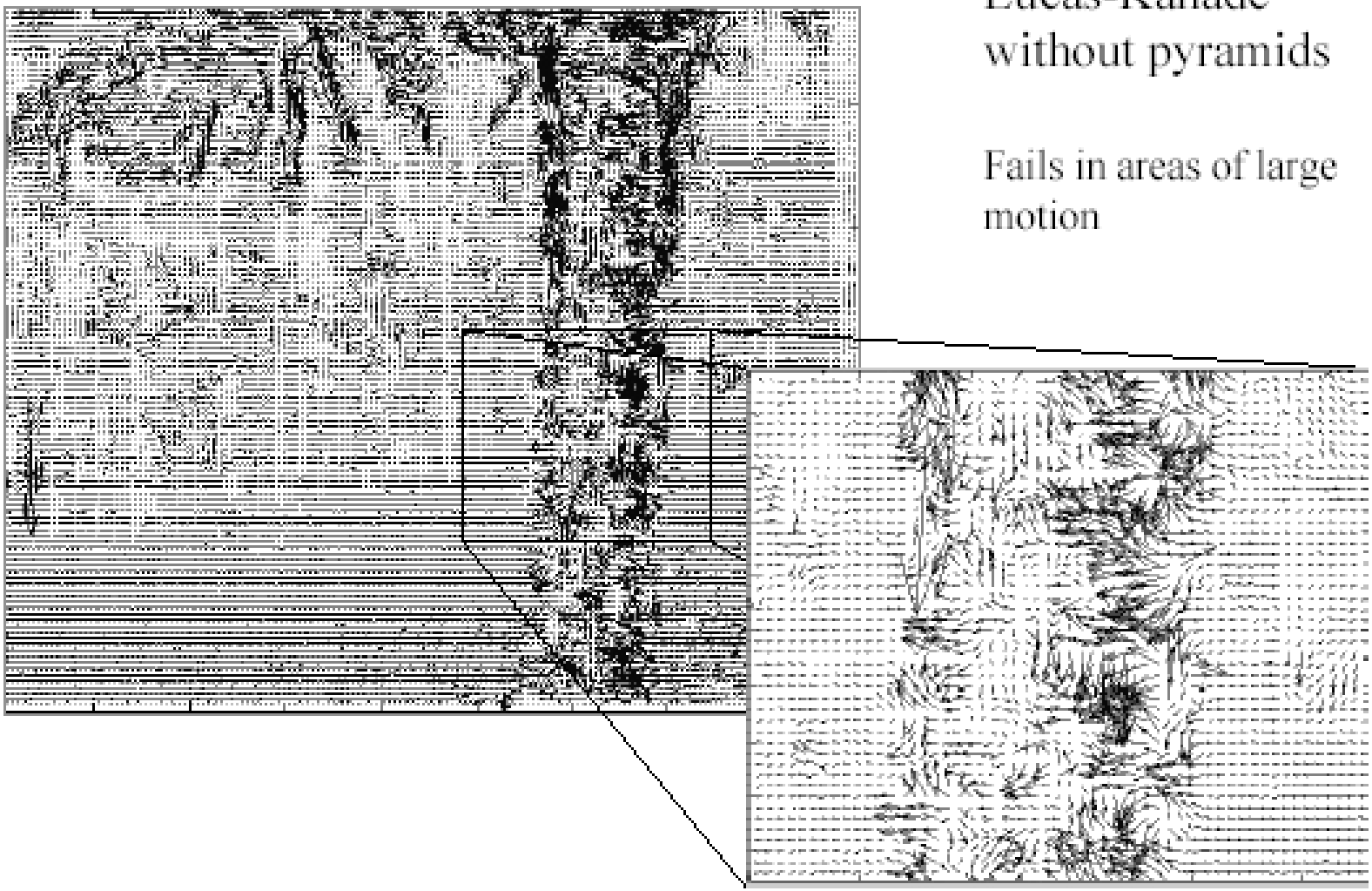
Multi-resolution registration





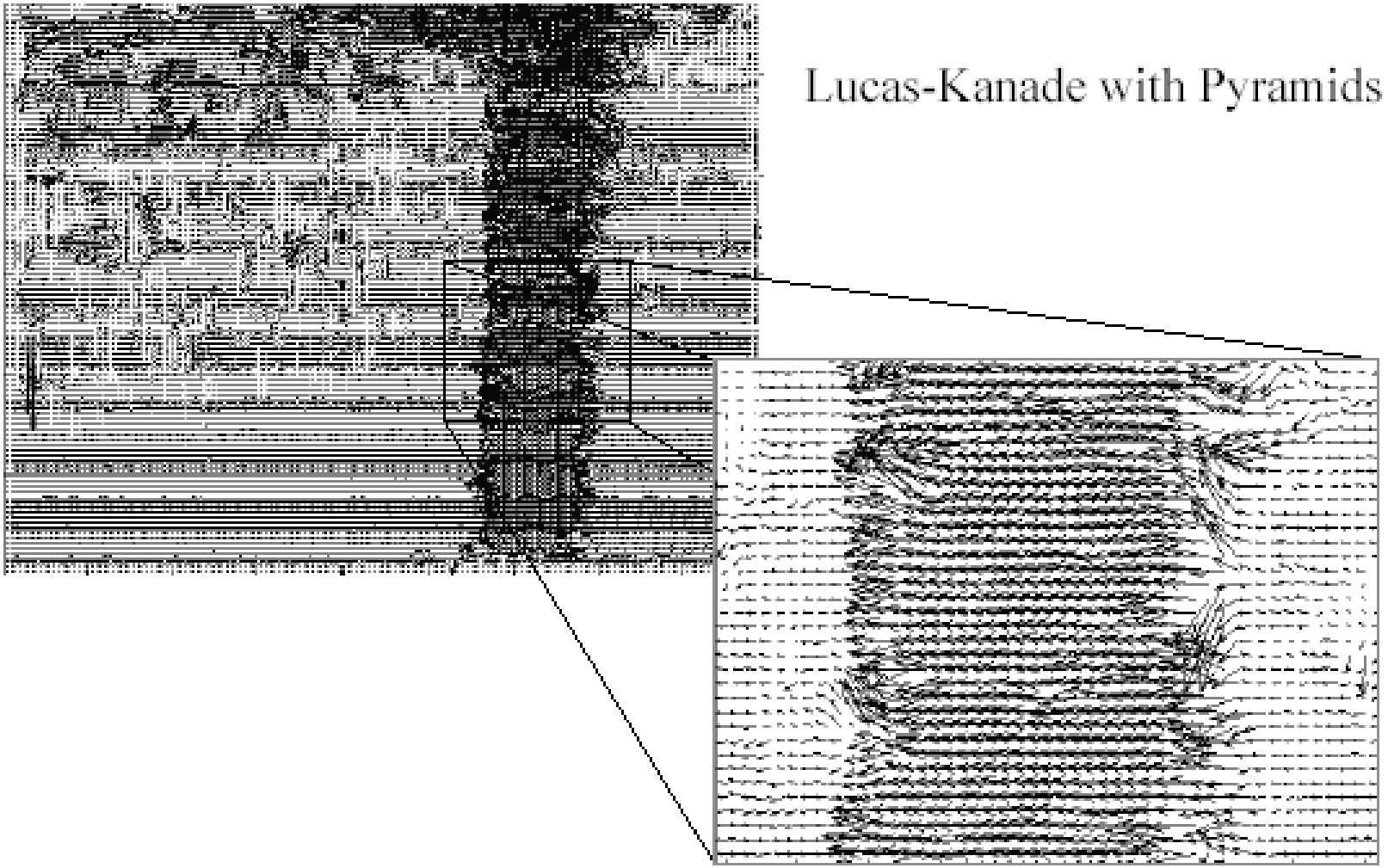
* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

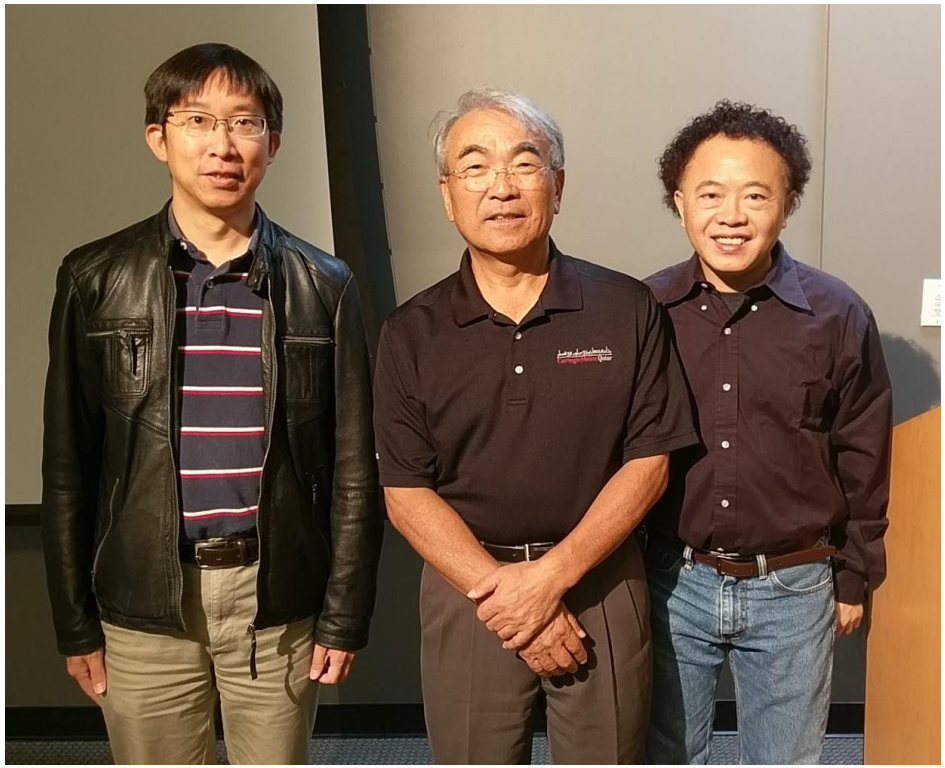
Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Lucas-Kanade Optical Flow

- https://youtu.be/D7r3-fHXvRU?t=1h40m54s



Errors of Lucas-Kanade

- The motion is large
  - Possible Fix: Keypoint matching
- A point does not move like its neighbors
  - Possible Fix: Region-based matching
- Brightness constancy does not hold
  - Possible Fix: Gradient constancy

Epicflow

- Main remaining problems:
  - large displacements
  - occlusions
  - motion discontinuities

- **EpicFlow**

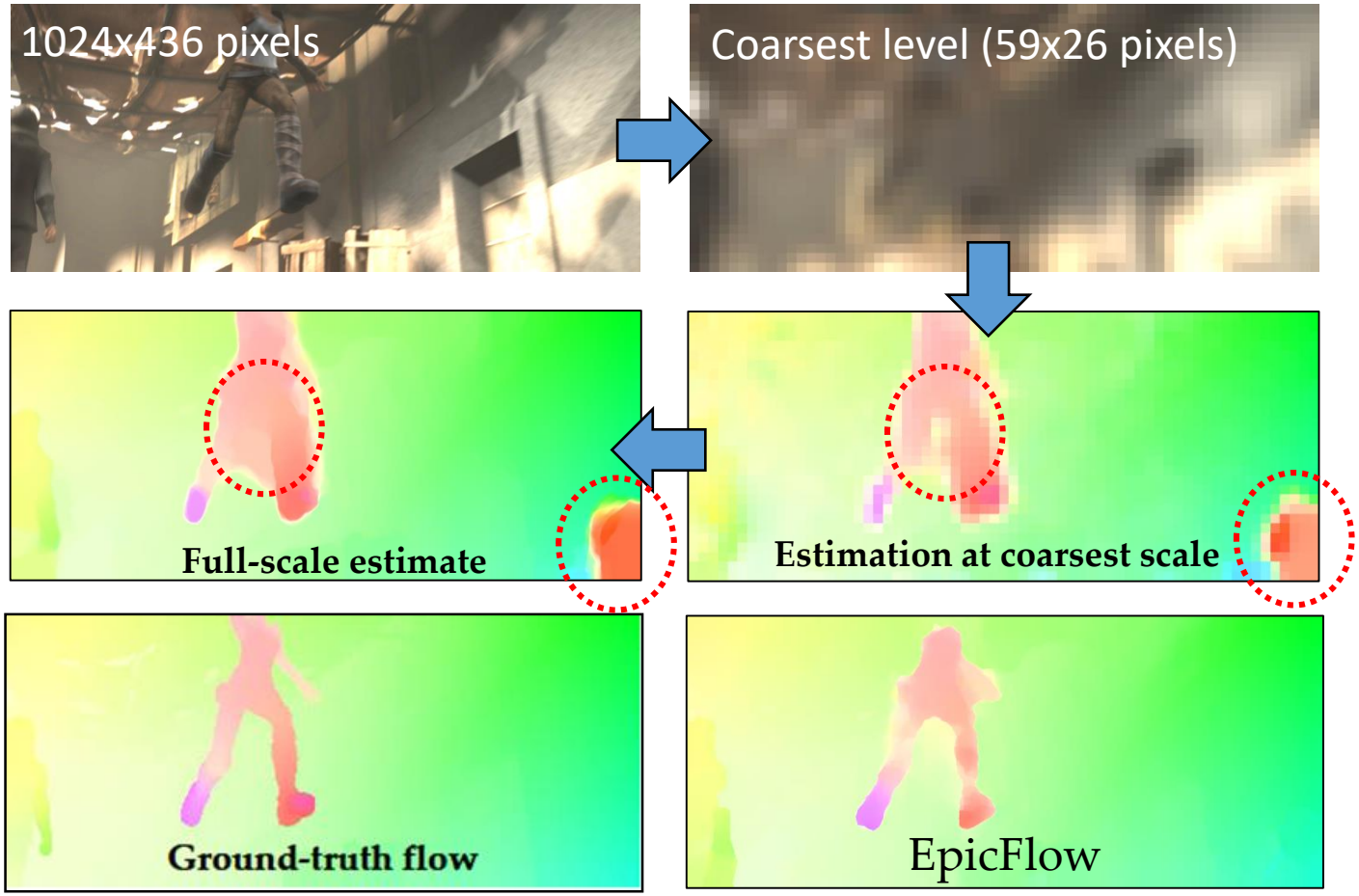
  Epic: Edge-Preserving Interpolation of Correspondences

- leverages state-of-the-art matching algorithm
  - invariant to large displacements
- incorporate an edge-aware distance:
  - handles occlusions and motion discontinuities
- state-of-the-art results

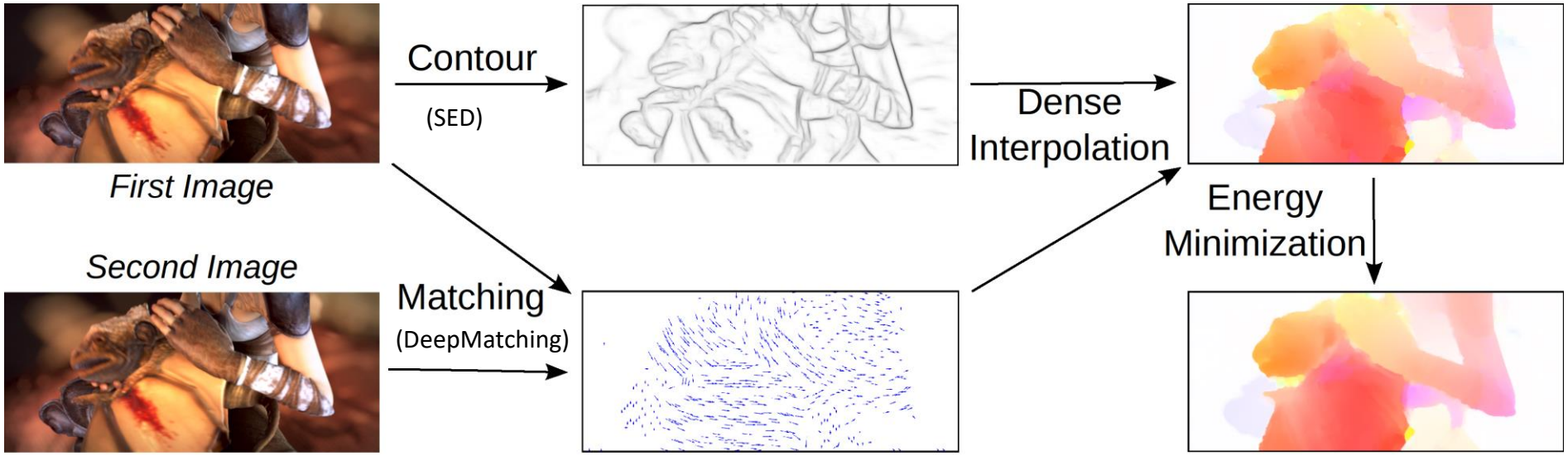
Ref: J. Revaud, P. Weinzaepfel, Z. Harchaoui and C. Schmid, "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow," *CVPR 2015*.

Epicflow

- Problems with coarse-to-fine:
  - flow discontinuities overlap at coarsest scales
  - errors are propagated across scales
  - no theoretical guarantees or proof of convergence!



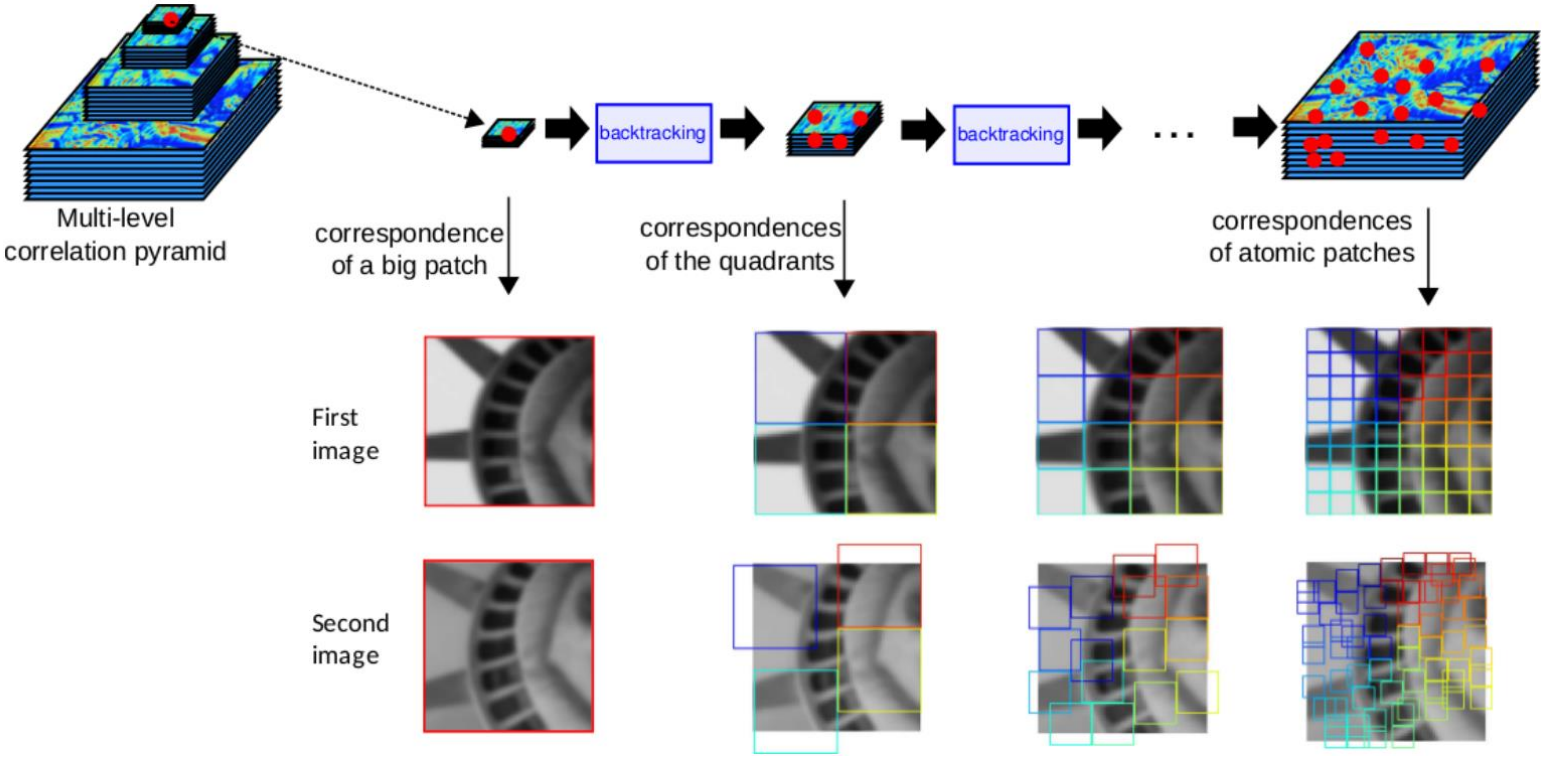
Epicflow: Overview



- Avoid coarse-to-fine scheme

Epicflow: DeepMatching

- Based on correlation of SIFT features of a patch



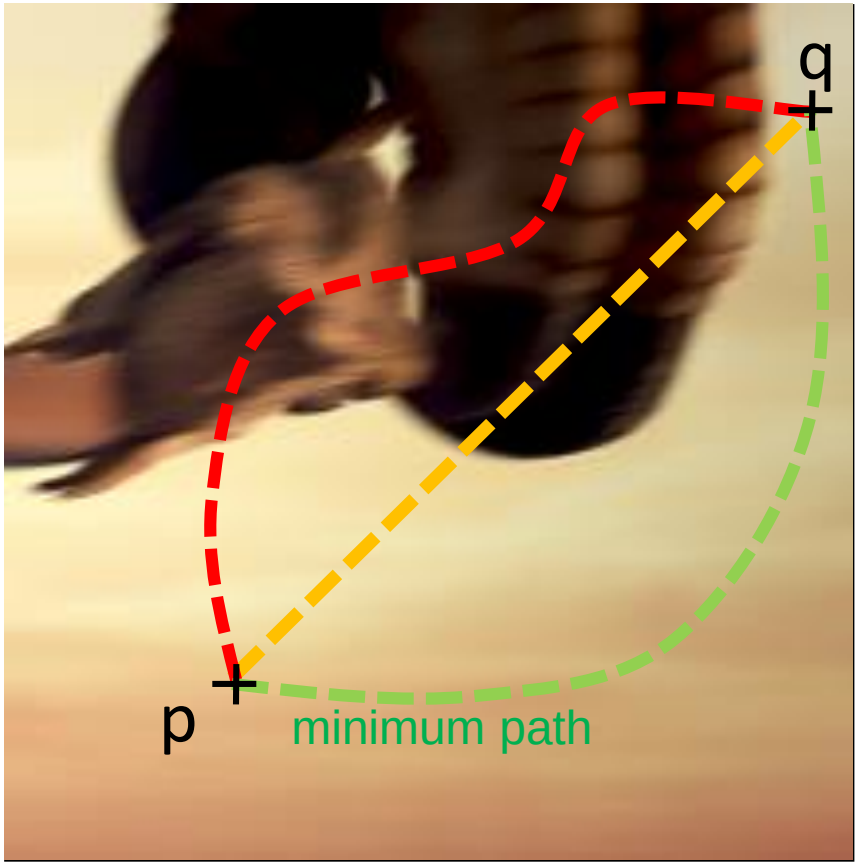
Epicflow: Dense Interpolation

- Replace Euclidean distance with edge-aware distance to find NNs

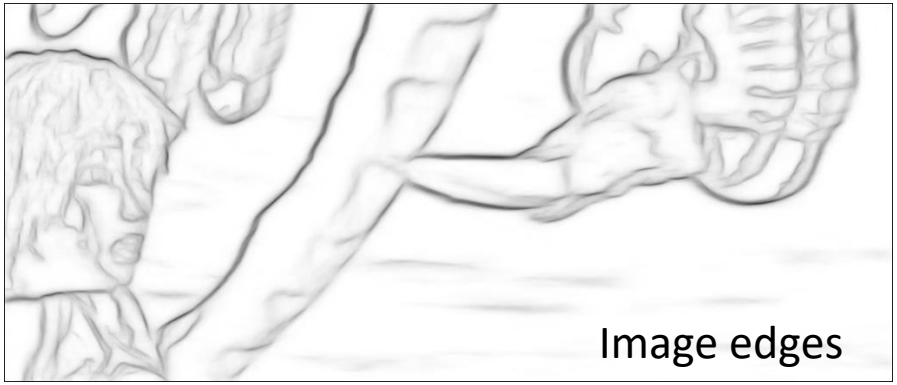
- Geodesic distance:
  - Anisotropic cost map = **image edges**

