Global Motion Estimation

- Bergen *et al*
  - affine transformation
- Mann & Piccard
  - Homography,
  - Bilinear,
  - Pseudo Perspective

- Global Flow
- Registration
- Alignment
- Motion Compensation
- Parametric motion estimation
- Warping

- Applications
  - Removing camera jitter/motion
  - Moving object detection
  - Mosaic generation
  - Geo registration
  - Tracking

KLT Tracking

Lecture-10

Tracking

- Tracking deals with estimating the trajectory
- of an object in the image plane as it moves around a scene.
  - Object tracking (car, airplane, person)
  - Feature (Harris corners) Tracking
  - Single object tracking
  - Multiple Object tracking
  - Tracking in fixed camera
  - Tracking in moving camera
  - Tracking in multiple cameras

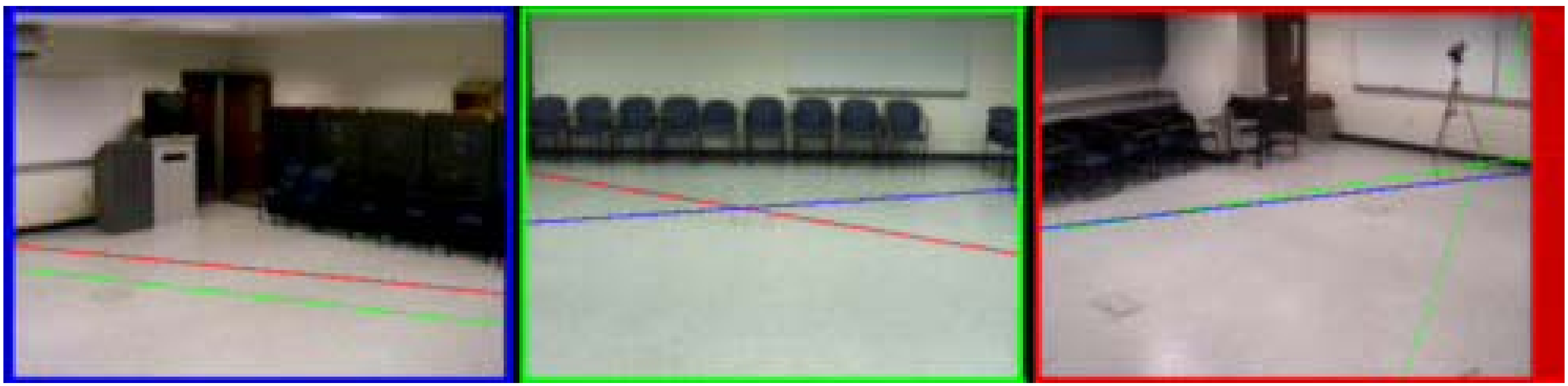
Tracking A Single Point

Tracking Bounding Boxes

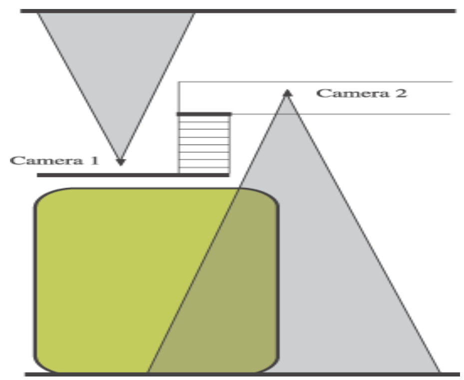
Tracking Object Contours

Multiple Fixed & Overlapping Cameras Tracking

Camera1 Camera2 Camera3



Multiple Fixed & Non-Overlapping Cameras Tracking



Tracking In Moving Camera

ECCV-2012

- Hamid Izadinia, Imran Saleemi, Wenhui Li and Mubarak Shah, (MP)^{2}T: Multiple People Multiple Parts Tracker, European Conference on Computer Vision 2012, Florence, Italy, October 7-13, 2012. [Video of Presentation]
  - http://www.youtube.com/watch?v=YhyMcWnJf9g&feature=plcp

- Amir Roshan Zamir, Afshin Dehghan and Mubarak Shah, GMCP-Tracker: Global Multi-object Tracking Using Generalized Minimum Clique Graphs, European Conference on Computer Vision 2012, Florence, Italy, October 7-13, 2012. [Video of Presentation]
  - http://www.youtube.com/watch?v=f4Muu1d7NhA&feature=plcp