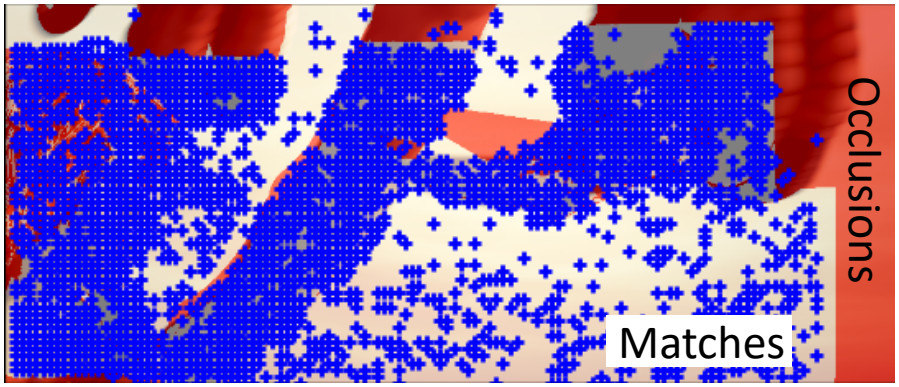
  ►Cost of a path: sum of the cost of all traversed pixels

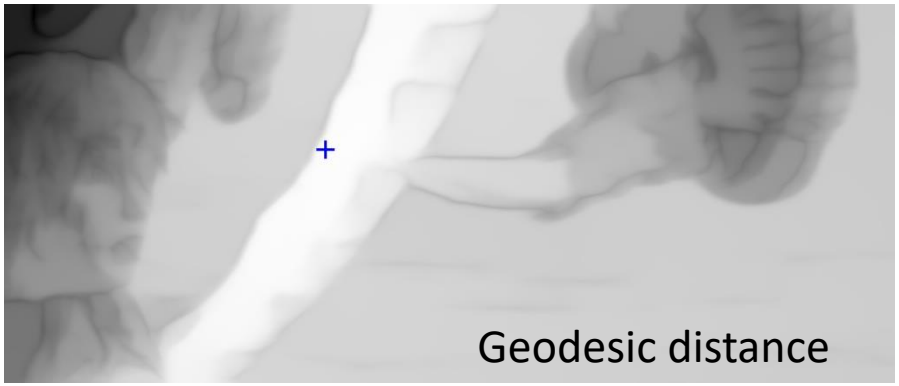
  ►Geodesic distance: minimum cost among all possible paths

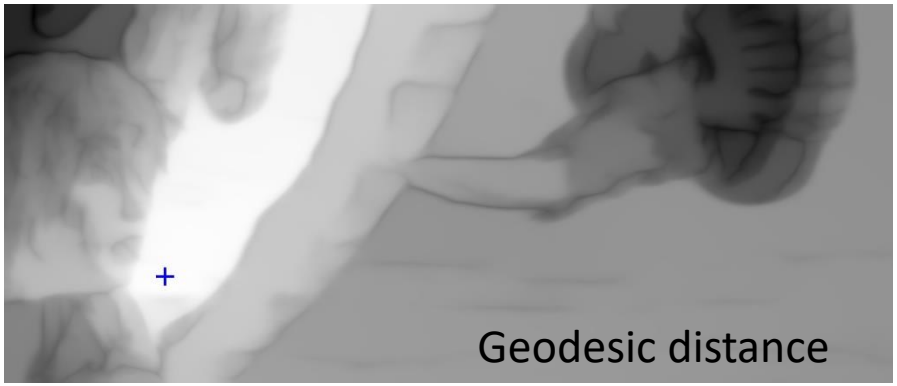


Geodesic Distance

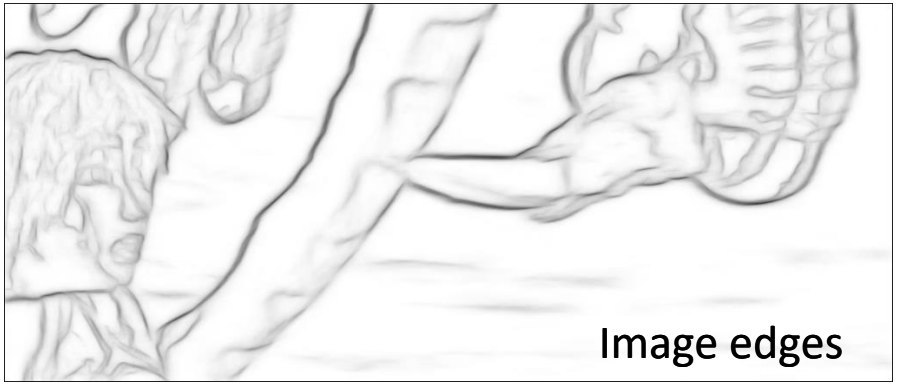


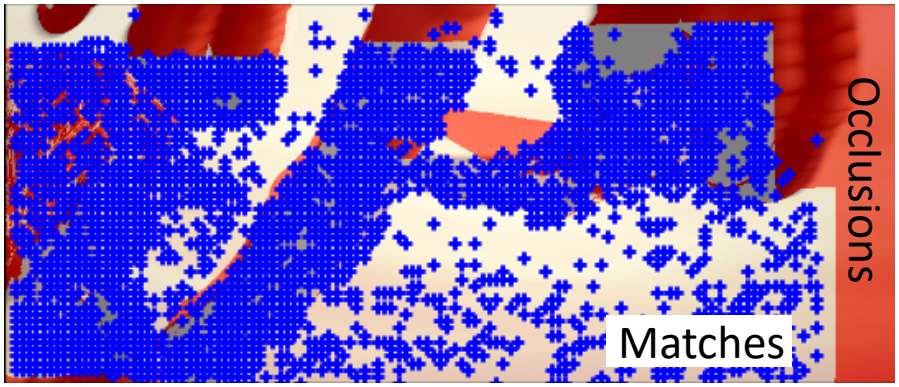


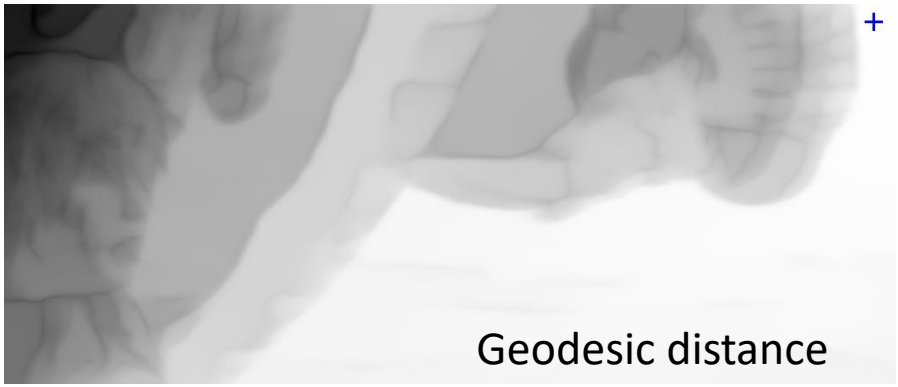


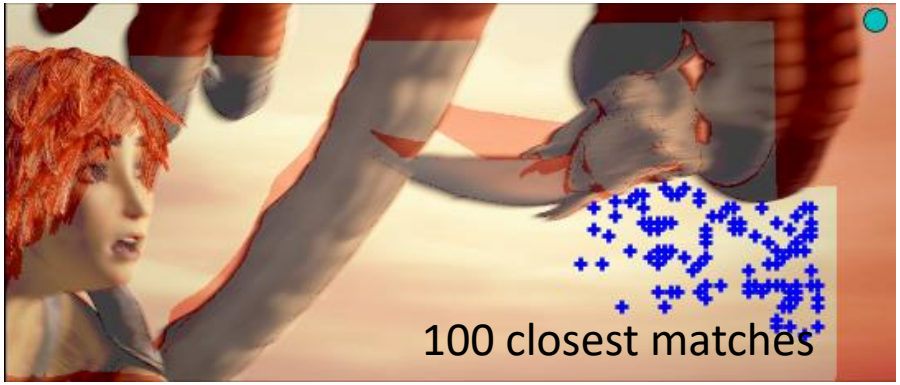


Geodesic Distance



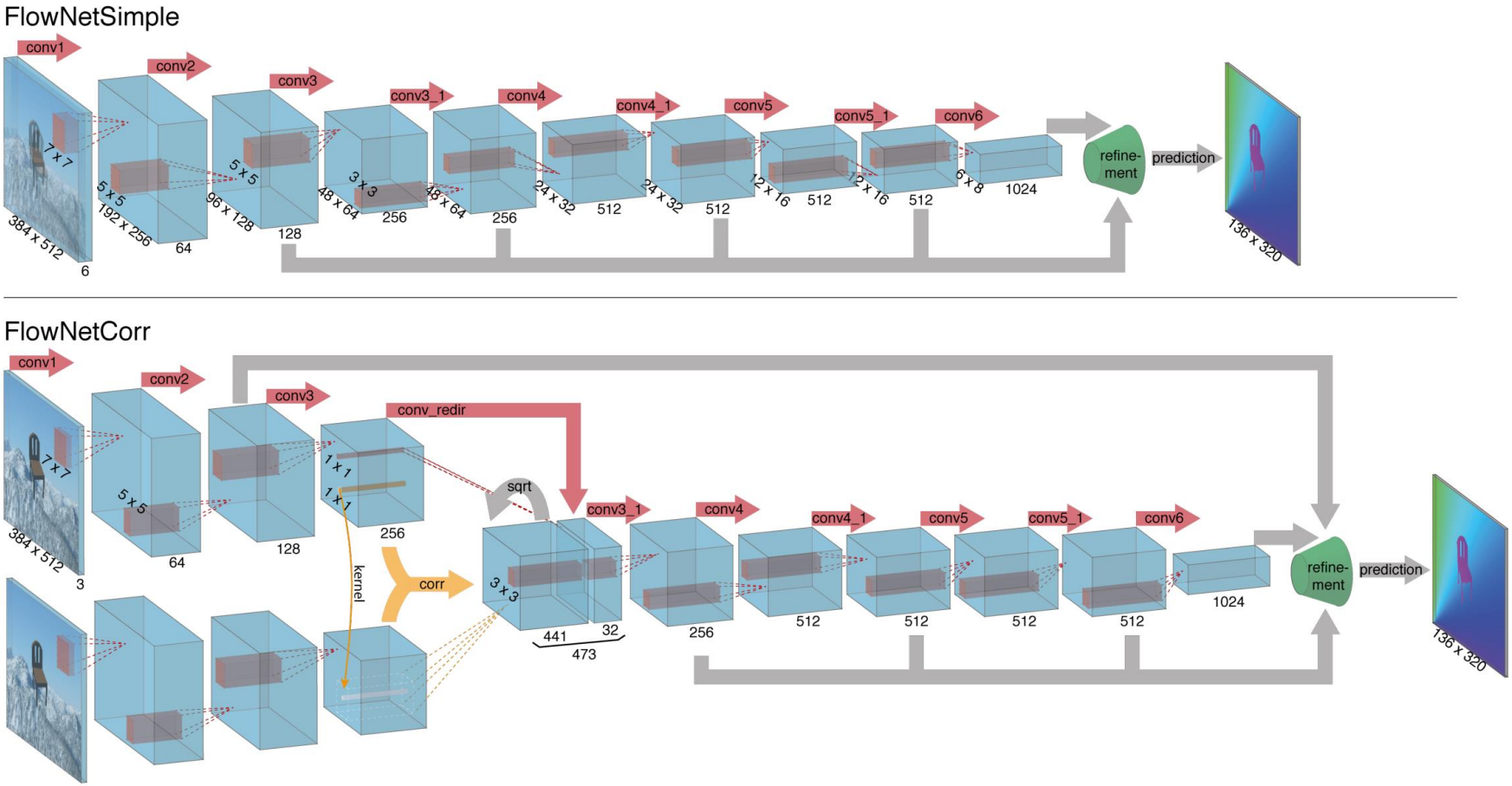






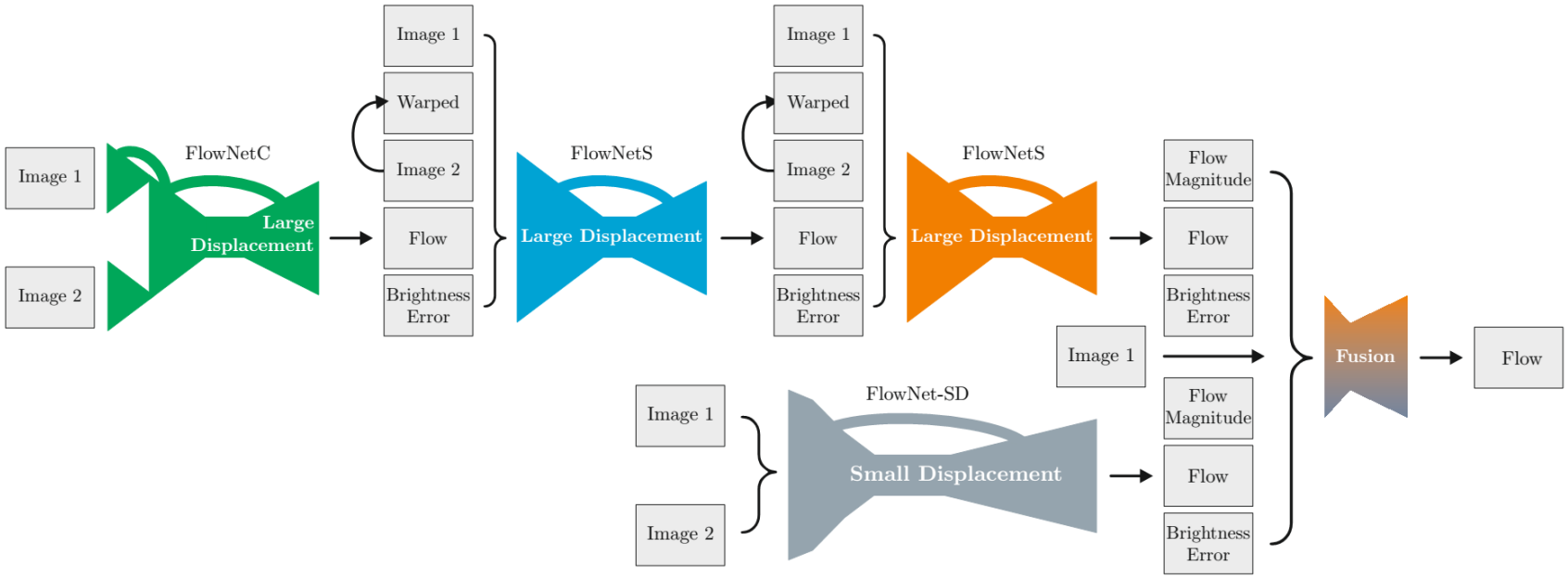
Experimental Results

FlowNet



Ref: A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, "FlowNet: Learning Optical Flow With Convolutional Networks," *CVPR2015*.

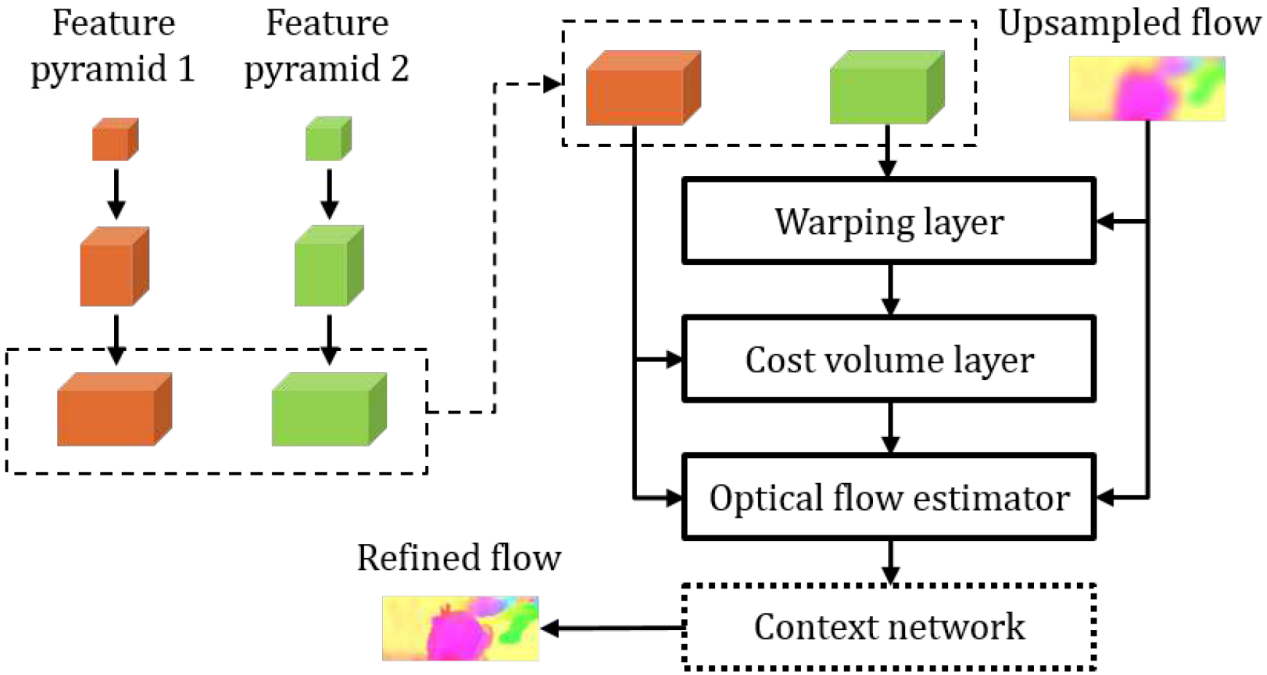
FlowNet 2.0



Ref: E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks," *CVPR2017*.

PWC-Net

- Pyramid of features
- Warping features
- Cost volume with small search range

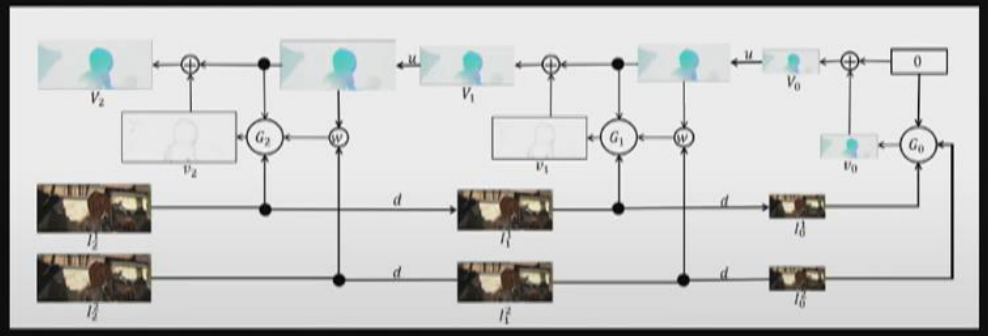


Ref: D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," *CVPR2018*.