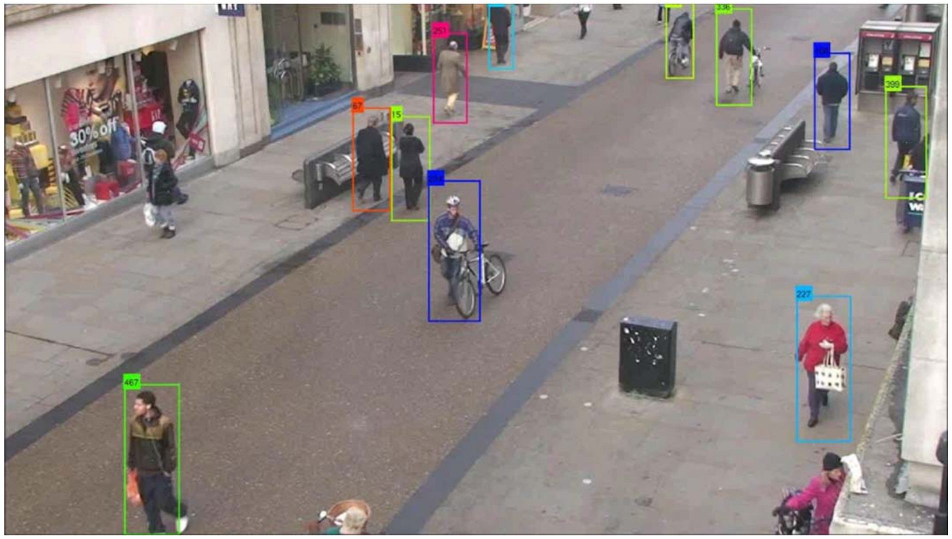
PETS2009-S2L1 Results

http://www.youtube.com/watch?v=f4Muu1d7NhA&feature=plcp

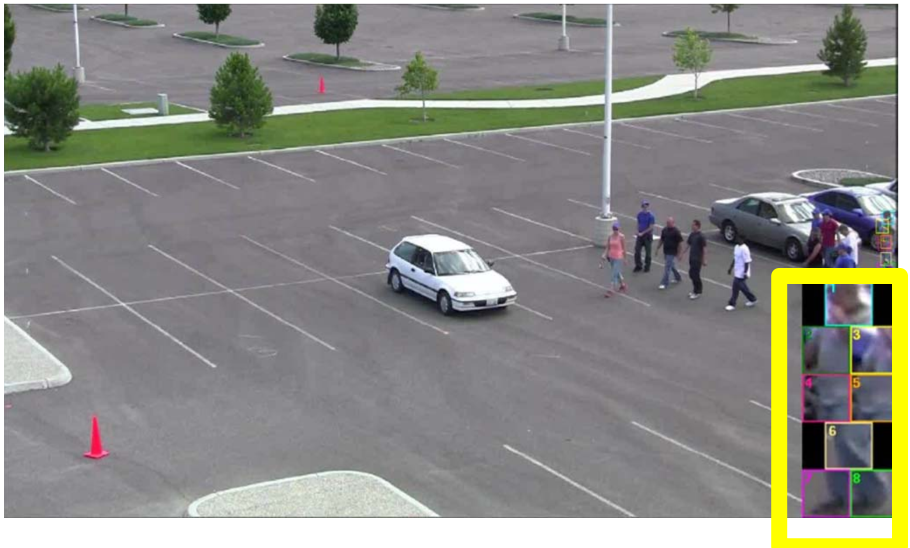
ECCV 2012

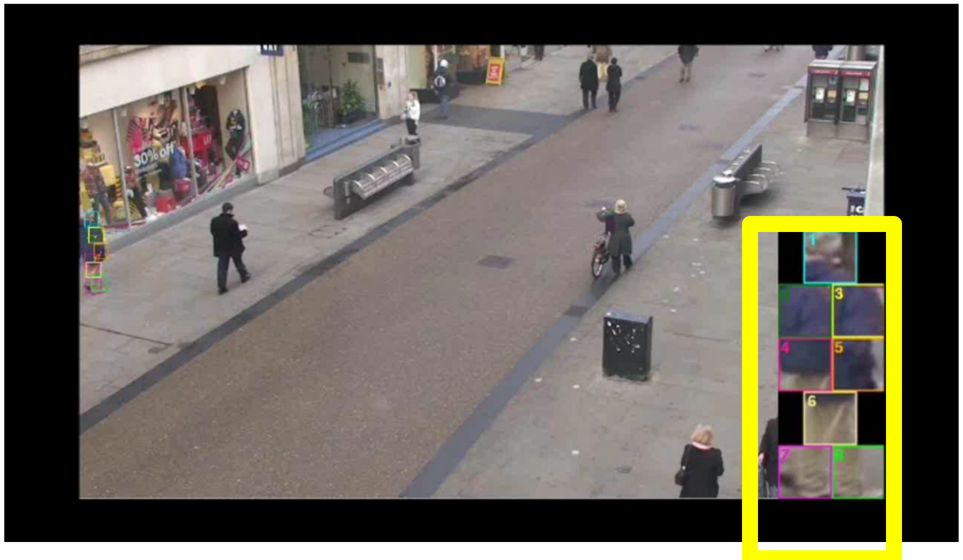
http://www.youtube.com/watch?v=YhyMcWnJf9g&feature=youtu.be
ECCV2012