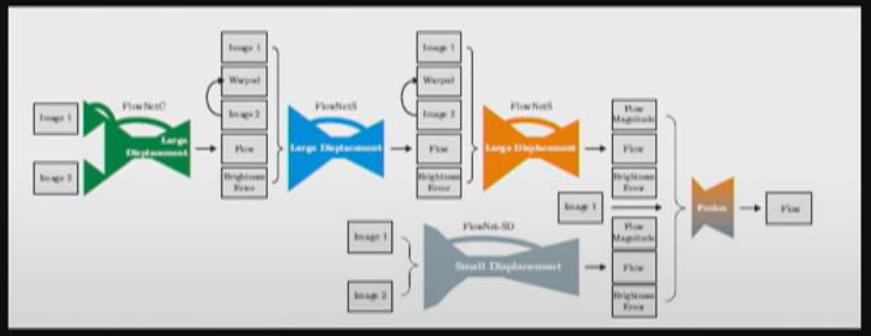
Comparison with Prior Work

Pyramid, Warping, and Cost volume

	SpyNet	FlowNet2	PWC-Net
Pyramid	Image	Feature (3-level)	Feature (6-level)
Warping	Image	Image	Feature
Cost volume	-	Once search range: 41x41	Every level search range: 9x9

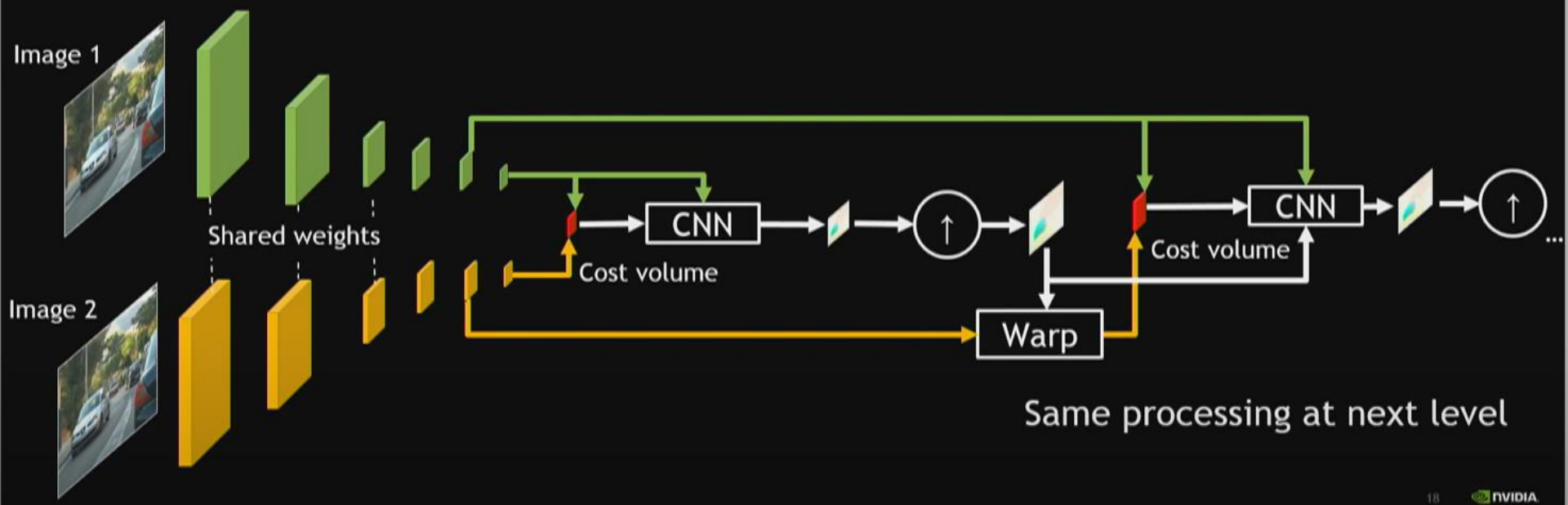


SpyNet: Ranjan & Black CVPR'17

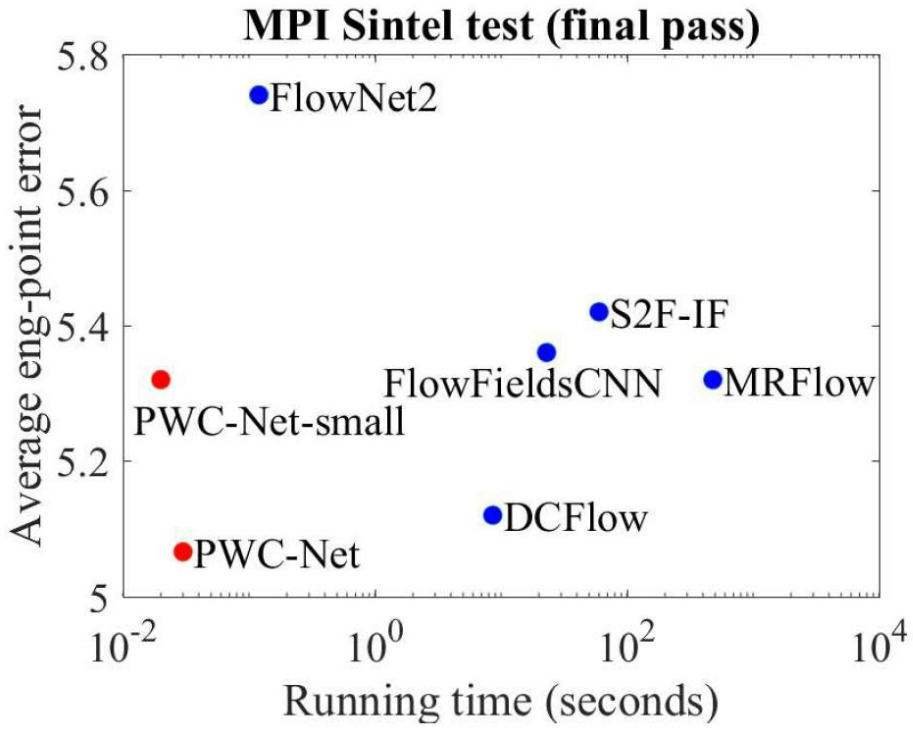


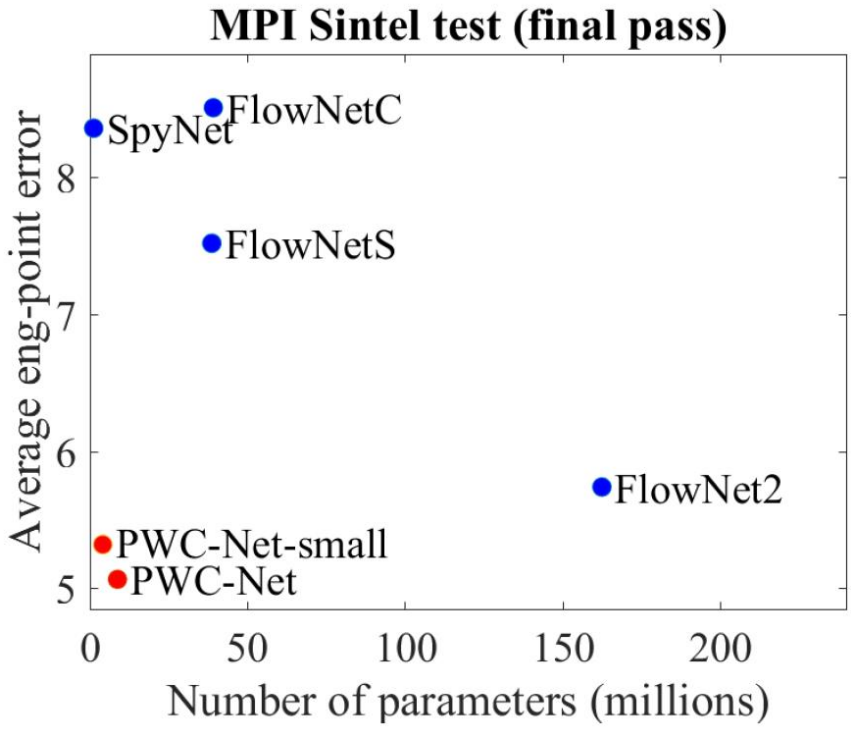
FlowNet2: Ilg *et al.* CVPR'17

Warping Features

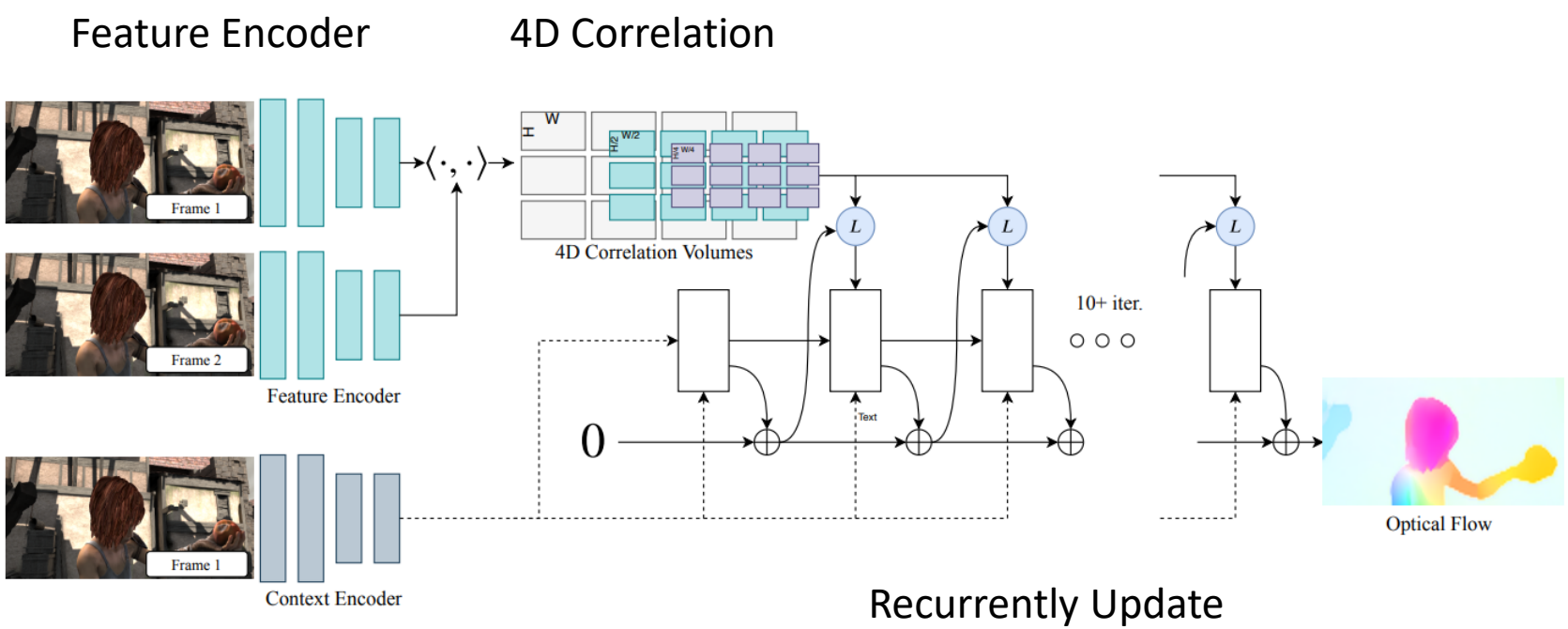


PWC-Net


MPI Sintel test (final pass)
Average eng-point error
Running time (seconds)
FlowNet2
S2F-IF
FlowFieldsCNN
MRFlow
PWC-Net-small
DCFlow
PWC-Net
5.8
5.6
5.4
5.2
5
10^{-2}
10^{0}
10^{2}
10^{4}

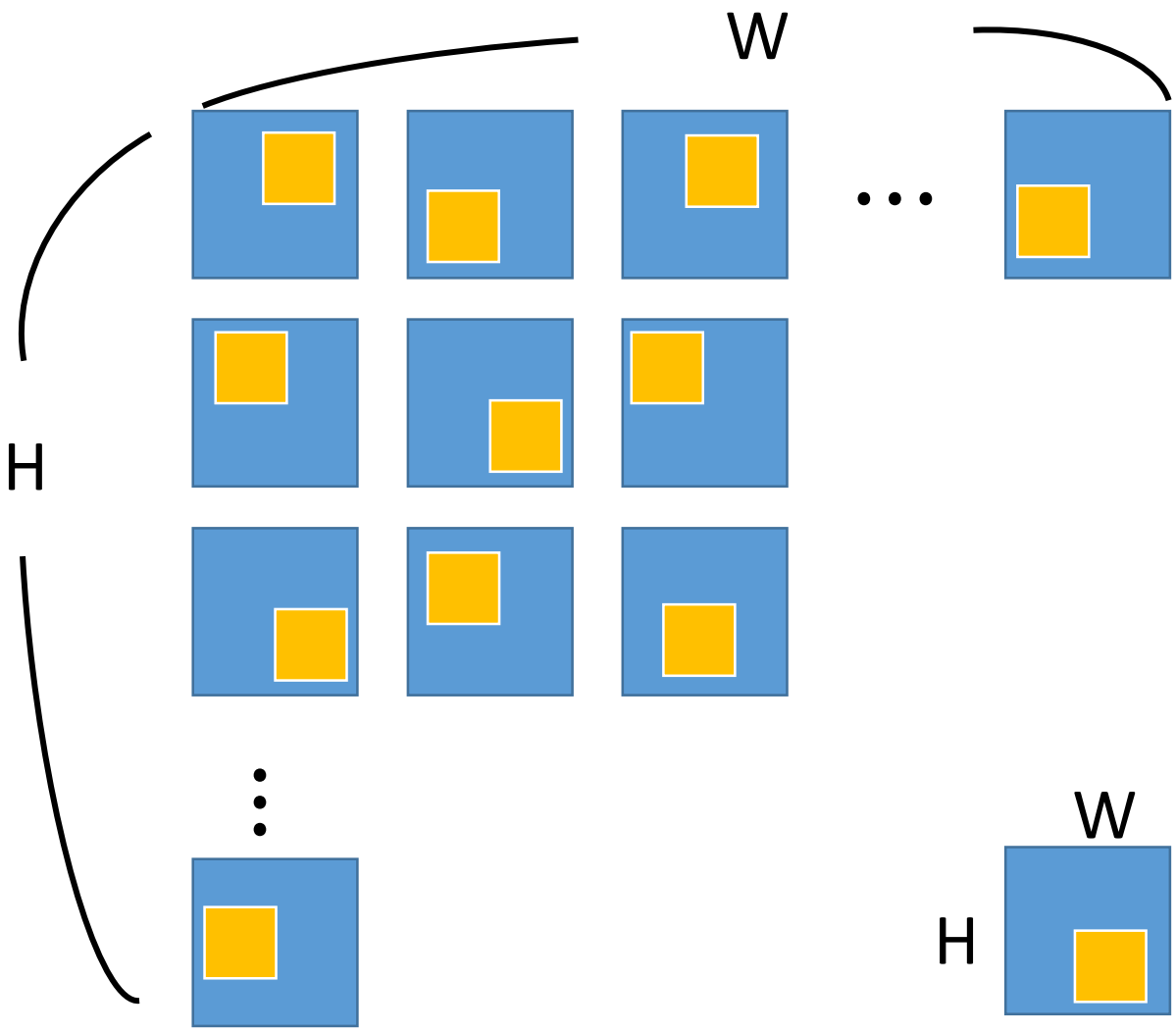

MPI Sintel test (final pass)
Average eng-point error
Number of parameters (millions)
SpyNet
FlowNetC
FlowNetS
FlowNet2
PWC-Net-small
PWC-Net
8
7
6
5
0
50
100
150
200

RAFT

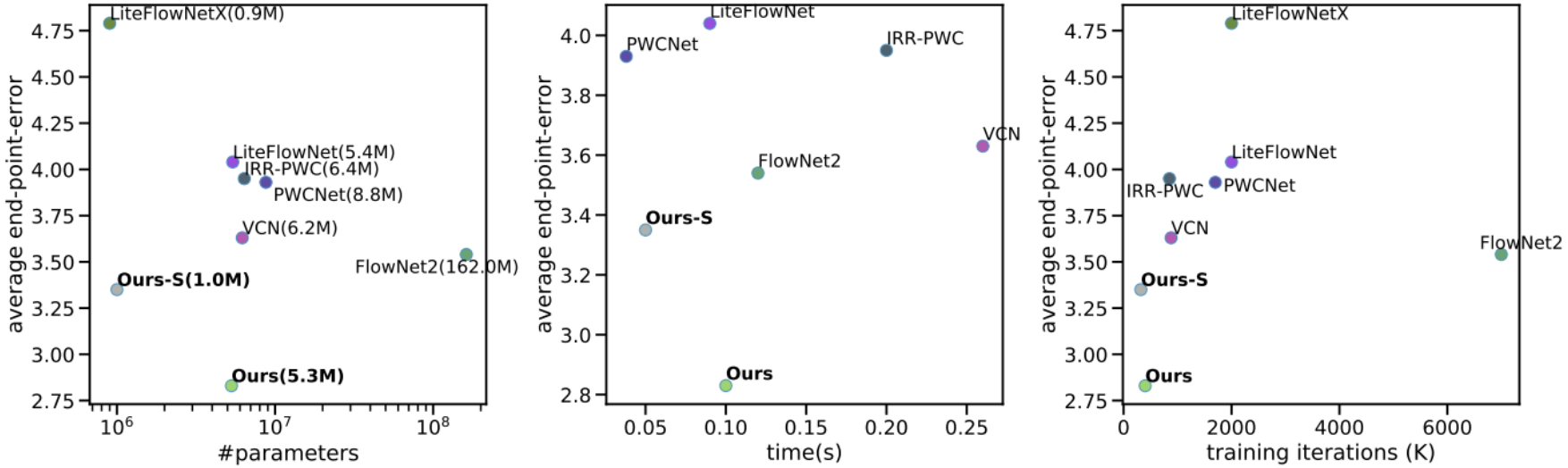


Ref: Z. Teed and J. Deng, "Recurrent All-Pairs Field Transforms for Optical Flow," in Proc. ECCV 2020.

4D Correlation



Experimental Results



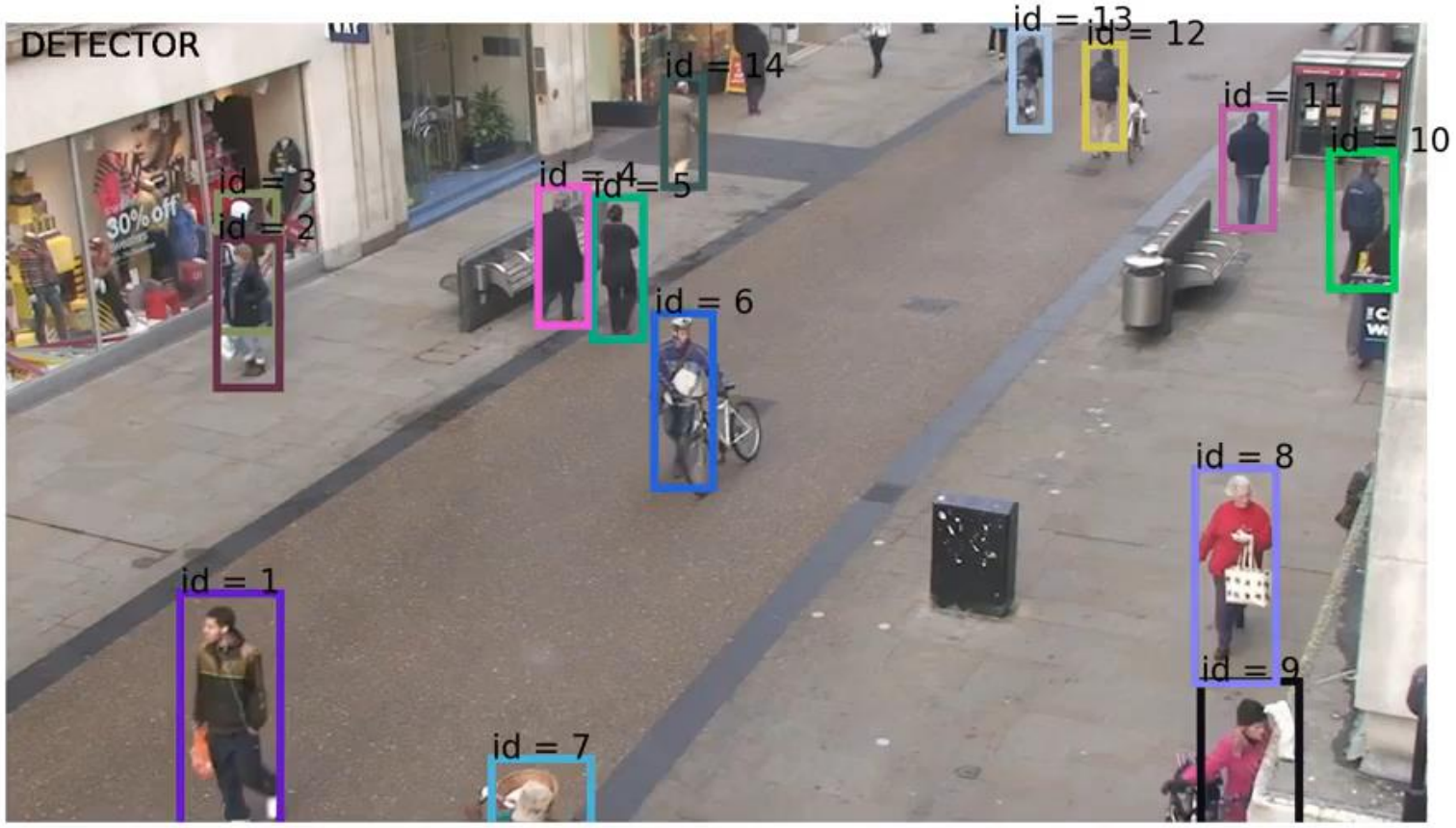
Ref: Z. Teed and J. Deng, "Recurrent All-Pairs Field Transforms for Optical Flow," in Proc. ECCV 2020.