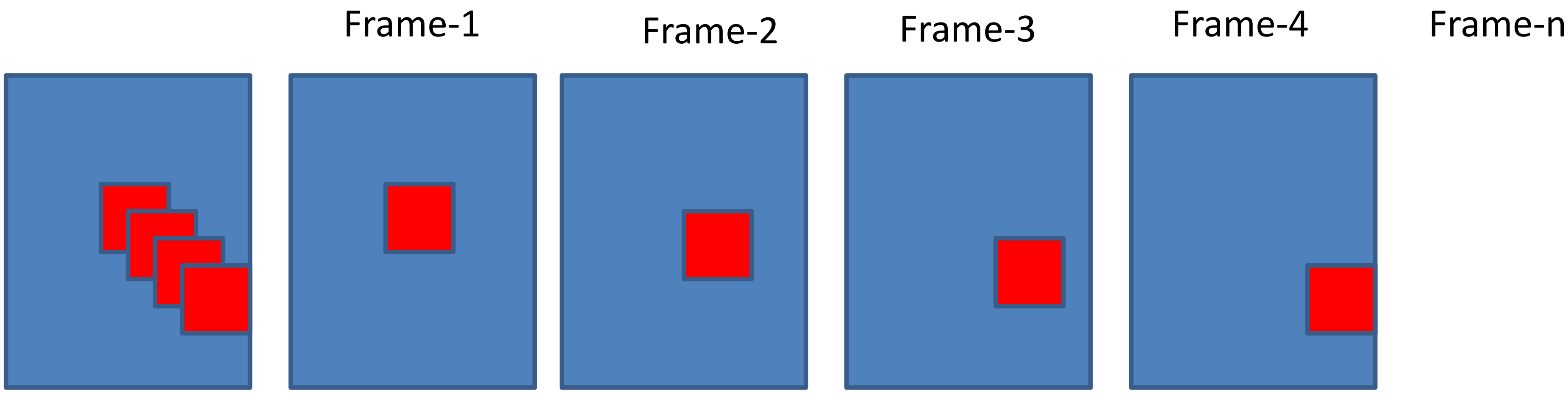
Person Tracking



Part Tracking

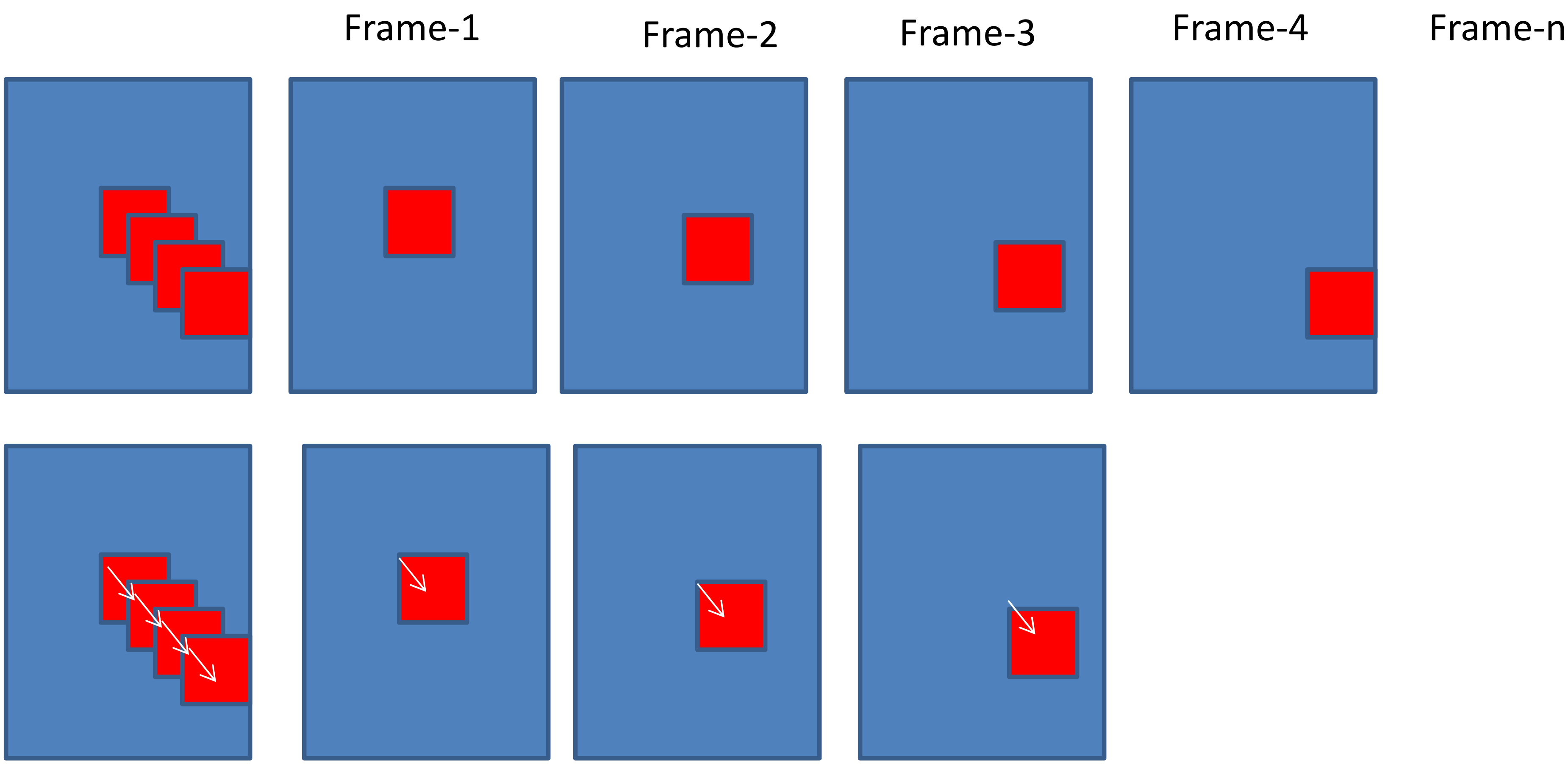






Frame-1
Frame-2
Frame-3
Frame-4
Frame-n

KLT(Kanade-Lucas-Tomasi) Tracker

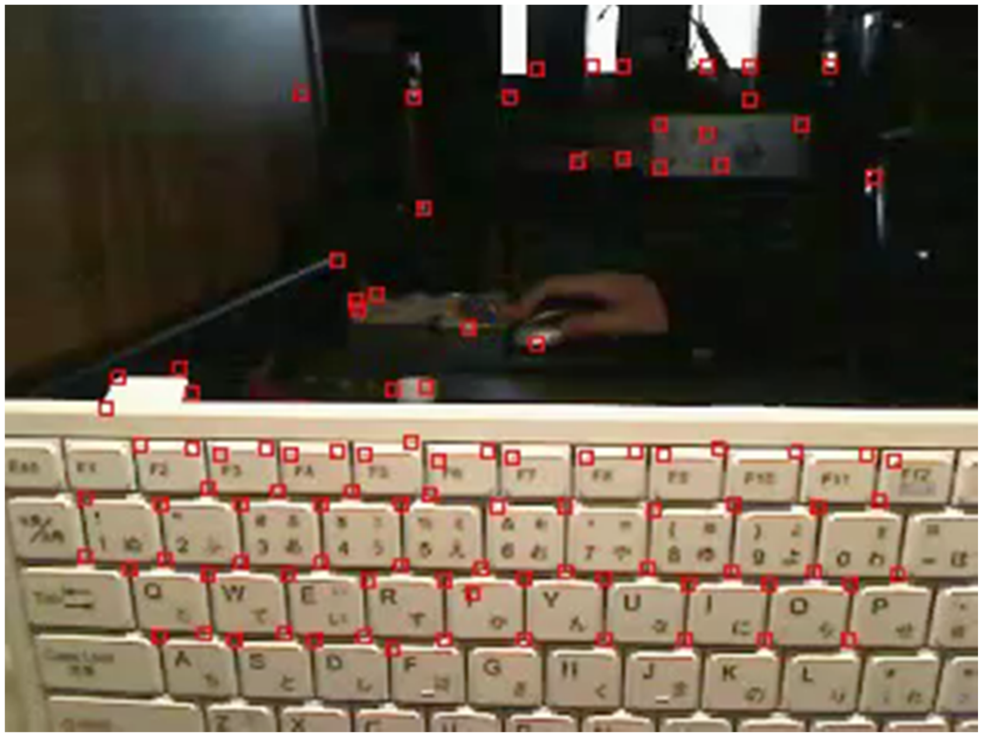


Simple KLT Algorithm

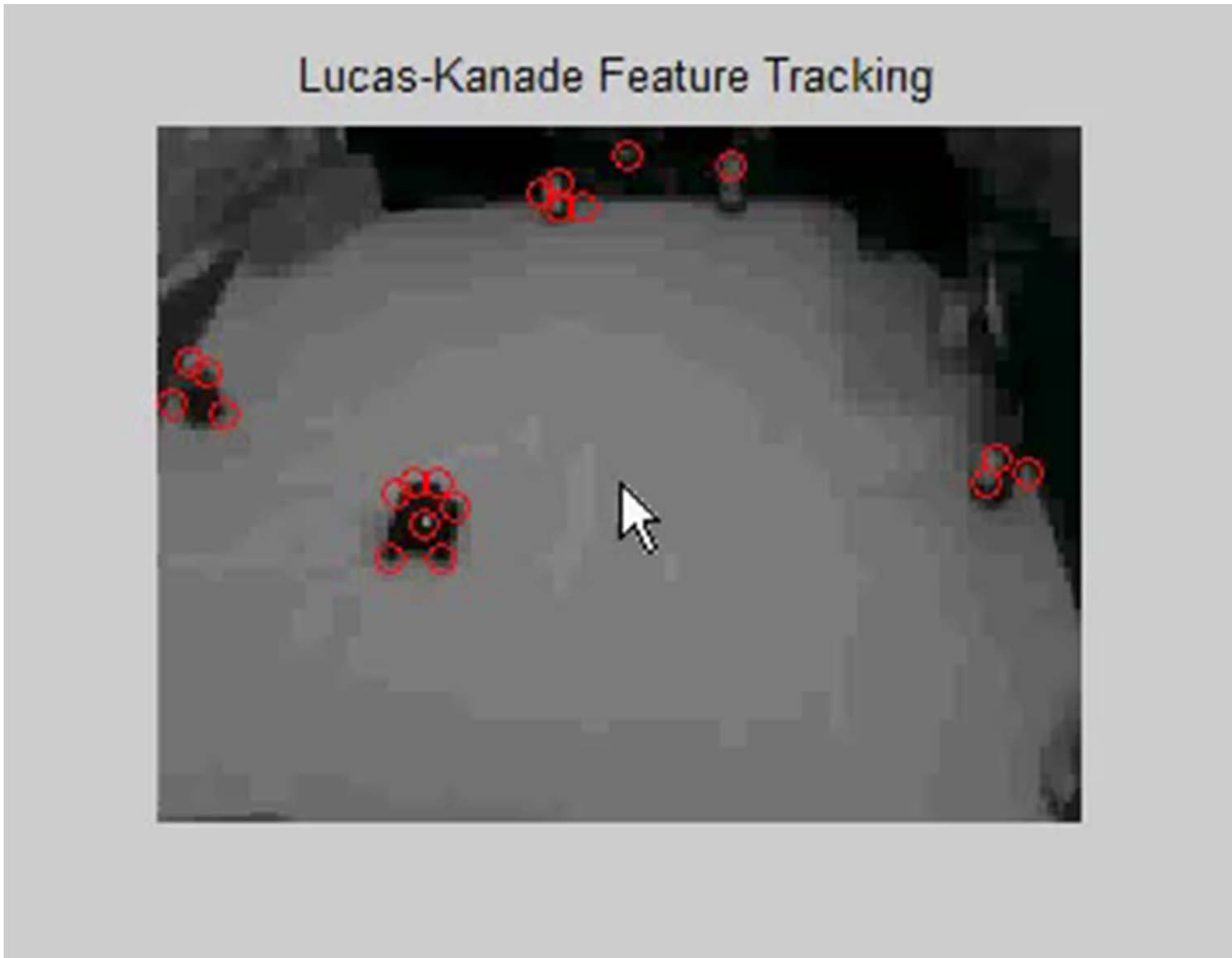
1. Detect Harris corners in the first frame
2. For each Harris corner compute motion (translation or affine) between consecutive frames.
3. Link motion vectors in successive frames to get a track for each Harris point
4. Introduce new Harris points by applying Harris detector at every *m* (10 or 15) frames
5. Track new and old Harris points using steps 1-3.

KLT Results

KLT Results



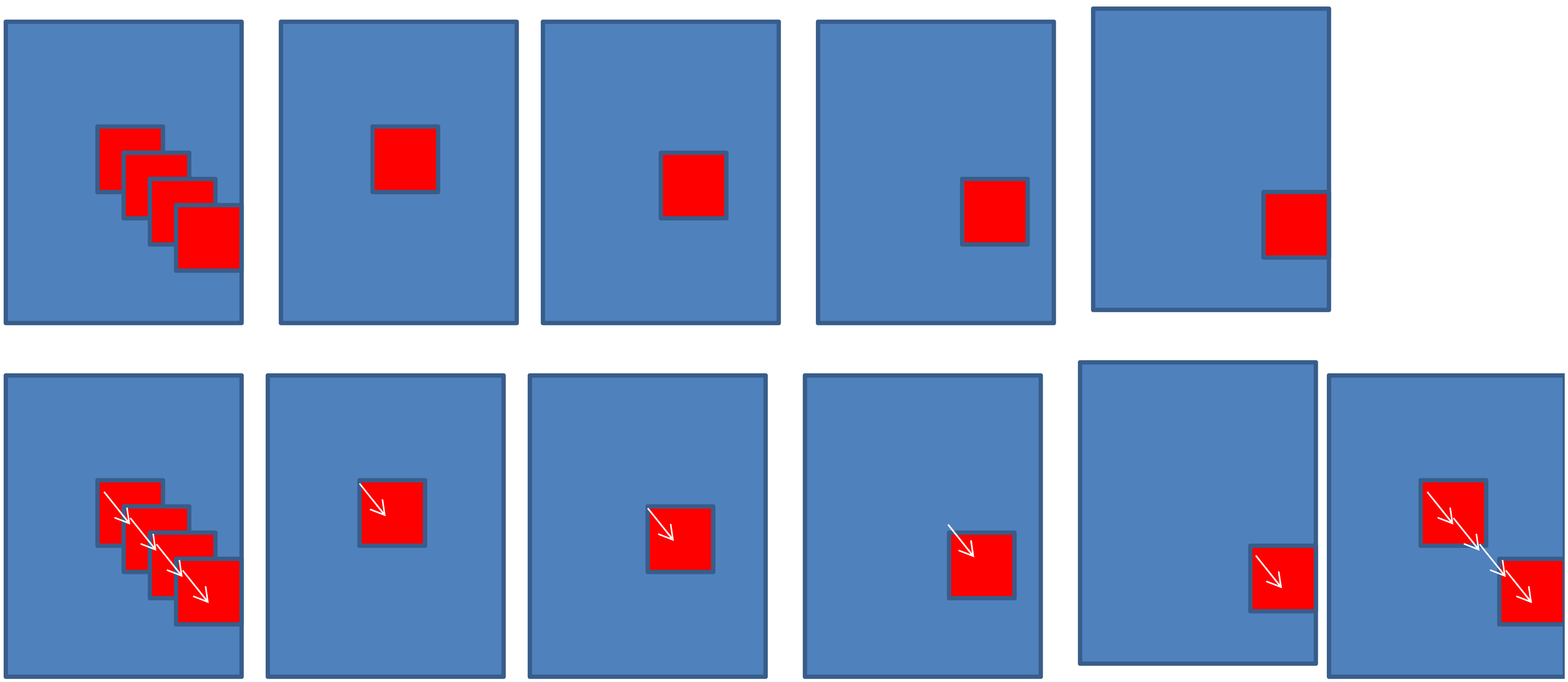
KLT Results



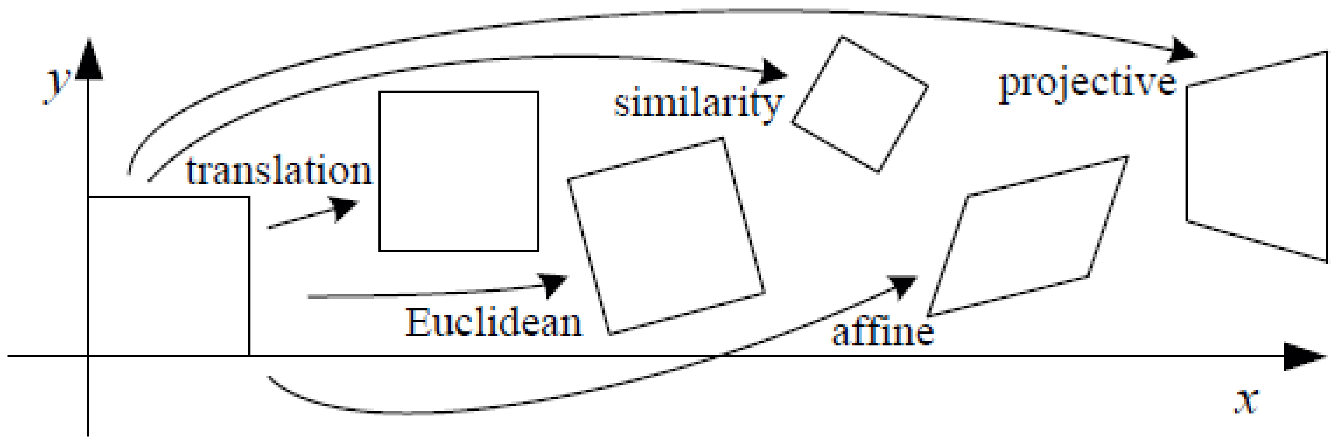
KLT Results

KLT Results

How to estimate alignment?



Basic Set of 2-D Transformation



Richard Szeliski, "Computer Vision: Algorithms and Application".