	Method	Setting	Code	Fl-bg	Fl-fg	Fl-all	Density	Runtime	Environment	Compare
1	CamLiRAFT		code	**2.08 %**	7.37 %	**2.96 %**	100.00 %	1 s	GPU @ 2.5 Ghz (Python + C/C++)	
H. Liu, T. Lu, Y. Xu, J. Liu and L. Wang: Learning Optical Flow and Scene Flow with Bidirectional Camera-LiDAR Fusion. arXiv preprint arXiv:2303.12017 2023.										
2	CamLiFlow		code	2.31 %	7.04 %	3.10 %	100.00 %	1.2 s	GPU @ 2.5 Ghz (Python + C/C++)	
H. Liu, T. Lu, Y. Xu, J. Liu, W. Li and L. Chen: CamLiFlow: Bidirectional Camera-LiDAR Fusion for Joint Optical Flow and Scene Flow Estimation. CVPR 2022.										
3	TBD			2.90 %	5.05 %	3.26 %	100.00 %	TBD s	1 core @ 2.5 Ghz (C/C++)	
4	CamLiRAFT-NR		code	2.76 %	6.78 %	3.43 %	100.00 %	1 s	GPU @ 2.5 Ghz (Python + C/C++)	
H. Liu, T. Lu, Y. Xu, J. Liu and L. Wang: Learning Optical Flow and Scene Flow with Bidirectional Camera-LiDAR Fusion. arXiv preprint arXiv:2303.12017 2023.										
5	M-FUSE		code	2.66 %	7.47 %	3.46 %	100.00 %	1.3 s	GPU	
L. Mehl, A. Jahedi, J. Schmalfuss and A. Bruhn: M-FUSE: Multi-frame Fusion for Scene Flow Estimation. Proc. Winter Conference on Applications of Computer Vision (WACV) 2023.										
6	RigidMask+ISF		code	2.63 %	7.85 %	3.50 %	100.00 %	3.3 s	GPU @ 2.5 Ghz (Python)	
G. Yang and D. Ramanan: Learning to Segment Rigid Motions from Two Frames. CVPR 2021.										
7	ScaleRAFT3D			2.37 %	9.26 %	3.51 %	100.00 %	1 s	1 core @ 2.5 Ghz (C/C++)	
8	TPCV+RAFT3D			2.48 %	10.19 %	3.76 %	100.00 %	0.2 s	1 core @ 2.5 Ghz (C/C++)	
9	RAFT3D+mscv			2.48 %	10.23 %	3.77 %	100.00 %	0.2 s	1 core @ 2.5 Ghz (C/C++)	
10	CMAX+			3.39 %	6.21 %	3.86 %	100.00 %	1.5 s	GPU @ 2.5 Ghz (Python)	
11	RAFT-it+_RVC		code	3.62 %	5.33 %	3.90 %	100.00 %	0.14 s	1 core @ 2.5 Ghz (Python)	
D. Sun, C. Herrmann, F. Reda, M. Rubinstein, D. Fleet and W. Freeman: Disentangling Architecture and Training for Optical Flow. ECCV 2022.										
12	RAFT-OCTC			3.72 %	5.39 %	4.00 %	100.00 %	0.2 s	GPU @ 2.5 Ghz (Python)	
J. Jeong, J. Lin, F. Porikli and N. Kwak: Imposing Consistency for Optical Flow Estimation (Qualcomm AI Research). CVPR 2022.										
13	RCA-Flow			3.67 %	6.25 %	4.10 %	100.00 %	0.16 s	1 core @ 2.5 Ghz (Python)	
14	SF2SE3		code	3.17 %	8.79 %	4.11 %	100.00 %	2.7 s	GPU @ >3.5 Ghz (Python)	
L. Sommer, P. Schröppel and T. Brox: SF2SE3: Clustering Scene Flow into SE (3)-Motions via Proposal and Selection. DAGM German Conference on Pattern Recognition 2022.										
15	RAFT-CF-CE-PL3			3.80 %	5.65 %	4.11 %	100.00 %	0.05 s	GPU @ 2.5 Ghz (Python)	
16	RAFT-S-AF		code	3.86 %	5.38 %	4.12 %	100.00 %	1 s	1 core @ 2.5 Ghz (C/C++)	
17	MS_RAFT+_corr_RVC		code	3.83 %	5.71 %	4.15 %	100.00 %	0.65 s	GPU @ 2.5 Ghz (Python + C/C++)	
A. Jahedi, M. Luz, L. Mehl, M. Rivinius and A. Bruhn: High Resolution Multi-Scale RAFT. Robust Vision Challenge 2022, arXiv preprint arXiv:2210.16900 2022. A. Jahedi, L. Mehl, M. Rivinius and A. Bruhn: Multi-Scale Raft: Combining Hierarchical Concepts for Learning-Based Optical Flow Estimation. IEEE International Conference on Image Processing (ICIP) 2022.										
18	MS_RAFT+_RVC			3.89 %	5.67 %	4.19 %	100.00 %	0.65 s	GPU @ 2.5 Ghz (Python + C/C++)	
19	DIP		code	3.86 %	5.96 %	4.21 %	100.00 %	0.15 s	1 core @ 2.5 Ghz (Python)	
Z. Zheng, N. Nie, Z. Ling, P. Xiong, J. Liu, H. Wang and J. Li: DIP: Deep Inverse Patchmatch for High- Resolution Optical Flow. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022.										
20	APCAFlow			3.86 %	6.14 %	4.24 %	100.00 %	0.2 s	1 core @ 2.5 Ghz (Python)	

Outline

- Tracking
  - Introduction
  - Single camera tracking
    - Feature tracker, KLT tracker
    - Region area tracker
      - Mean-shift
      - Tracking with adaptive filters
      - On-line learning
  - Multi-camera tracking

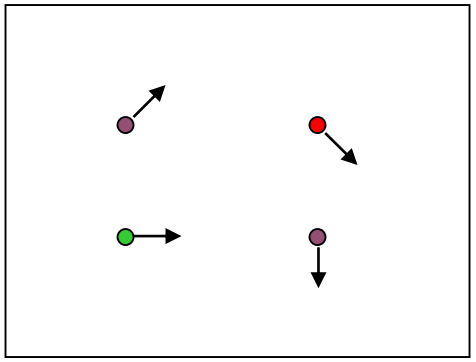
Introduction to Tracking



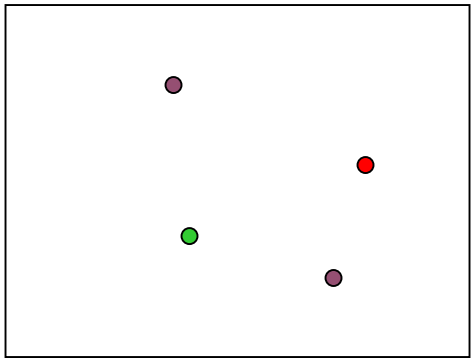
Introduction to Tracking

- Input: target
- Objective: Estimate target state over time
- State:
  - Position
  - Apperance
  - Shape
  - Velocity
  - …
- It is about data association, similarity measurement, and correlation

Feature Tracking



$I(x,y,t)$

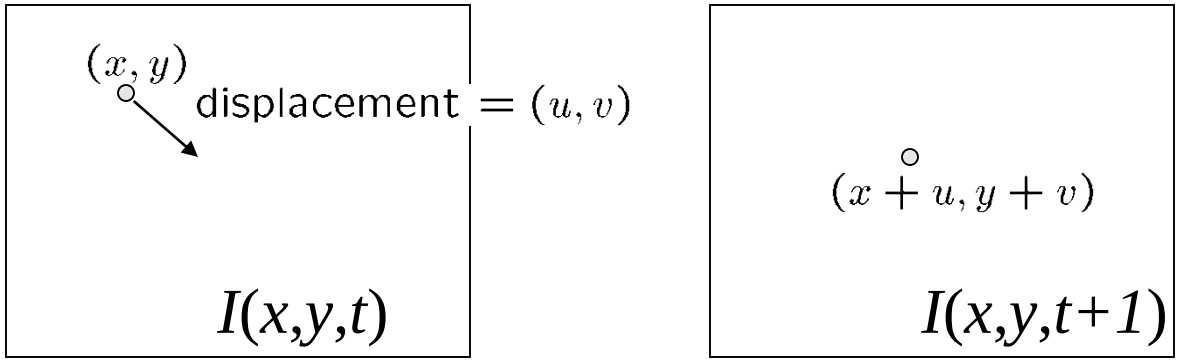


$I(x,y,t+1)$

- Given two subsequent frames, estimate the point translation

- Key assumptions of Lucas-Kanade Tracker
  - **Brightness constancy:** projection of the same point looks the same in every frame
  - **Small motion:** points do not move very far
  - **Spatial coherence:** points move like their neighbors

The brightness constancy constraint



- Brightness Constancy Equation:

$$I(x,y,t) = I(x+u, y+v, t+1)$$

Take Taylor expansion of $I(x+u, y+v, t+1)$ at (x,y,t) to linearize the right side:

Image derivative along x Difference over frames

$$I(x+u, y+v, t+1) \approx I(x,y,t) + I_x \cdot u + I_y \cdot v + I_t$$

$$I(x+u, y+v, t+1) - I(x,y,t) = I_x \cdot u + I_y \cdot v + I_t$$

So: $$I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \rightarrow \nabla \mathrm{I} \cdot \left[\mathrm{u}\ \mathrm{v}\right]^{\mathrm{T}} + \mathrm{I_t} = 0$$

The Brightness Constancy Constraint

Can we use this equation to recover image motion (u,v) at each pixel?

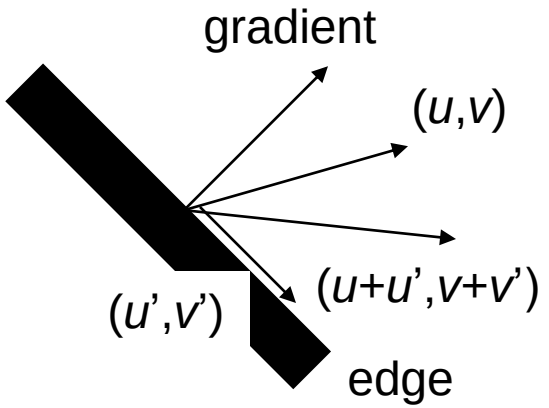
$$\nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$$

- How many equations and unknowns per pixel?
  - One equation (this is a scalar equation!), two unknowns (u,v)

The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If (*u*, *v*) satisfies the equation, so does (*u+u'*, *v+v'*) if

$$\nabla I \cdot \begin{bmatrix} u' & v' \end{bmatrix}^T = 0$$



Solving the ambiguity…

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

- How to get more equations for a pixel?
- **Spatial coherence constraint**
  - Assume the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u\ v]$$

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}$$

Solving the ambiguity…

- Least squares problem:

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Matching patches across images

- Overconstrained linear system

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix} \qquad \underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Least squares solution for *d* given by $(A^T A)\ d = A^T b$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $\qquad\qquad$ $A^T b$

The summations are over all pixels in the K x K window

Conditions for solvability

Optimal (u, v) satisfies Lucas-Kanade equation

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $\qquad$ $A^T b$

When is this solvable? I.e., what are good points to track?

- $\mathbf{A^T A}$ should be invertible
- $\mathbf{A^T A}$ should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of $\mathbf{A^T A}$ should not be too small
- $\mathbf{A^T A}$ should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Does this remind you of anything?

Criteria for Harris corner detector

Low-texture region

$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude

– small λ_1, small λ_2

Edge

$$\sum \nabla I (\nabla I)^T$$

– gradients very large or very small

– large λ_1, small λ_2

High-texture region

$$\sum \nabla I (\nabla I)^T$$

– gradients are different, large magnitudes

– large λ_1, large λ_2

Dealing with Larger Movements: Iterative Refinement

Original (x,y) position

1. Initialize (x',y') = (x,y)
2. Compute (u,v) by

$I_t = I(x', y', t+1) - I(x, y, t)$

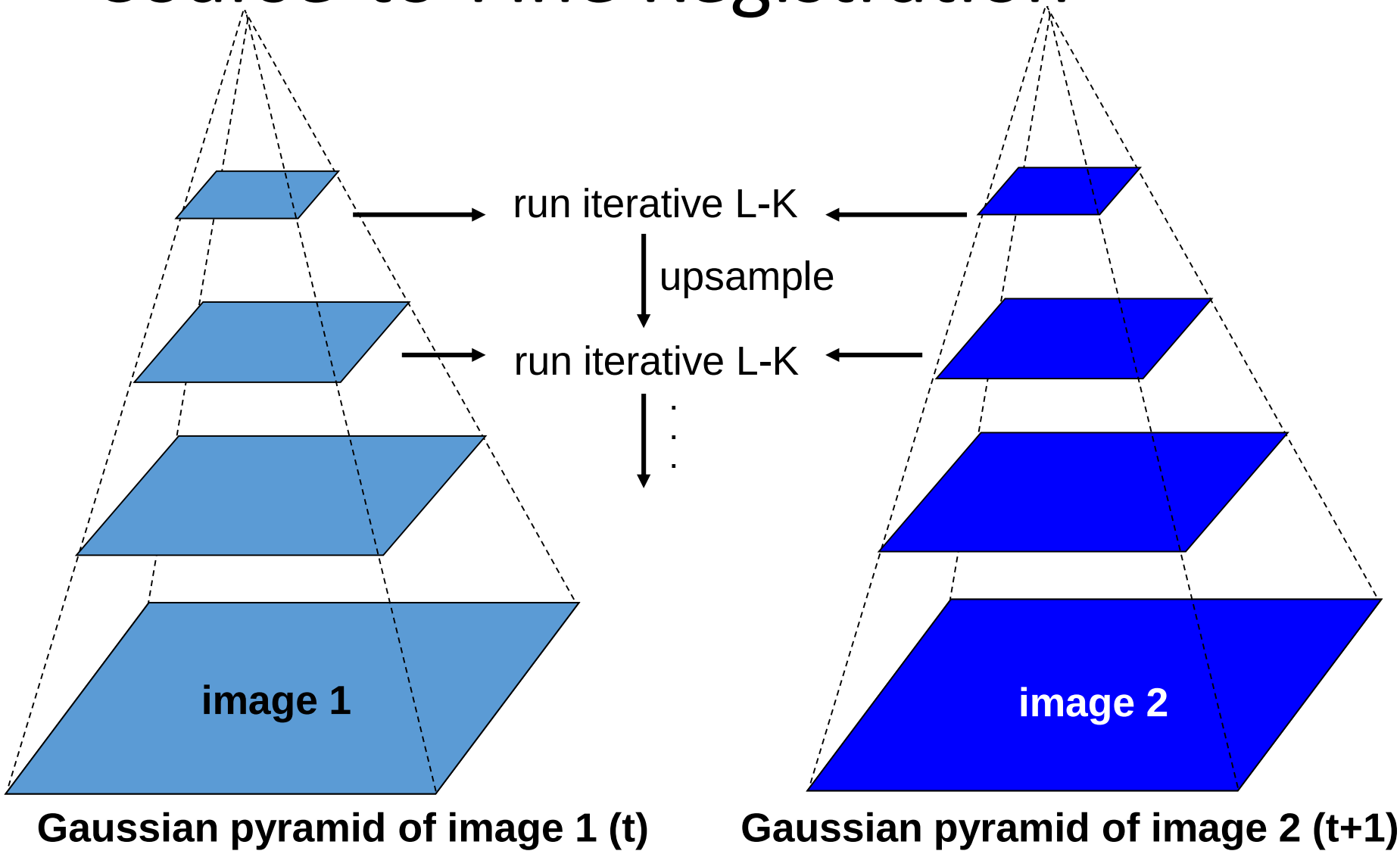
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

2nd moment matrix for feature patch in first image

displacement

3. Shift window by (u, v): `x'=x'+u;  y'=y'+v;`
4. Recalculate I_t
5. Repeat steps 2-4 until small change
   - Use interpolation for subpixel values

Dealing with Larger Movements: Coarse-to-Fine Registration



Shi-Tomasi Feature Tracker

- Find good features using eigenvalues of second-moment matrix (e.g., Harris detector or threshold on the smallest eigenvalue)
  - Key idea: "good" features to track are the ones whose motion can be estimated reliably

- Track from frame to frame with Lucas-Kanade
  - This amounts to assuming a translation model for frame-to-frame feature movement

- Check consistency of tracks by *affine* registration to the first observed instance of the feature
  - Affine model is more accurate for larger displacements
  - Comparing to the first frame helps to minimize drift

Ref: J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Tracking example

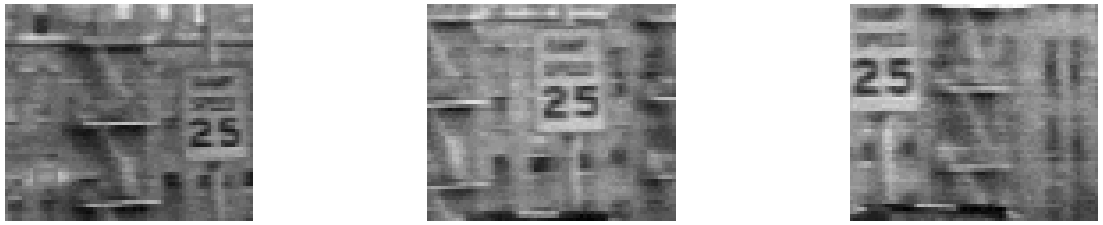


Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

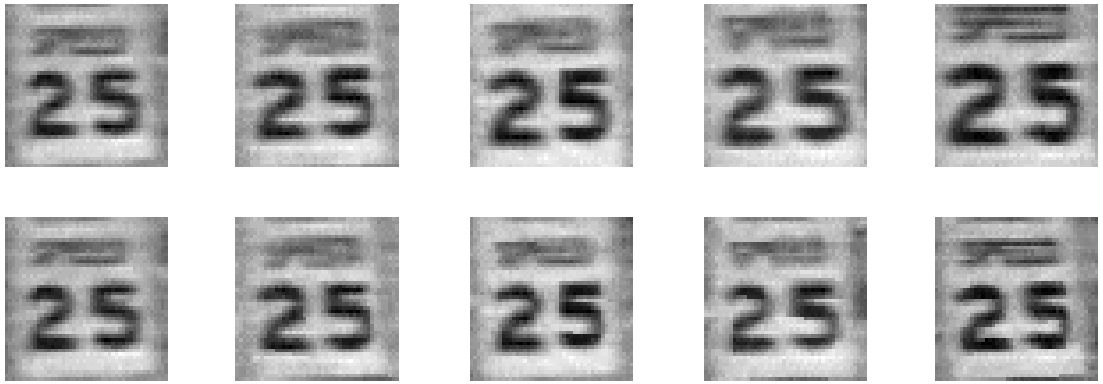


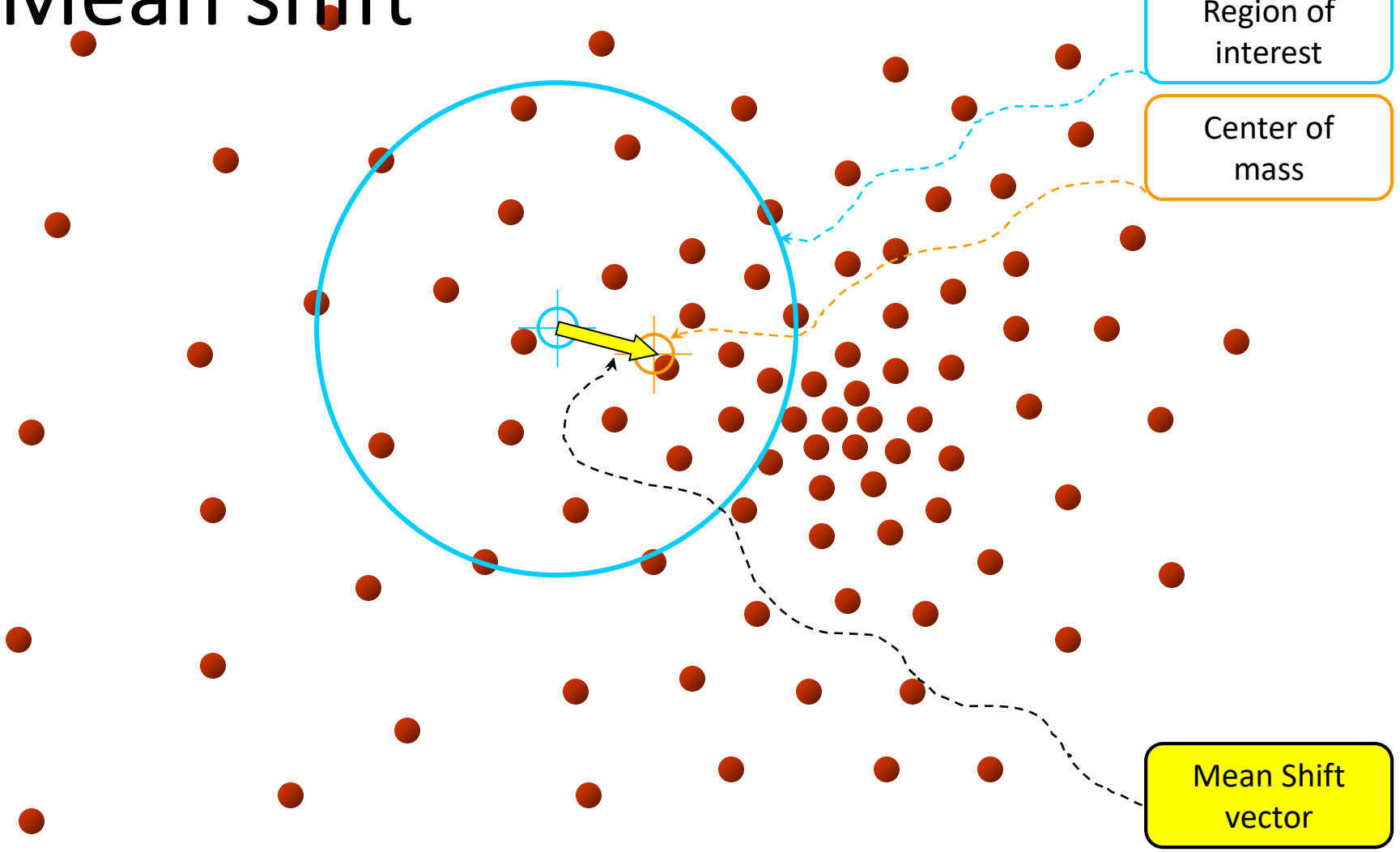
Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