Summary of Displacement Models (2-D Transformations)

Translation

$$x' = x + b_1$$
$$y' = y + b_2$$

Rigid

$$x' = x\cos\theta - y\sin\theta + b_1$$
$$y' = x\sin\theta + y\cos\theta + b_2$$

Affine

$$x' = a_1 x + a_2 y + b_1$$
$$y' = a_3 x + a_4 y + b_2$$

Projective

$$x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}$$
$$y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}$$

Bi-quadratic

$$x' = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 y^2 + a_6 xy$$
$$y' = a_7 + a_8 x + a_9 y + a_{10} x^2 + a_{11} y^2 a_{12} xy$$

Bi-Linear

$$x' = a_1 + a_2 x + a_3 y + a_4 xy$$
$$y' = a_5 + a_6 x + a_7 y + a_8 xy$$

Pseudo-Perspective

$$x' = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy$$
$$y' = a_6 + a_7 x + a_8 y + a_4 xy + a_5 y^2$$

$$W(x;p) = \begin{bmatrix} x + b_1 \\ y + b_2 \end{bmatrix}$$

$$W(x;p) = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W(x;p) = [R \mid t]_{2X3} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W(x;p) = [sR \mid t]_{2X3} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W(x;p) = [A]_{2X3} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W(x;p) = [H]_{3X3} \begin{bmatrix} x \\ y \end{bmatrix}$$

Displacement Models Parameterizations

Homogenous coordinates

Translation

$$x' = x + b_1$$
$$y' = y + b_2$$
$$W(\mathbf{x};\mathbf{p}) = (x + b_1, y + b_2)$$

$$W(\mathbf{x};\mathbf{p}) = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rigid

$$x' = x\cos\theta - y\sin\theta + b_1$$
$$y' = x\sin\theta + y\cos\theta + b_2$$
$$W(\mathbf{x};\mathbf{p}) = (x\cos\theta - y\sin\theta + b_1, x\sin\theta + y\cos\theta + b_2)$$

$$W(\mathbf{x};\mathbf{p}) = \begin{bmatrix} c\theta & -s\theta & b_1 \\ s\theta & c\theta & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

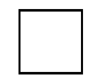
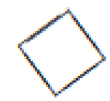
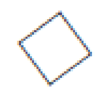
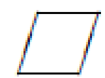
$$W(\mathbf{x};\mathbf{p}) = [R \mid t]_{2X3} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

$$x' = a_1 x + a_2 y + b_1$$
$$y' = a_3 x + a_4 y + b_2$$
$$W(\mathbf{x};\mathbf{p}) = (a_1 x + a_2 y + b_1, a_3 x + a_4 y + b_2)$$

$$W(\mathbf{x};\mathbf{p}) = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W(\mathbf{x};\mathbf{p}) = A_{2X3} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

Richard Szeliski, "[Computer Vision: Algorithms and Application]".

Derivative & Gradient

$$\text{Function: } f(x)$$

$$\text{Derivative: } f'(x) = \frac{df}{dx},\ x \text{ is a scalar}$$

$$\text{Function: } f(x_1, x_2, \ldots, x_n)$$

$$\text{Gradient: } \nabla f(x_1, x_2, \ldots, x_n) = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n})$$

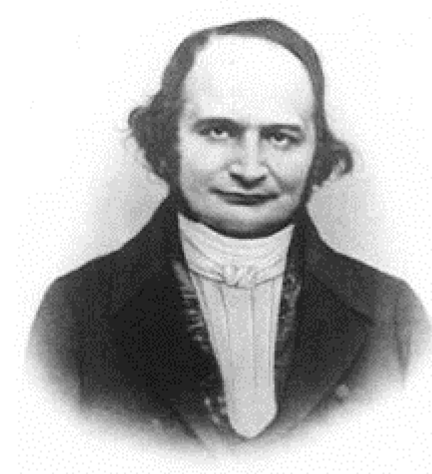
Jacobian

$$F(x_1, x_2, \ldots, x_n) = (f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_m(x_1, x_2, \ldots, x_n))$$

Vector Valued Function

Derivative?

$$J(F) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$



Carl Gustav Jacob Jacobi
10 December 1804—
18 February 1851

Displacement Model Jacobians

Translation

$$W(\mathbf{x};\mathbf{p}) = (x + b_1, y + b_2) \qquad \frac{\partial W}{\partial \mathbf{p}}?$$

$$\frac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Rigid

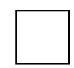
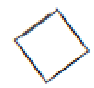
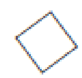
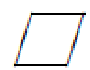
$$W(\mathbf{x};\mathbf{p}) = (x\cos\theta - y\sin\theta + b_1, x\sin\theta + y\cos\theta + b_2)$$

$$\frac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} 1 & 0 & -xs\theta - yc\theta \\ 0 & 1 & xc\theta - ys\theta \end{bmatrix}$$