Ref: J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Summary of KLT Tracking (Kanade-Lucas-Tomasi Feature Tracker)

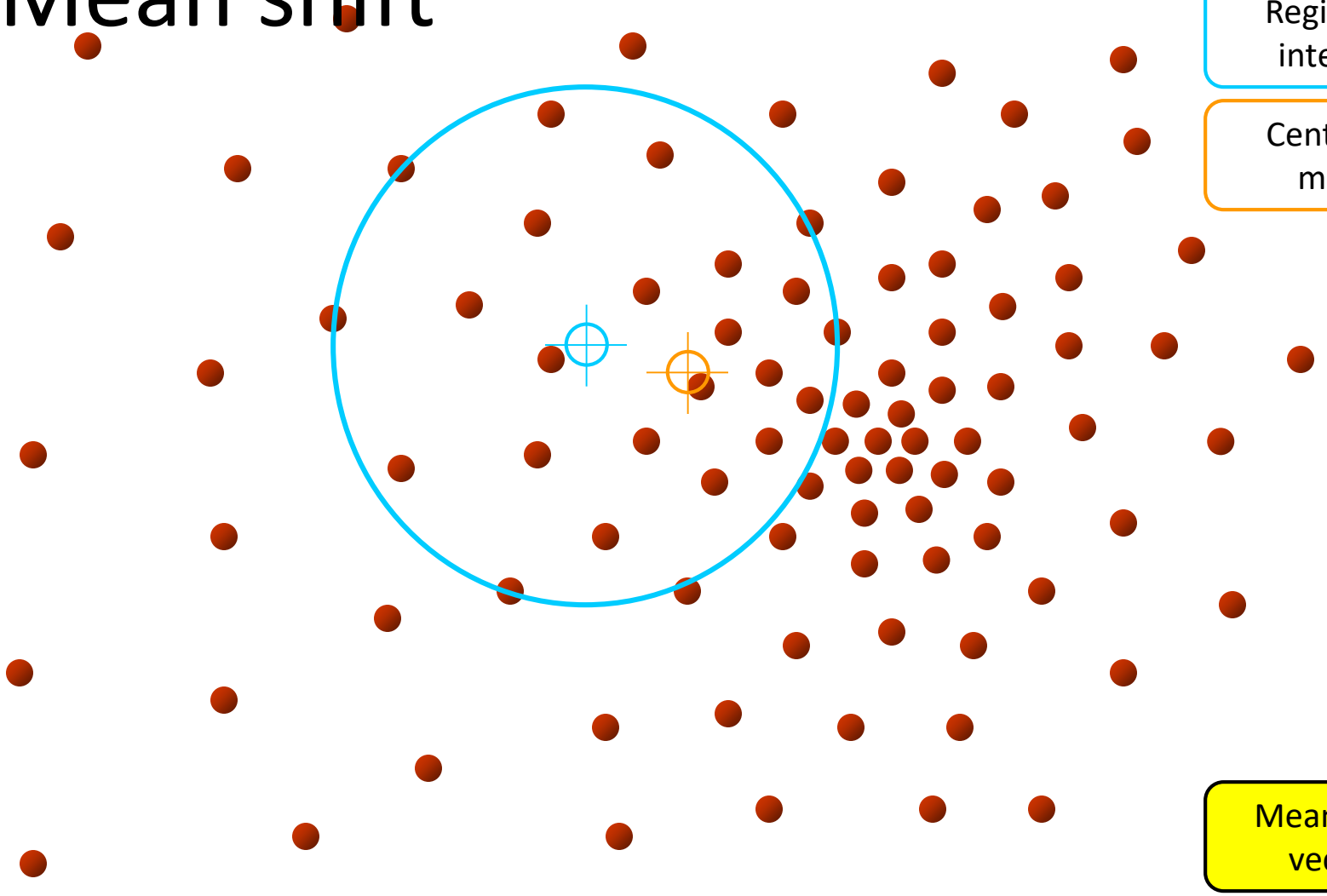
- Find a good point to track (harris corner)
- Use intensity second moment matrix and difference across frames to find displacement
- Iterate and use coarse-to-fine search to deal with larger movements
- When creating long tracks, check appearance of registered patch against appearance of initial patch to find points that have drifted

Mean shift



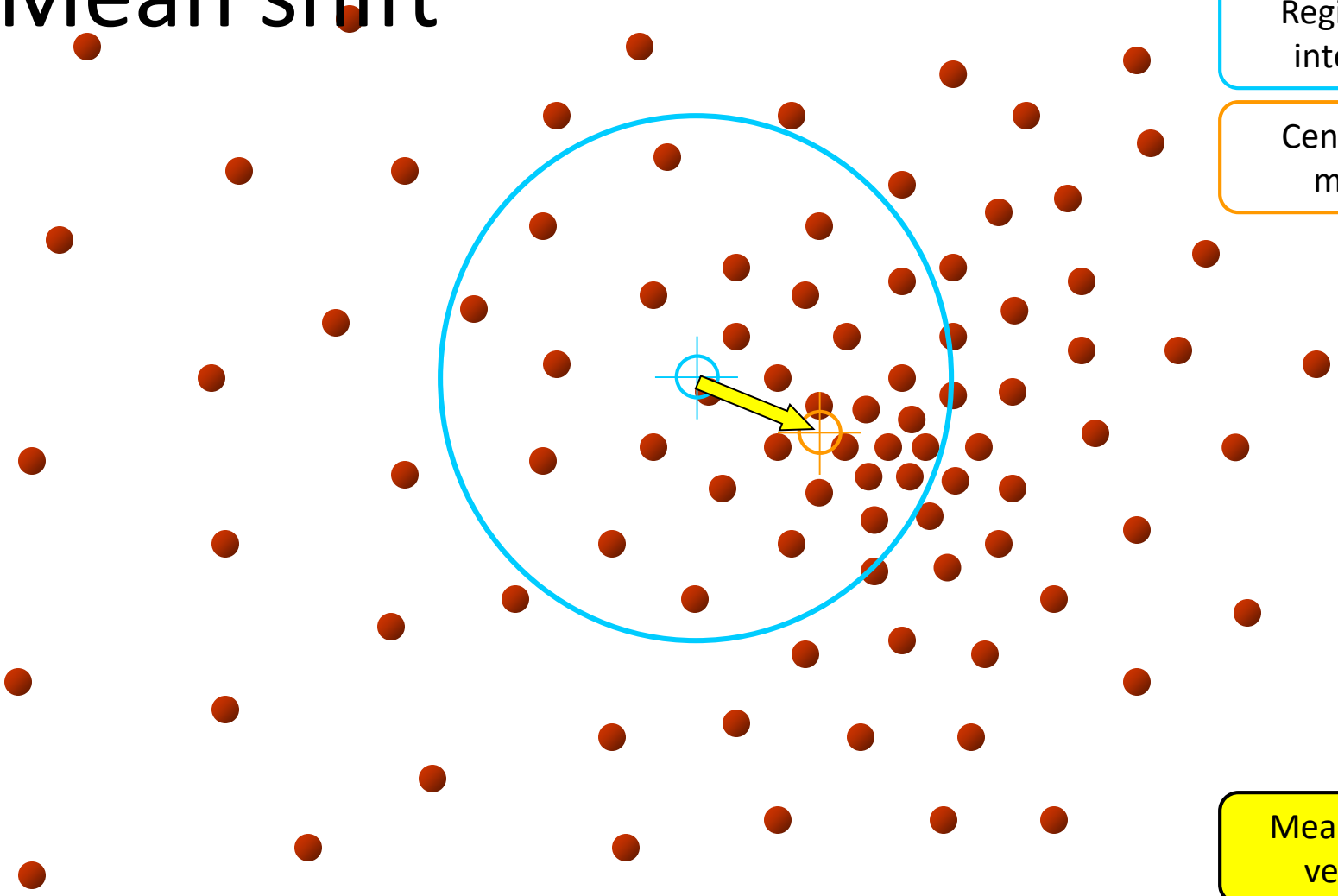
Slide by Y. Ukrainitz & B. Sarel

Mean shift



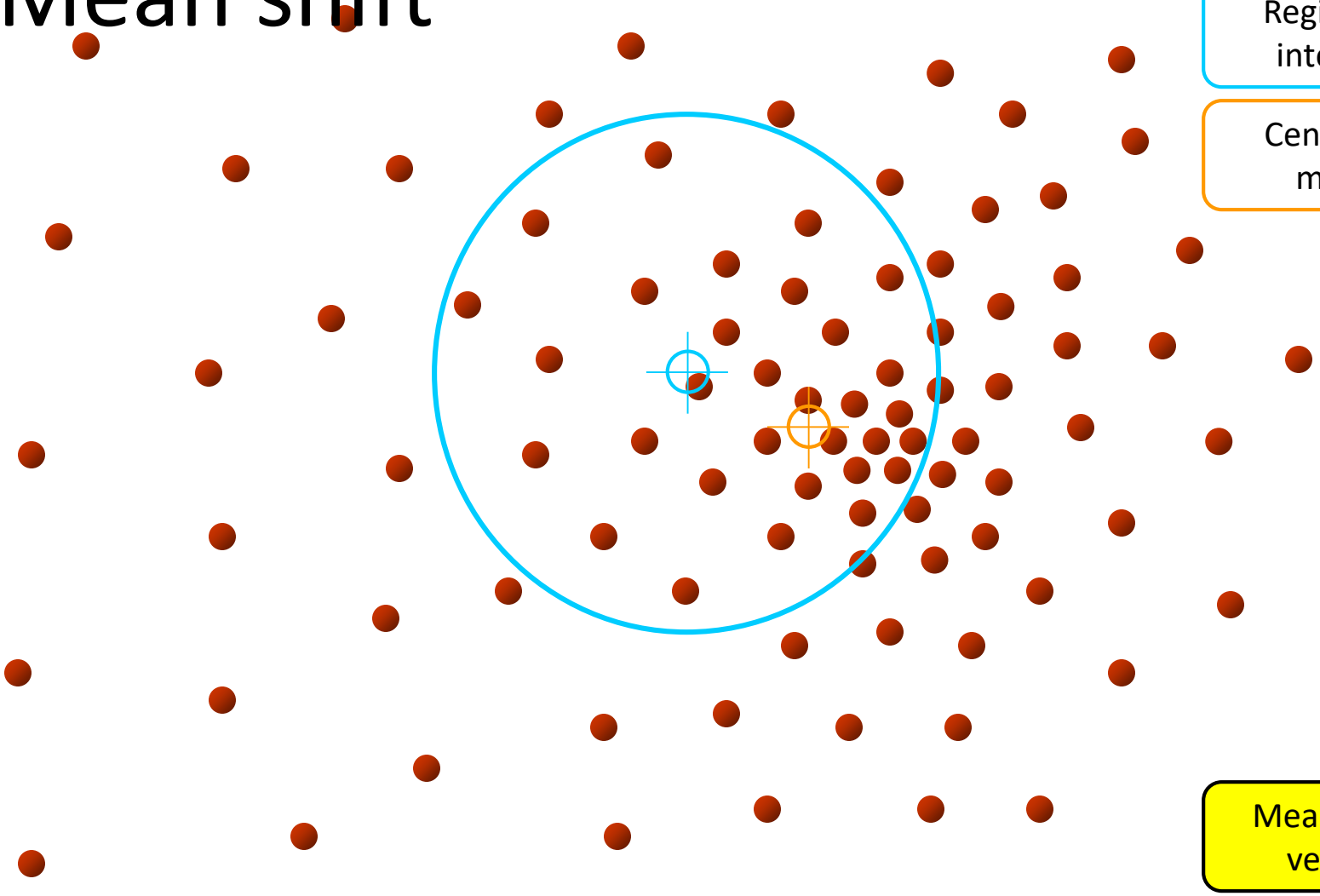
Slide by Y. Ukrainitz & B. Sarel

Mean shift



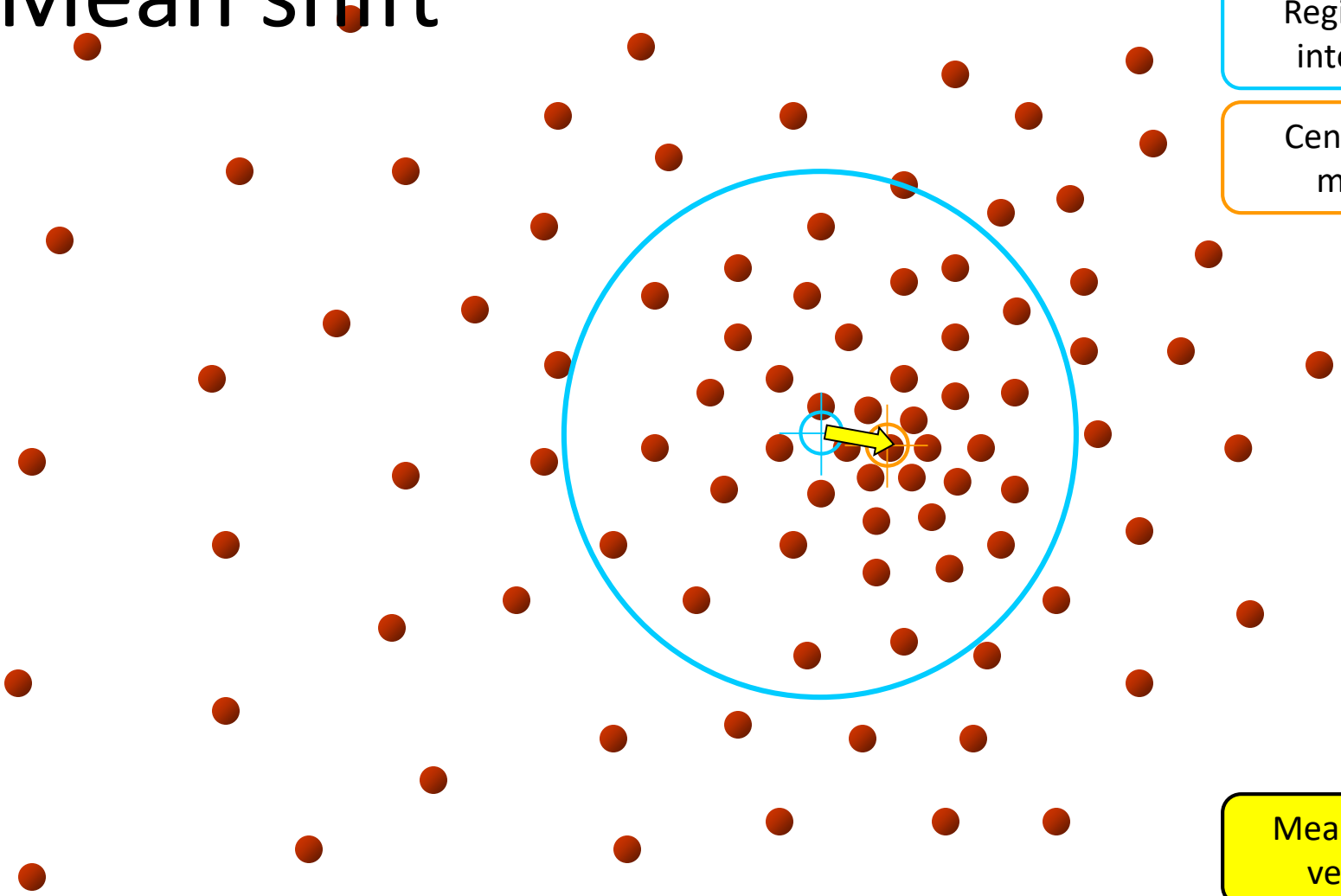
Slide by Y. Ukrainitz & B. Sarel

Mean shift



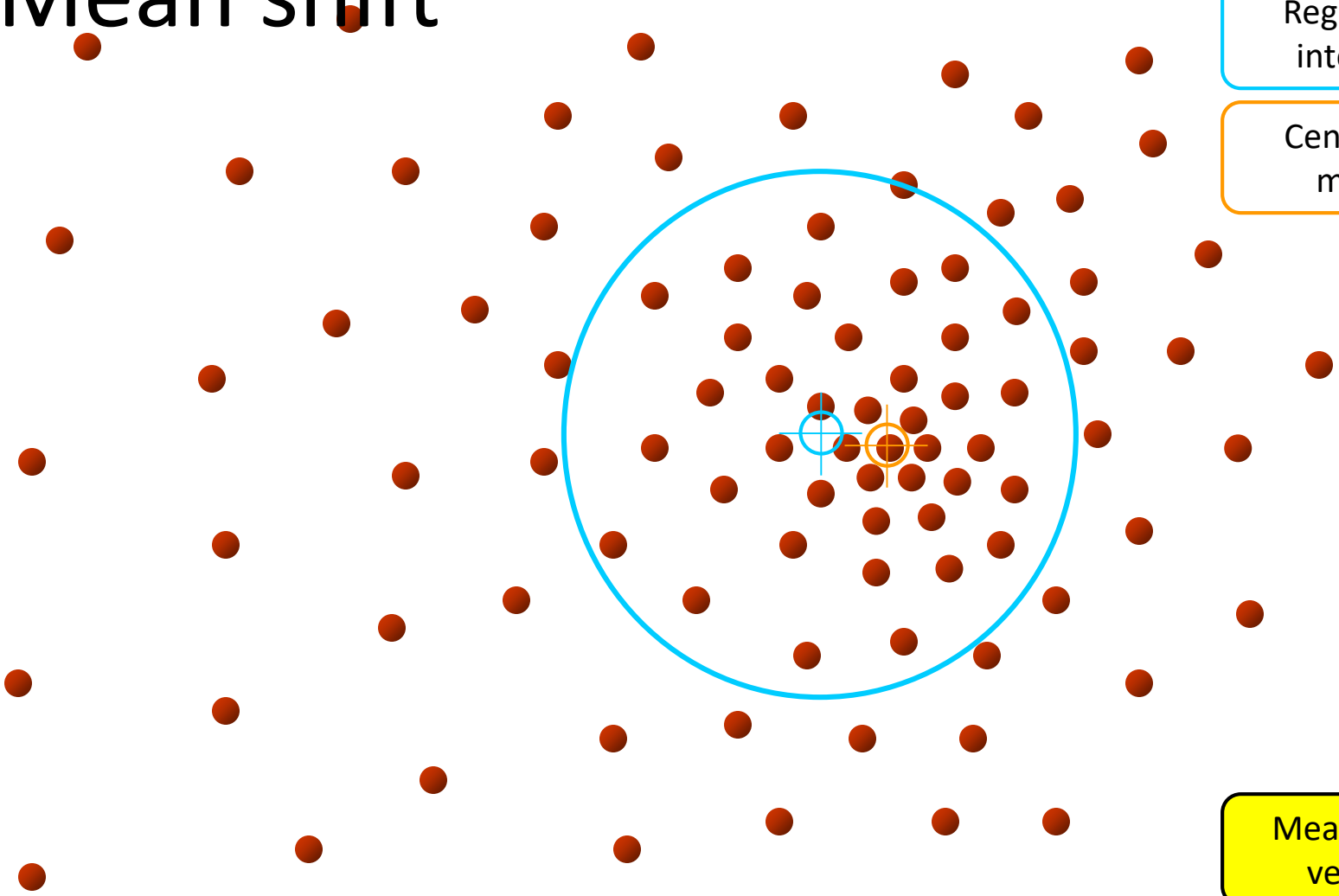
Slide by Y. Ukrainitz & B. Sarel

Mean shift



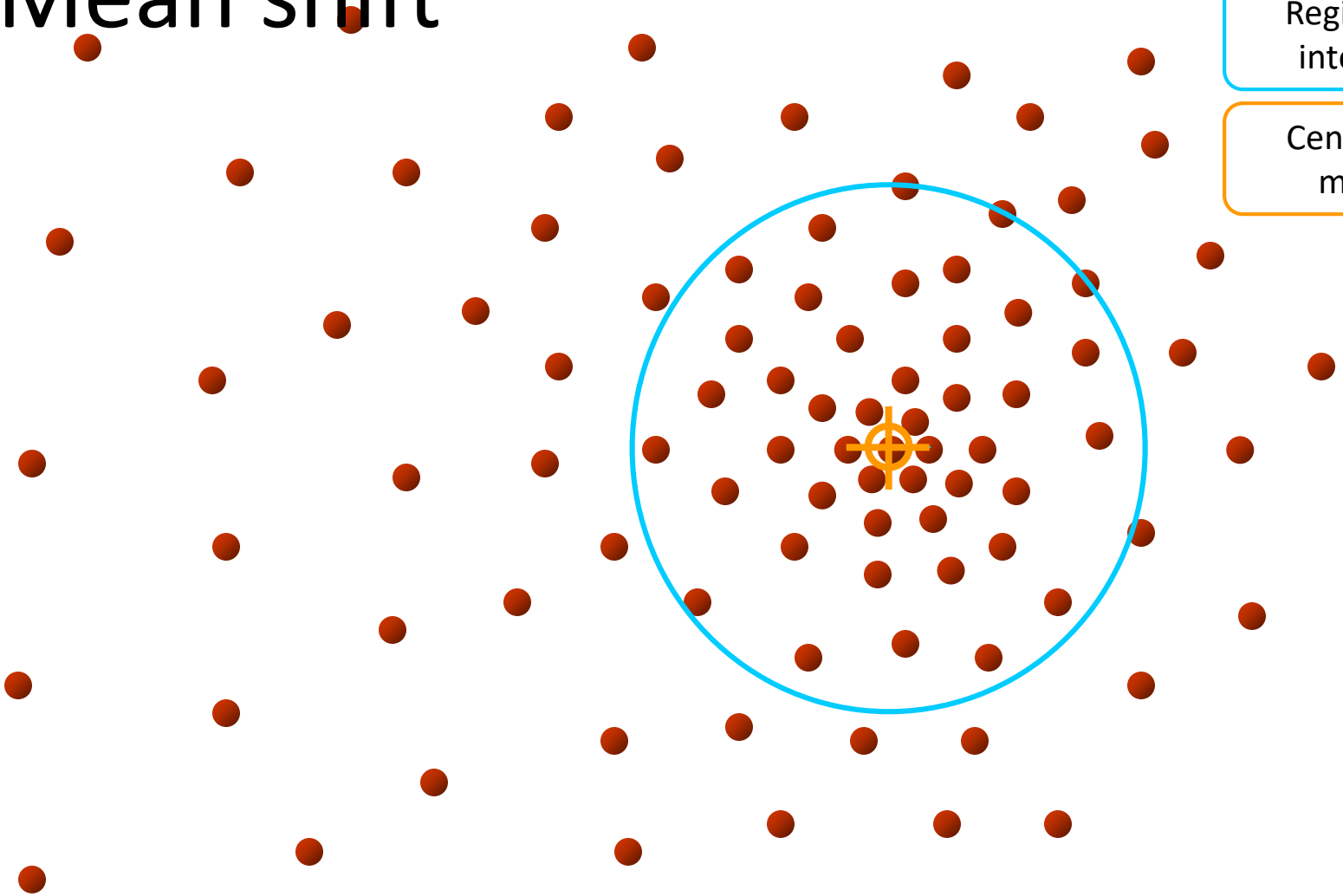
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

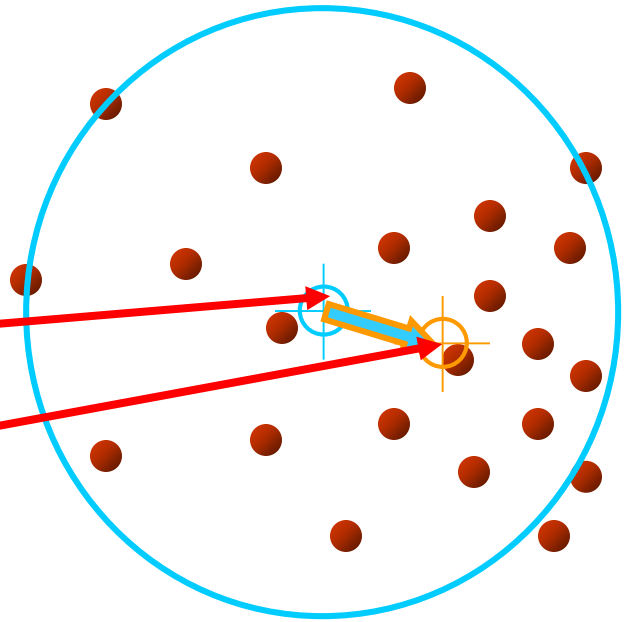
Computing the Mean Shift

Simple Mean Shift procedure:

- Compute mean shift vector

- Translate the Kernel window by **m(x)**

$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h} \right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h} \right)} - \mathbf{x} \right]$$



Slide by Y. Ukrainitz & B. Sarel

Kalman Filter

- Kalman filters are modeled on a Markov chain built on linear operators perturbed by Gaussian noises.
- At time *k*, each target has state x_k

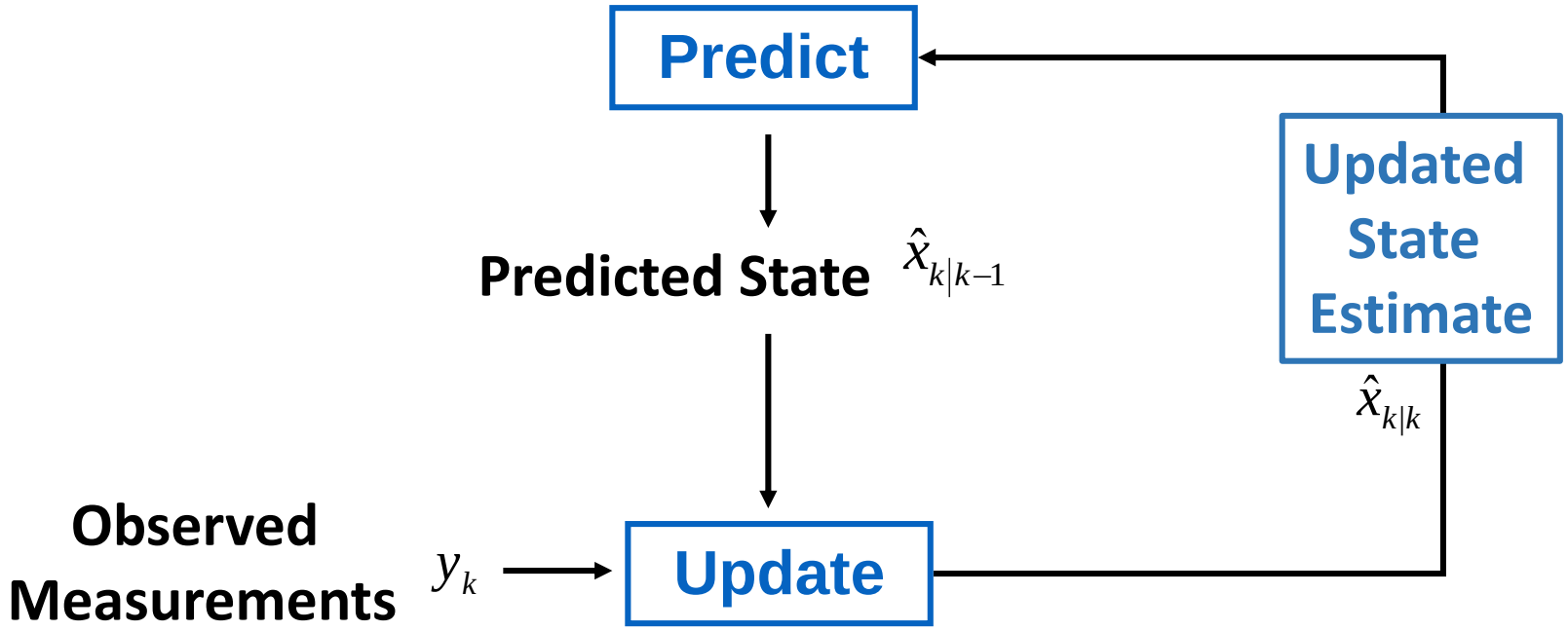
$$x_k = F_k x_{k-1} + w_k \quad \text{, where} \quad w_k \sim \mathrm{N}(0, Q_k)$$

and observation (measurement) y_k

$$y_k = H_k x_k + v_k \quad \text{, where} \quad v_k \sim \mathrm{N}(0, R_k)$$

Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME - Journal of Basic Engineering* Vol. 82: pp. 35-45, 1960.

Kalman Filter Phases



$$x_k = y_k = [u_k \; v_k \; \dot{u}_k \; \dot{v}_k]^T$$

$$F_k = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Kalman Filter Phases

Predict Phase

- Predicted State

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1}$$

- Predicted Estimate Covariance

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^{\ T} + Q_k$$

Update Phase

- Updated State Estimate

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$

- Updated Estimate Covariance

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

- Kalman Gain

$$K_k = P_{k|k-1} H_k^{\ T} S_k^{\ -1}$$

- Innovation Covariance

$$S_k = H_k P_{k|k-1} H_k^{\ T} + R_k$$

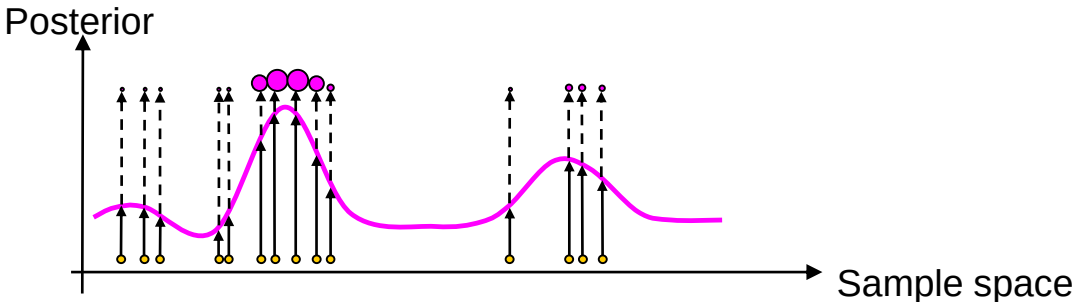
- Innovation (Measurement) Residual

$$\tilde{y}_k = y_k - H_k \hat{x}_{k|k-1}$$

Particle filtering

- Many variations, one general concept:

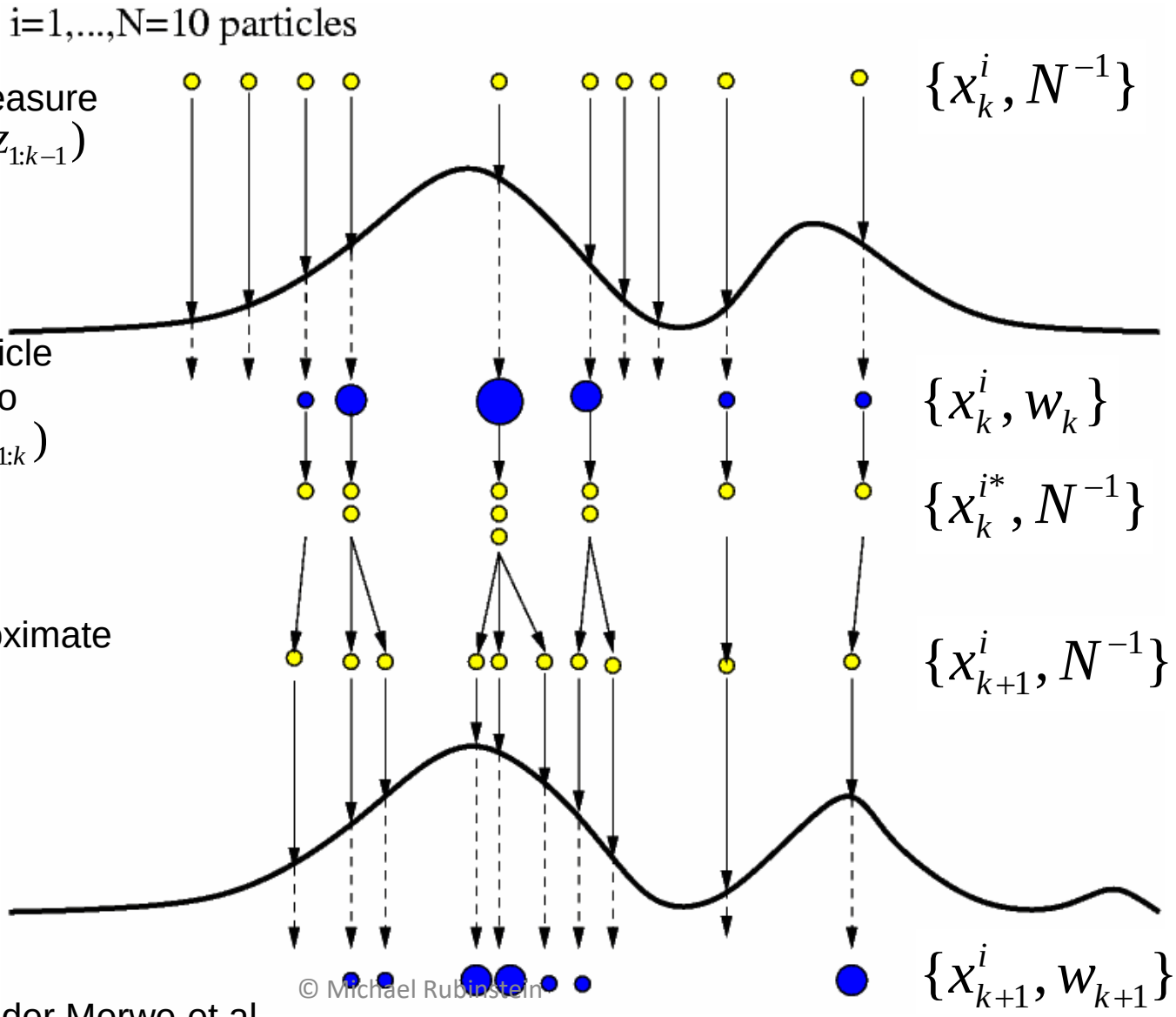
Represent the posterior pdf by a set of randomly chosen weighted samples (particles)



- Randomly Chosen = Monte Carlo (MC)
- As the number of samples become very large – the characterization becomes an equivalent representation of the true pdf

Slide credit: Michael Rubinstein

Generic PF



Uniformly weighted measure
Approximates $p(x_k \mid z_{1:k-1})$

Compute for each particle
its importance weight to
Approximate $p(x_k \mid z_{1:k})$

(Resample if needed)

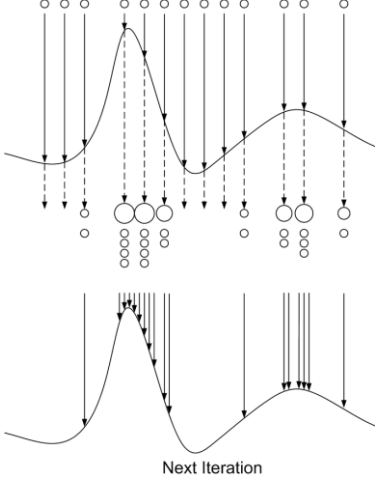
Project ahead to approximate
$p(x_{k+1} \mid z_{1:k})$

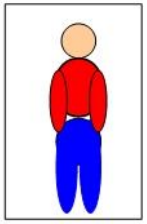
$p(x_{k+1} \mid z_{1:k+1})$

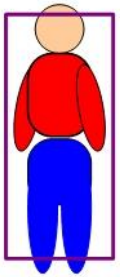
Van der Merwe et al.

Particle Filter

- Try to approximate $p(\mathbf{X} \mid \underline{Z}_t)$
- Sampling → evaluation → resampling



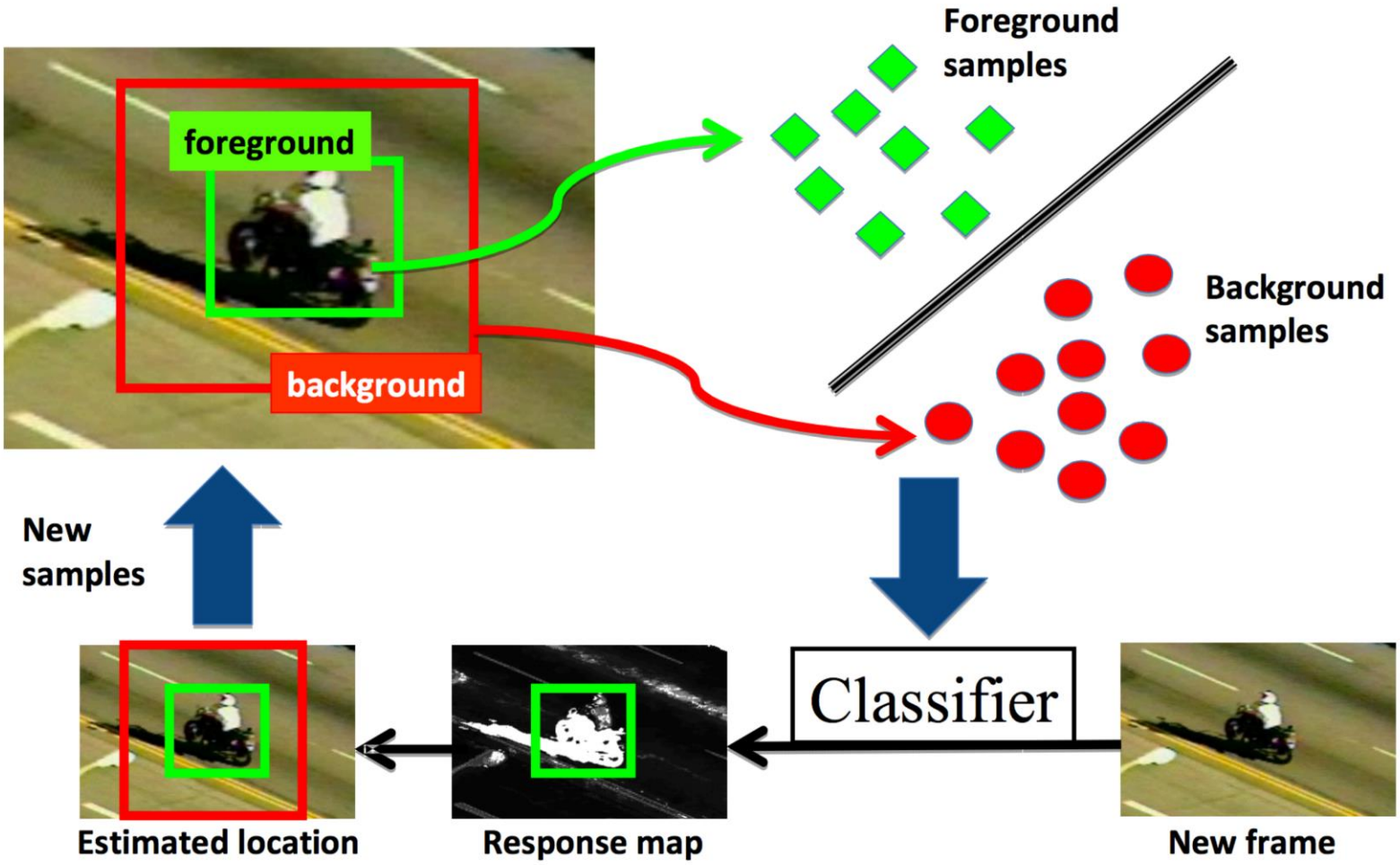

Target Model



On-line Learning

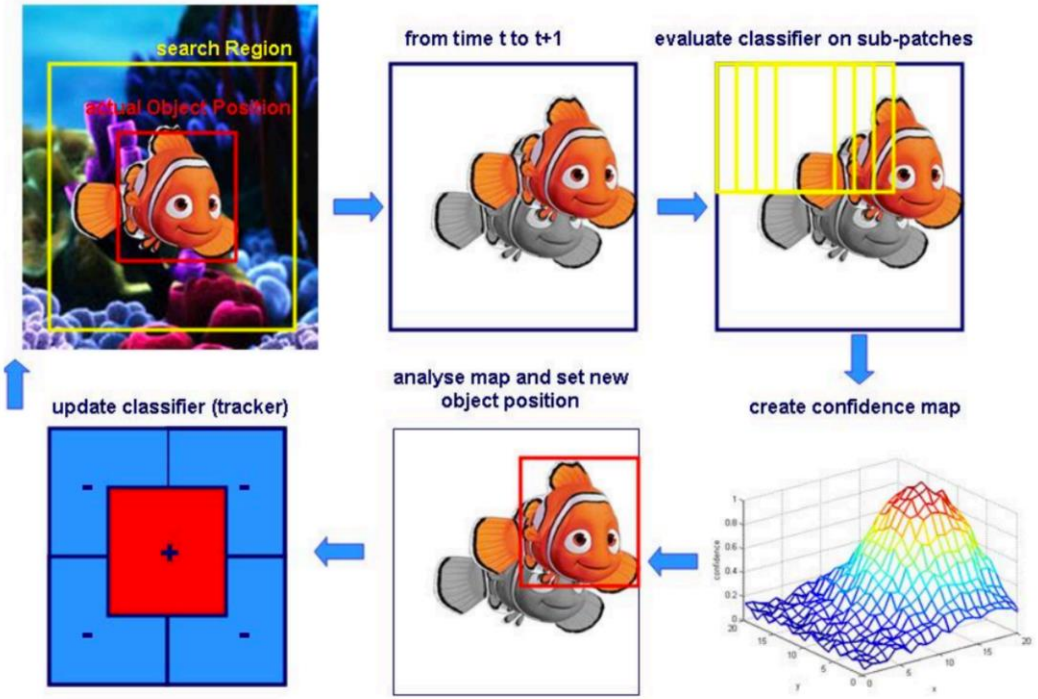
- Discriminative modeling (tracking-by-detection)
- Learn and apply a detector or predictor
- Challenges:
  - What are the training data? Label?
  - How to avoid drift? Handle occlusion?
  - How to control complexity?