Affine

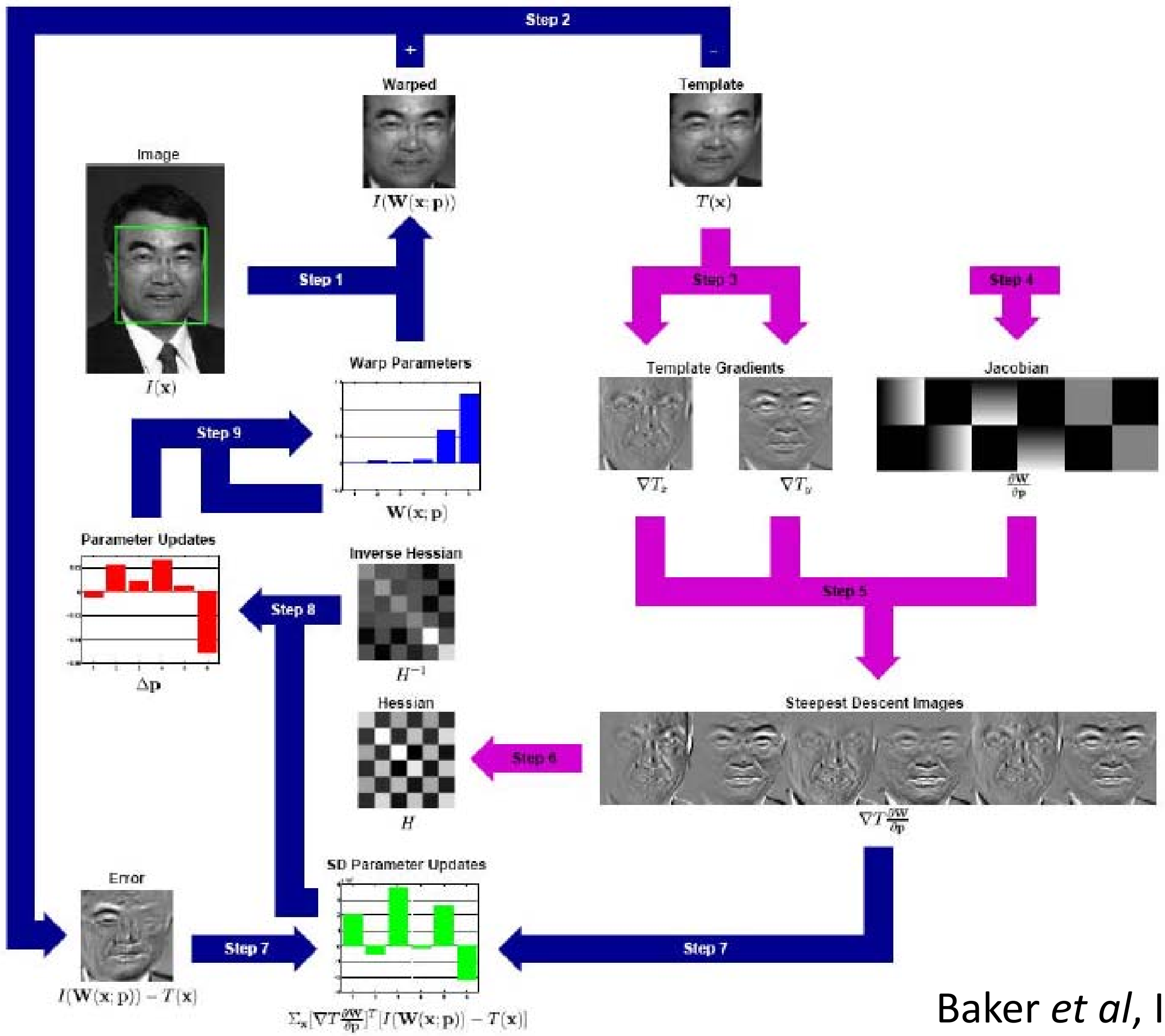
$$W(\mathbf{x};\mathbf{p}) = (a_1 x + a_2 y + b_1, a_3 x + a_4 y + b_2)$$

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

Transformation	Matrix	# DoF	Preserves	Icon	Parameters $\boldsymbol{p}$	Jacobian $\boldsymbol{J}$
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation		(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths		(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles		(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism		$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines		$(h_{00}, h_{01}, \ldots, h_{21})$	(see Section 6.1.3)

Richard Szeliski, "Computer Vision: Algorithms and Application".

Finding Alignment



Baker *et al*, IJCV, 2004.

Finding Alignment

Find $\mathbf{p}$ s.t. following is minimized

$$\sum_{\mathbf{x}}[I(W(\mathbf{x};\mathbf{p}))-T(\mathbf{x})]^2$$

Assume initial estimate of $\mathbf{p}$ is known, find $\Delta\mathbf{p}$

$$\sum_{\mathbf{x}}[\boxed{I(W(\mathbf{x};\mathbf{p}+\Delta\mathbf{p}))}-T(\mathbf{x})]^2$$

Find Taylor Series

$$\sum_{\mathbf{x}}[I(W(\mathbf{x};\mathbf{p}))+\nabla I\frac{\partial W}{\partial \mathbf{p}}\Delta\mathbf{p}-T(\mathbf{x})]^2 \qquad \nabla I=\begin{bmatrix} I_x & I_y \end{bmatrix}$$

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

Differentiate wrt $\Delta\mathbf{p}$ and equate it to zero

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

And equate it to zero to find

$$2\sum_{\mathbf{x}}\left[\nabla I\frac{\partial W}{\partial p}\right]^T[I(W(x;p))+\nabla I\frac{\partial W}{\partial p}\Delta p-T(x)]=0$$

$$\Delta p=H^{-1}2\sum_{\mathbf{x}}\left[\nabla I\frac{\partial W}{\partial p}\right]^T[T(x)-I(W(x;p))]$$

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

Hessian for Translation Motion

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

$$\frac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$H^{-1} = \sum_{\mathbf{x}} \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

$$\nabla I = \begin{bmatrix} I_x & I_y \end{bmatrix}$$

Harris Corner detector

Algorithm (KLT)

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

Algorithm (KLT-Baker *et. al*.)

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

1. Warp I with $W(\mathbf{x};\mathbf{p})$
2. Subtract T from I $[I(W(\mathbf{x};\mathbf{p}))-T(\mathbf{x})]$
3. Compute gradient ∇T
4. Evaluate the Jacobian $\frac{\partial W}{\partial p}$
5. Compute steepest descent $\nabla T\frac{\partial W}{\partial p}$
6. Compute Inverse Hessian H^{-1}
7. Multiply steepest descend with error $\sum_{\mathbf{x}}\left[\nabla T\frac{\partial W}{\partial \mathbf{p}}\right]^{T}[I(W(\mathbf{x};\mathbf{p}))-T(\mathbf{x})]$
8. Compute $\Delta\mathbf{p}$ $\Delta p = H^{-1}\sum_{\mathbf{x}}\left[\nabla T\frac{\partial W}{\partial p}\right]^{T}[I(W(\mathbf{x};\mathbf{p}))-T(\mathbf{x})]$
9. Update parameters $\mathbf{p}\rightarrow\mathbf{p}+\Delta\mathbf{p}$