On-line Learning



On-line Learning

- On-line discriminative learning
- One-shot learning
- On-line update of the classifier



Grabner and Bischof CVPR06

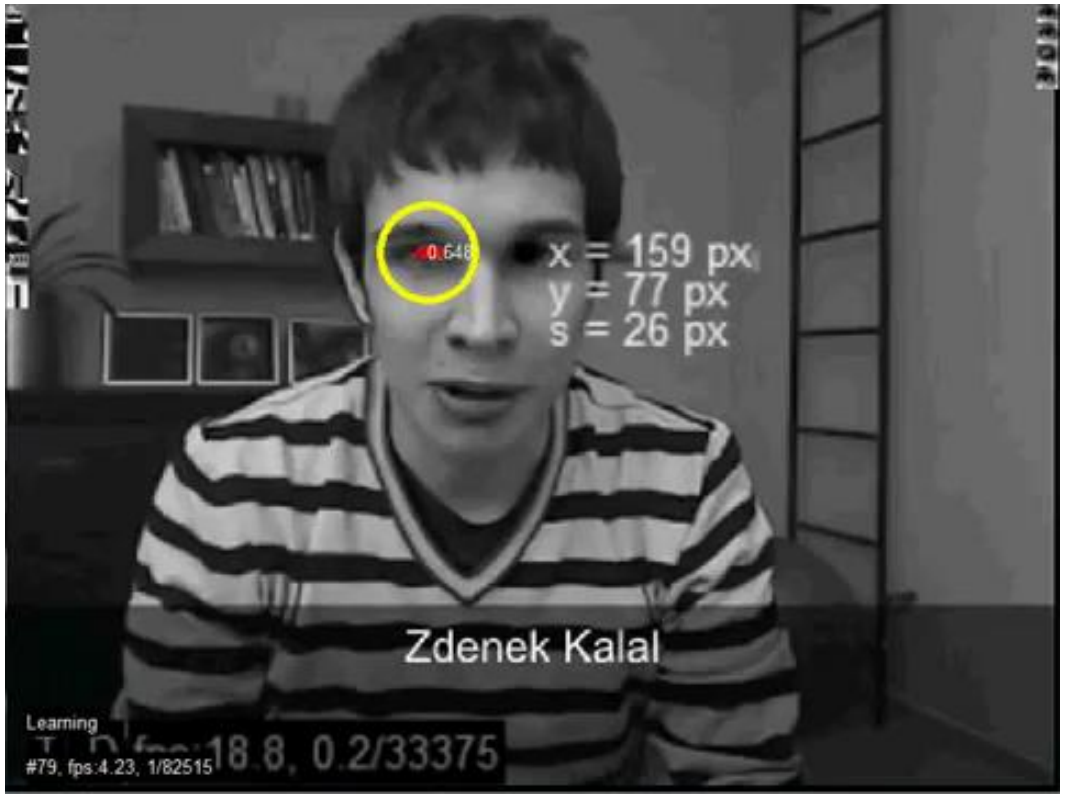
On-line Learning

- Examples of on-line discriminative learning
  - Multiple Instance Learning [1]
  - Kernelized Structured SVM [2]
  - Combine short track + detector [3]

[1] Babenko, Boris, Ming-Hsuan Yang, and Serge Belongie. "Visual tracking with online multiple instance learning." CVPR 2009
[2] Hare, Sam, Amir Saffari, and Philip HS Torr. "Struck: Structured output tracking with kernels." ICCV 2011
[3] Kalal, Zdenek, KrysEan Mikolajczyk, and Jiri Matas. "Tracking-learning-detection." PAMI 2012

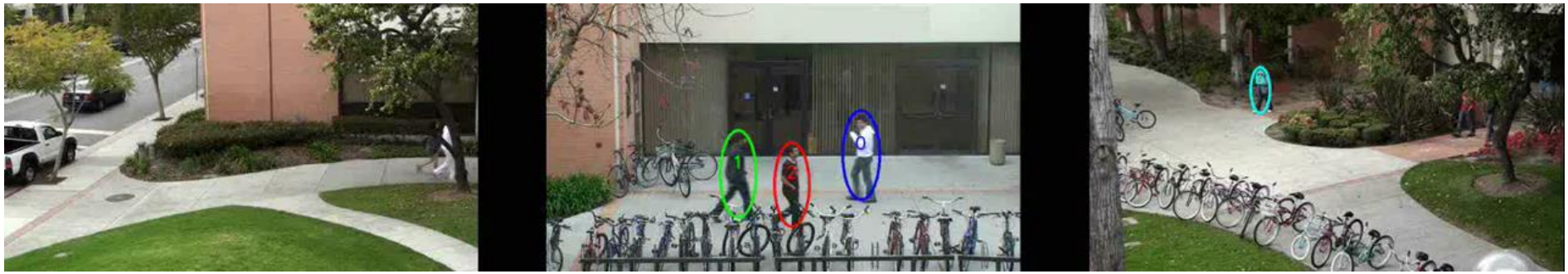
On-line Learning

- Example: TLD



Multi-camera Tracking

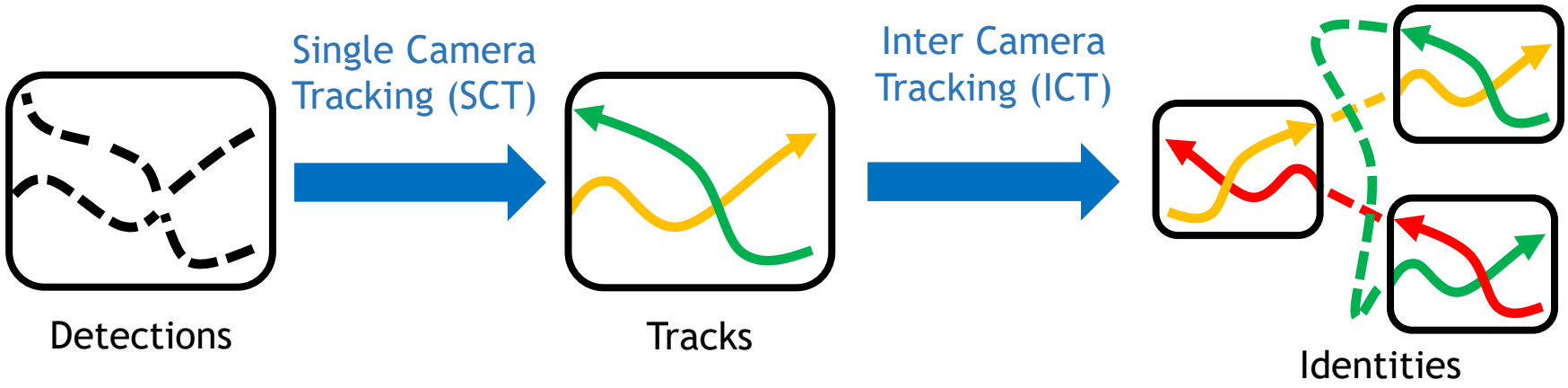
- Track objects in across multiple cameras



Ref: Y. Cai and G. Medioni, "Exploring context information for inter-camera multiple target tracking," *WACV2014*.

Multi-camera Tracking

- Multi-camera Tracking = single camera tracking (SCT) + inter camera tracking (ICT)

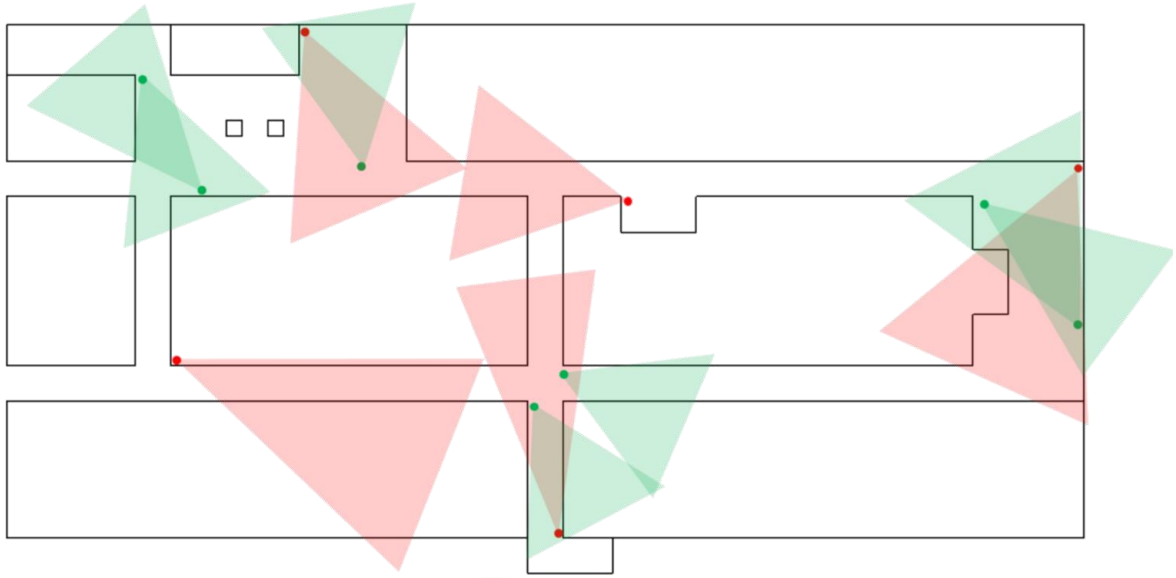


Ref: S. Zhang, E. Staudt, T. Faltemier, and A. K. Roy-Chowdhury, "A Camera Network Tracking (CamNeT) Dataset and Performance Baseline," *WACV 2015*.

Multi-camera Tracking

- Applications
  - Surveillance camera network
  - Customer flow analysis
  - Homeland security
  - Monitoring of elders, children, and patients
  - Crime prevention
  - Traffic control
  - …

Re-identification



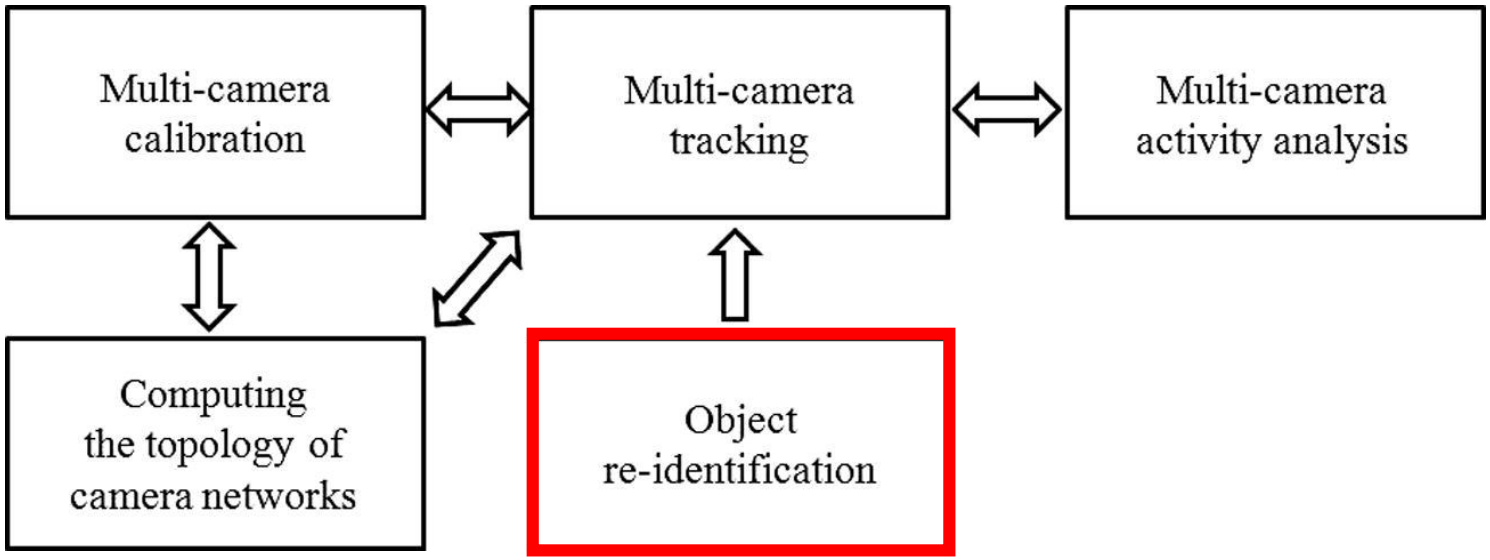
Same person?

Re-identification

- is not an easy task…

The Relationship of Multi-camera Tracking and Re-identification

- Multi-camera tracking framework

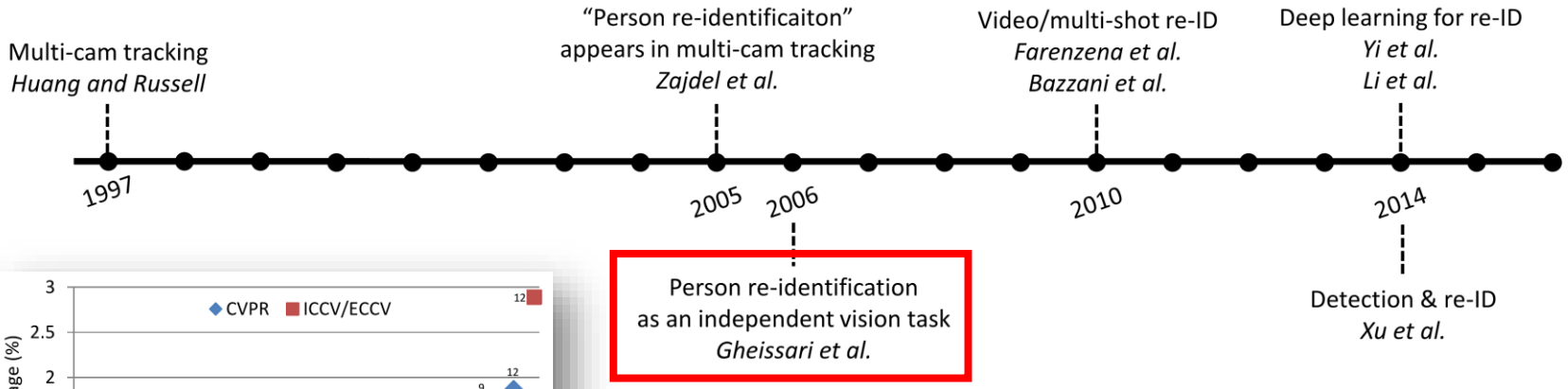


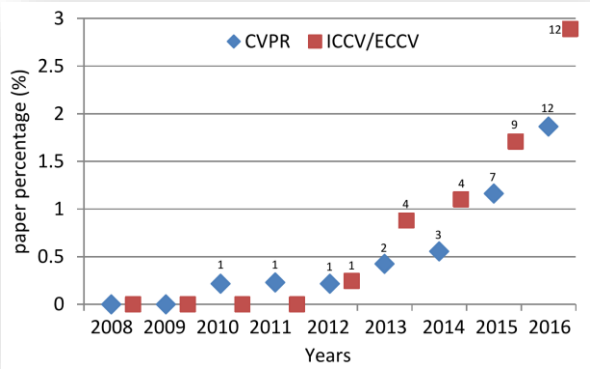
An important part of multi-camera tracking

Ref: X. Wang, “Intelligent multi-camera video surveillance: a review,” *Pattern recognition letters*, vol. 34, no. 1, pp. 3–19, 2013.

The Relationship of Multi-camera Tracking and Re-identification

- History of re-identification



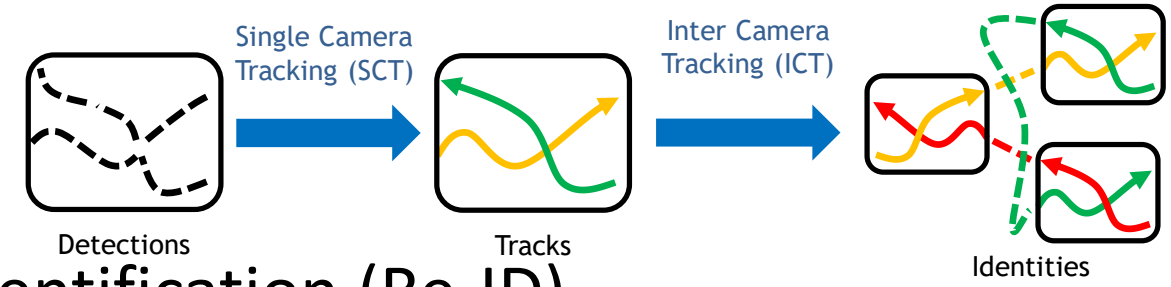


The first time image re-identification became an independent computer vision task

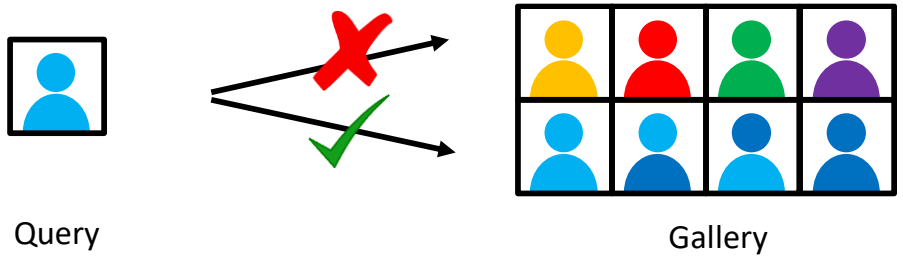
Ref: L. Zheng, Y. Yang, and A. G. Hauptmann, "Person reidentification: Past, present and future," *arXiv preprint arXiv:1610.02984*, 2016.

The Relationship of Multi-camera Tracking and Re-identification

- Multi-camera tracking (MCT)
  - A data association problem



- Re-identification (Re-ID)
  - A retrieval problem

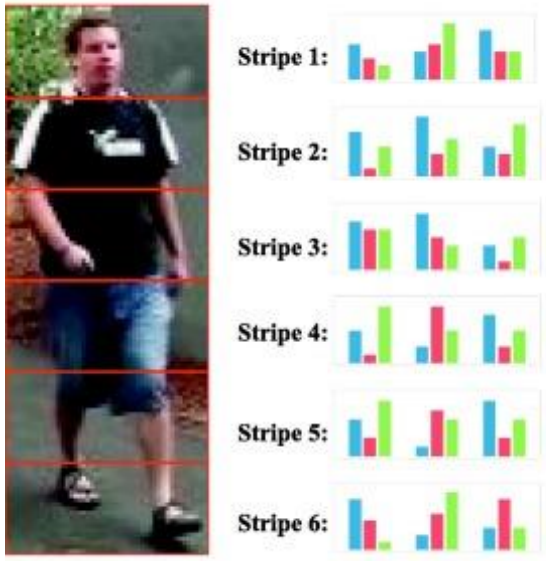


Person Re-identification

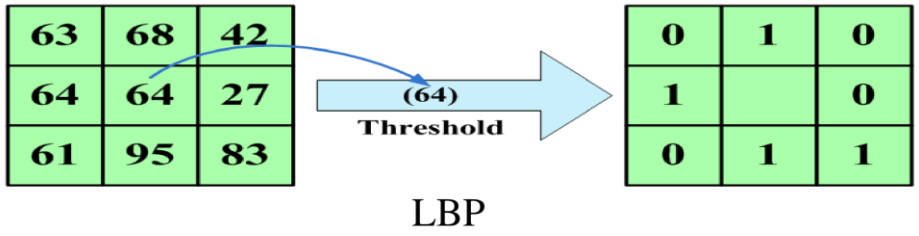
- Feature + distance function
  - Feature: LOMO, BoW, HistLBP, DGD-CNN, …
  - Distance function: XQDA, KISSME, Mahalanobis distance, and Euclidean distance, …
- Re-identification system with deep learning

Features

- Typical feature



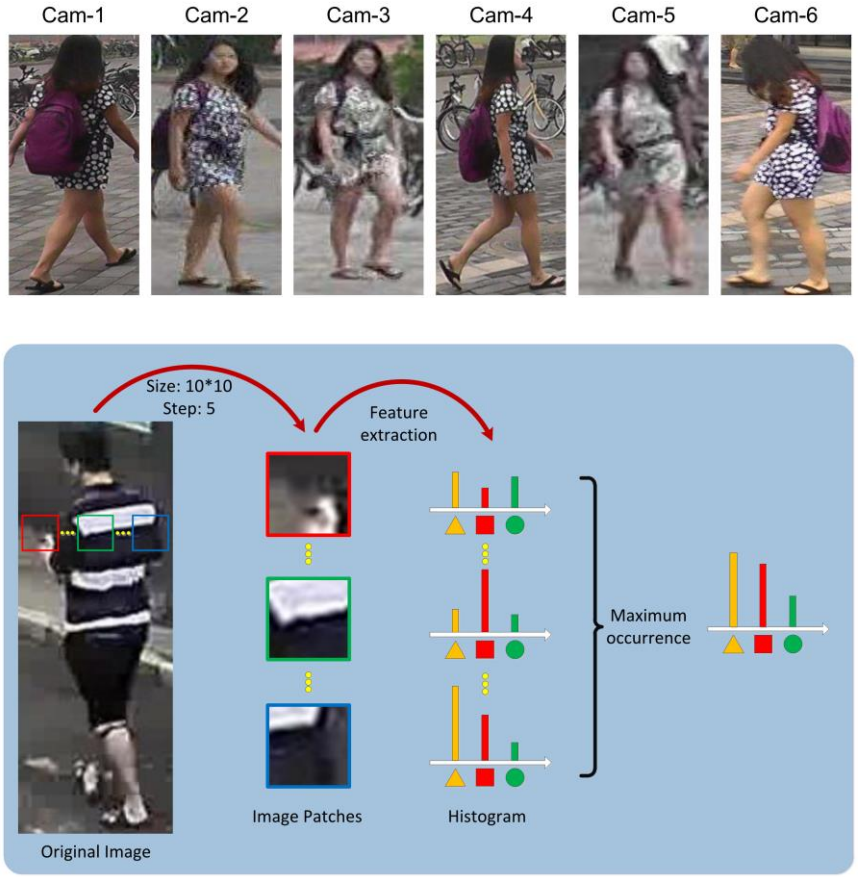
- HistLBP



Ref: B. Prosser, W.-S. Zheng, S. Gong, T. Xiang, and Q. Mary, “Person re-identification by support vector ranking,” *BMVC2010*.

Features

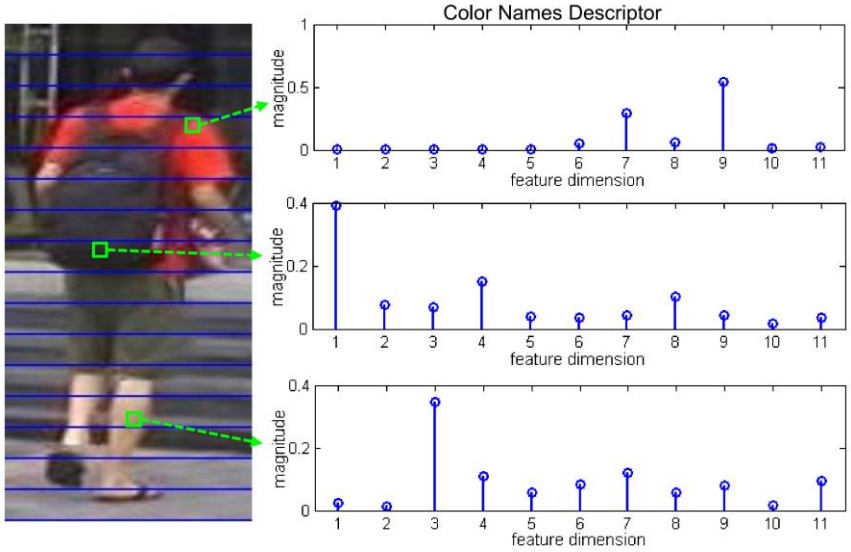
- LOMO
  - Retinex algorithm for illumination variation
  - HSV color histogram as a color descriptor, Scale Invariant Local Ternary Pattern (SILTP) as a texture descriptor
  - Sliding window + maximum occurrence for view invariance



Ref: S. Liao, Y. Hu, X. Zhu, and S. Z. Li, “Person re-identification by local maximal occurrence representation and metric learning,” *CVPR 2015*.

Features

- BoW
  - Feature Extraction
    - Color name (CN) descriptor for each 4x4 non-overlapped patch
    - Local features are quantized, and pooled in a histogram for each horizontal stripe.
  - Code book is generated from a training set
  - Quantization: a feature is represented by 10 visual words



Ref: L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: a benchmark," *ICCV2015*.

Distance Function

- Mahalanobis distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \ \Sigma^{-1}(\mathbf{x}_i - \mathbf{x}_j)$$

- KISSME

Ω_I Intra-person variation

Ω_E Extra-person variation

$$P(\Delta|\Omega_I) = \frac{1}{(2\pi)^{d/2}|\Sigma_I|^{1/2}} e^{-\frac{1}{2}\Delta^T \Sigma_I^{-1} \Delta},$$

$$P(\Delta|\Omega_E) = \frac{1}{(2\pi)^{d/2}|\Sigma_E|^{1/2}} e^{-\frac{1}{2}\Delta^T \Sigma_E^{-1} \Delta},$$

$$d(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T (\Sigma_I^{-1} - \Sigma_E^{-1})(\mathbf{x}_i - \mathbf{x}_j)$$

Ref: M. Kostinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof, “Large scale metric learning from equivalence constraints,” *CVPR 2012*.

Distance Function

- XQDA: Cross-view quadratic discriminant analysis
  - Goal: to find a subspace mapping matrix **w** to reduce the dimension for better classification

    - by maximizing

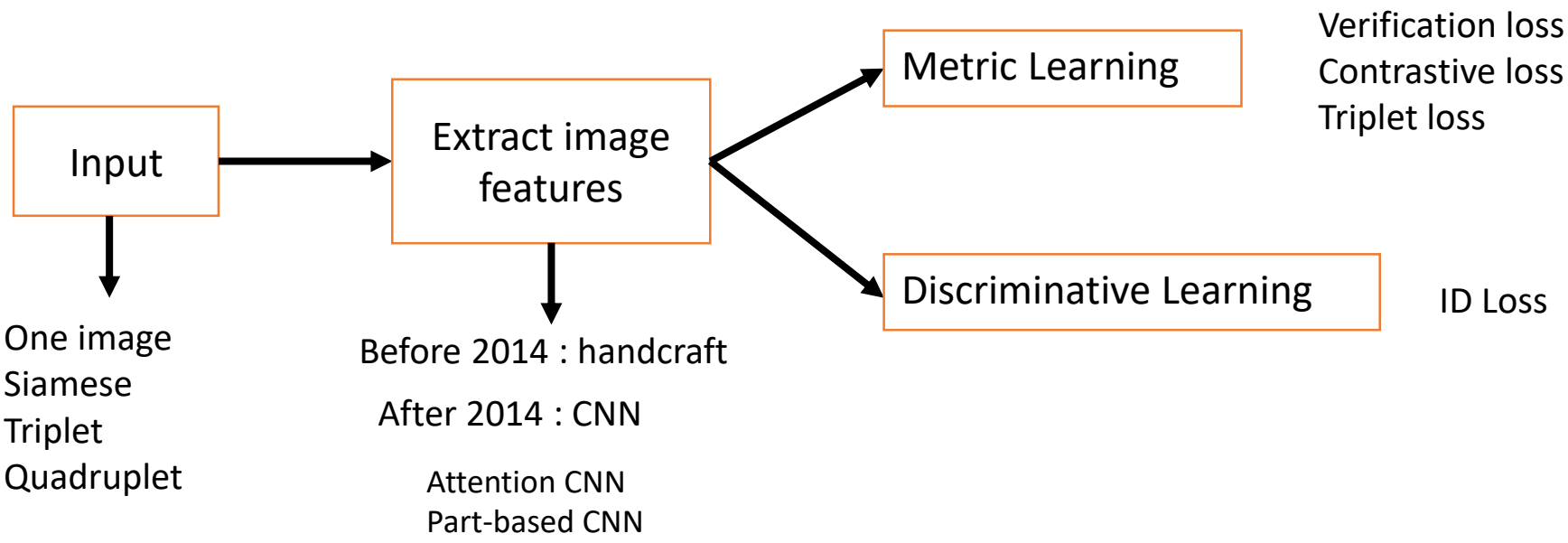
$$W = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_r) \in \mathbb{R}^{d \times r}$$

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \Sigma_E \mathbf{w}}{\mathbf{w}^T \Sigma_I \mathbf{w}}$$

Ref: S. Liao, Y. Hu, X. Zhu, and S. Z. Li, "Person re-identification by local maximal occurrence representation and metric learning," *CVPR 2015*.

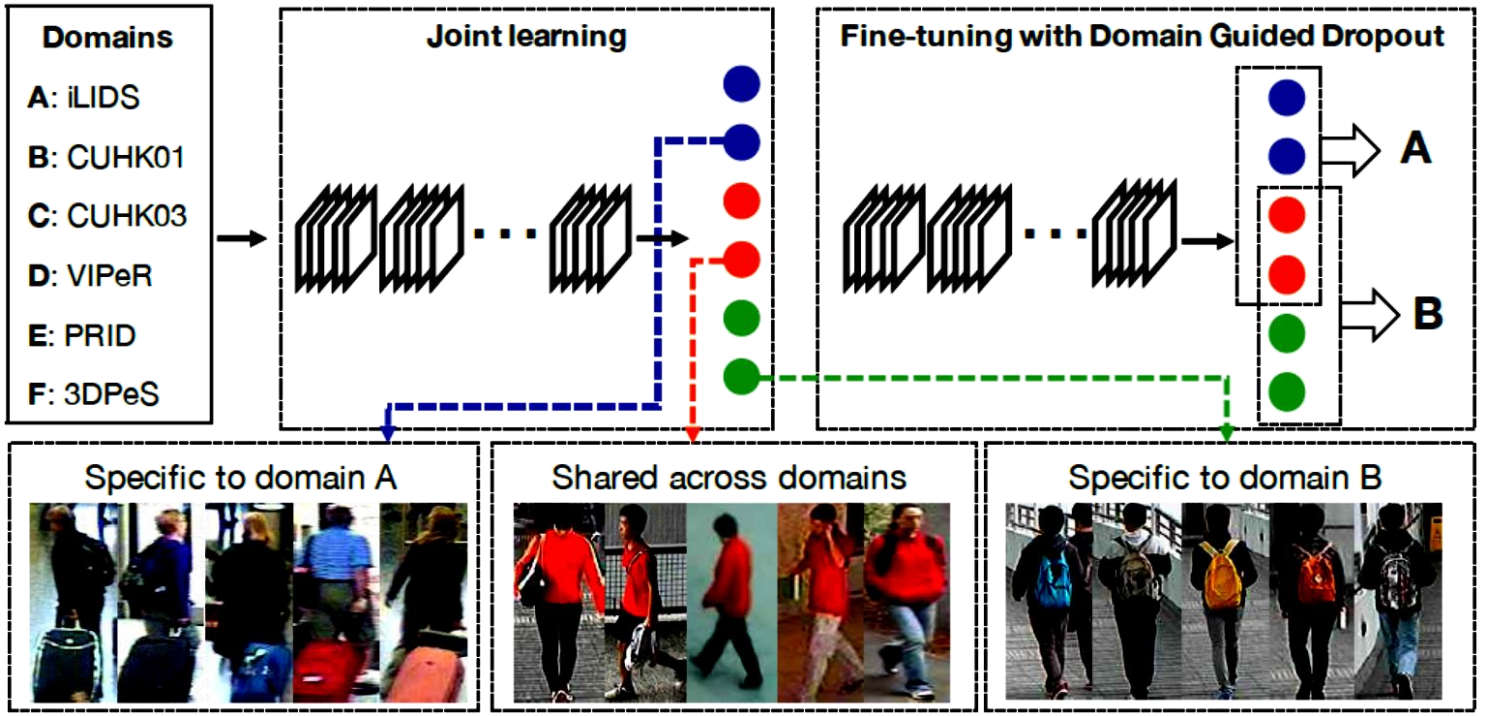
Re-identification System with Deep Learning

- Image-based Re-ID



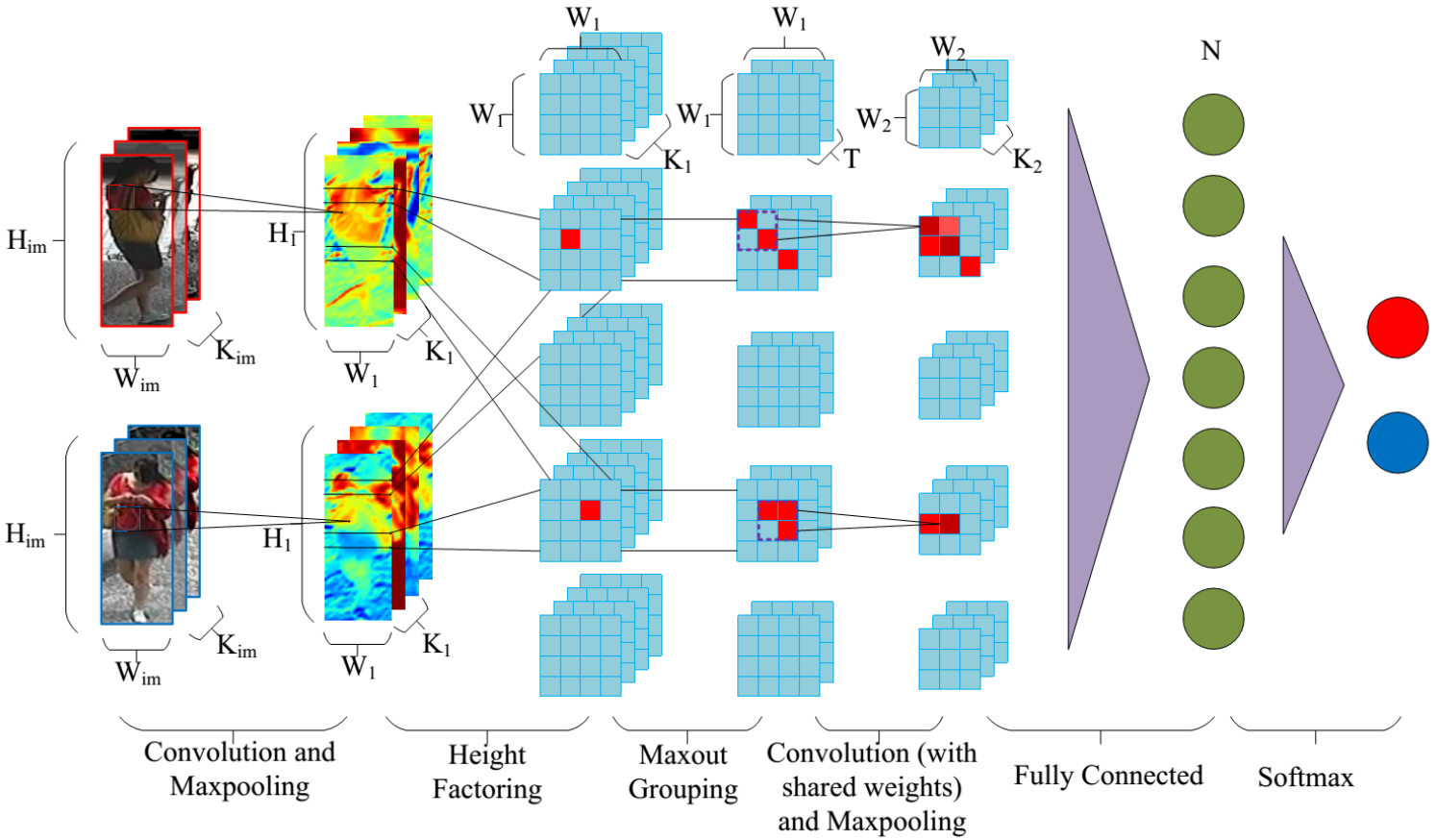
Re-identification System with Deep Learning

- Learning deep feature representation with domain guided dropout



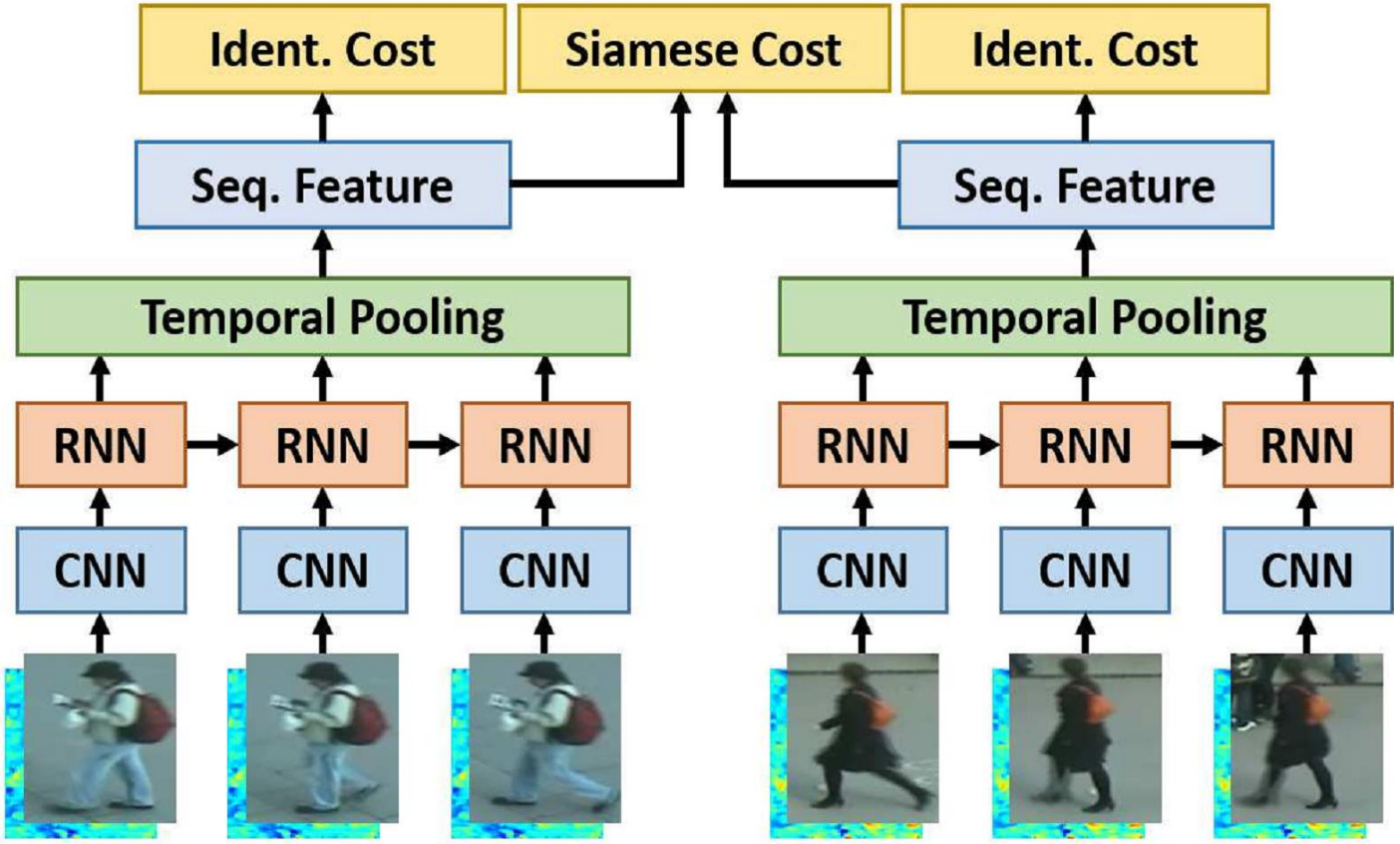
Ref: T. Xiao, H. Li, W. Ouyang, and X. Wang, "Learning deep feature representations with domain guided dropout for person re-identification," *CVPR 2016*.

Re-identification System with Deep Learning



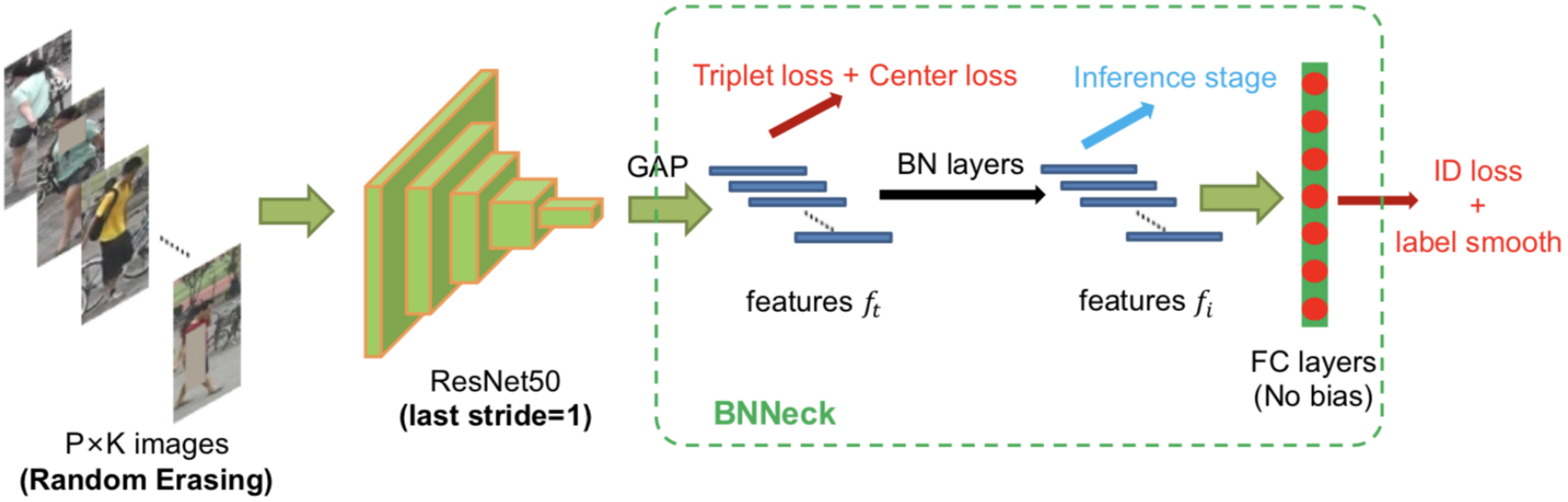
Ref: W. Li, R. Zhao, T. Xiao and X. Wang, "DeepReID: Deep Filter Pairing Neural Network for Person Re-identification," *CVPR2014.*

Re-identification System with Deep Learning



Ref: N. McLaughlin, J. M. del Rincon, and P. Miller, "Recurrent Convolutional Network for Video-Based Person Re-Identification," *CVPR 2016*.

Re-identification System with Deep Learning



Re: H. Luo, Y. Gu, X. Liao, S. Lai and W. Jiang, "Bag of Tricks and a Strong Baseline for Deep Person Re-Identification," *CVPR Workshops (CVPRW)* 2019.