Algorithm

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

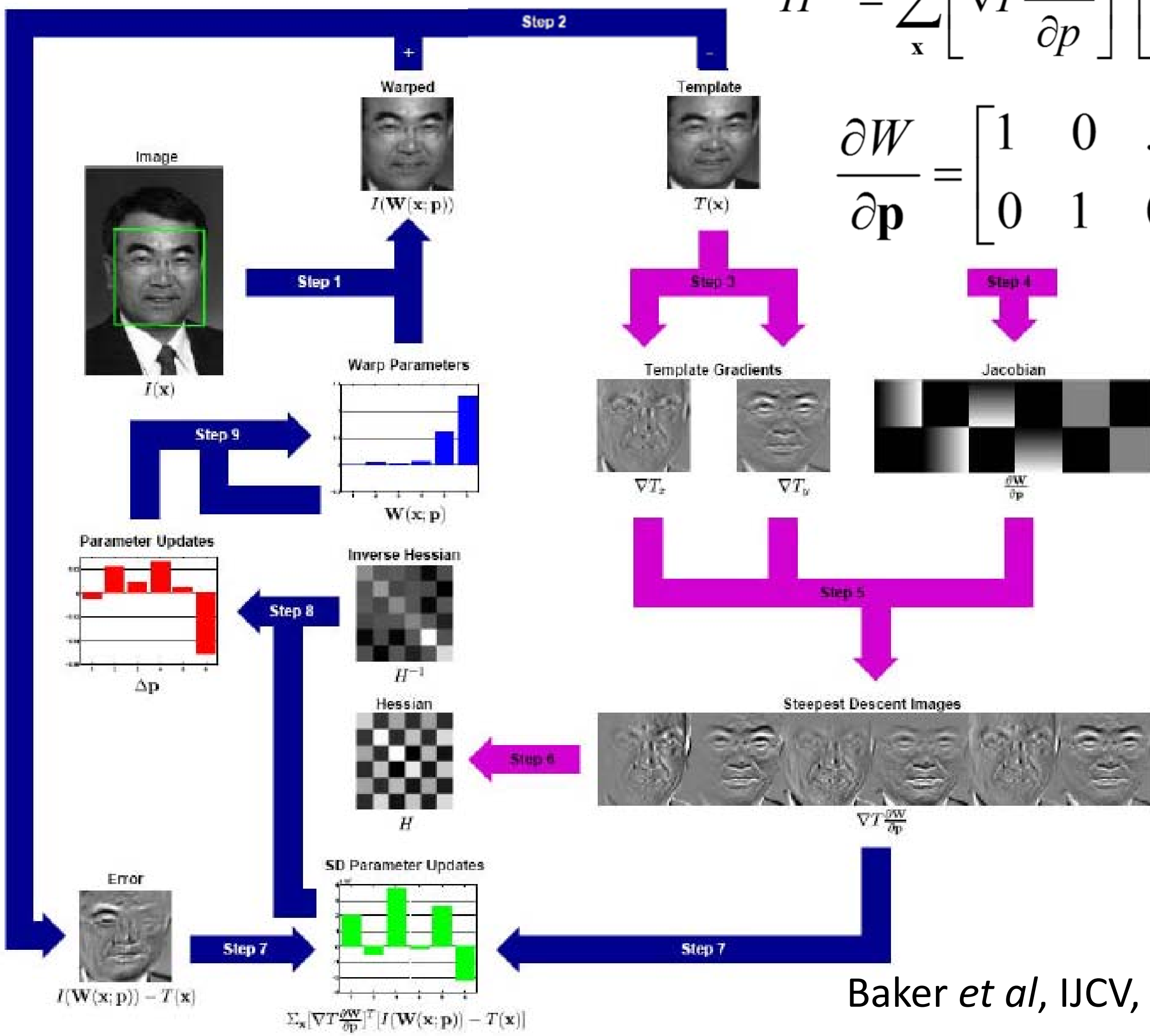
1. Warp I with $W(\mathbf{x};\mathbf{p})$
2. Subtract I from T $[T(\mathbf{x}) - I(W(\mathbf{x};\mathbf{p}))]$
3. Compute gradient ∇I
4. Evaluate the Jacobian $\frac{\partial W}{\partial p}$
5. Compute steepest descent $\nabla I \frac{\partial W}{\partial \mathbf{p}}$ $W(\mathbf{x};\mathbf{p})$
6. Compute Inverse Hessian H^{-1}
7. Multiply steepest descend with error $\sum_{\mathbf{x}} \left[\nabla I \frac{\partial W}{\partial \mathbf{p}} \right]^T [T(\mathbf{x}) - I(W(\mathbf{x};\mathbf{p}))]$
8. Compute $\Delta \mathbf{p}$
9. Update parameters $\mathbf{p} \rightarrow \mathbf{p} + \Delta \mathbf{p}$

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

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

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

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



Baker *et al*, IJCV, 2004.

Comparison of Bergan et al & KLT

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

Bergan

$$\Delta p = H^{-1} 2 \sum_{\mathbf{x}} \left[\nabla I \frac{\partial W}{\partial p}\right]^T [T(x) - I(W(x;p))]$$

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

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

KLT

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

Homework

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

KLT

References

- SIMON BAKER AND IAIN MATTHEWS, “**Lucas-Kanade 20 Years On: A Unifying Framework**”, IJCV, 2004.
- Section 8.2, Richard Szeliski, "Computer Vision: Algorithms and Application".

Implementations

- OpenCV implementation : http://www.ces.clemson.edu/~stb/klt/
- Some Matlab implementation:
  Lucas Kanade with Pyramid
  - http://www.mathworks.com/matlabcentral/fileexchange/30822
  - Affine tracking : http://www.mathworks.com/matlabcentral/fileexchange/24677-lucas-kanade-affine-template-tracking
  - http://vision.eecs.ucf.edu/Code/Optical_Flow/Lucas%20Kanade.zi