Computer Vision

(Course Code: 4047)

Module-3:Lecture-3: KLT Tracking

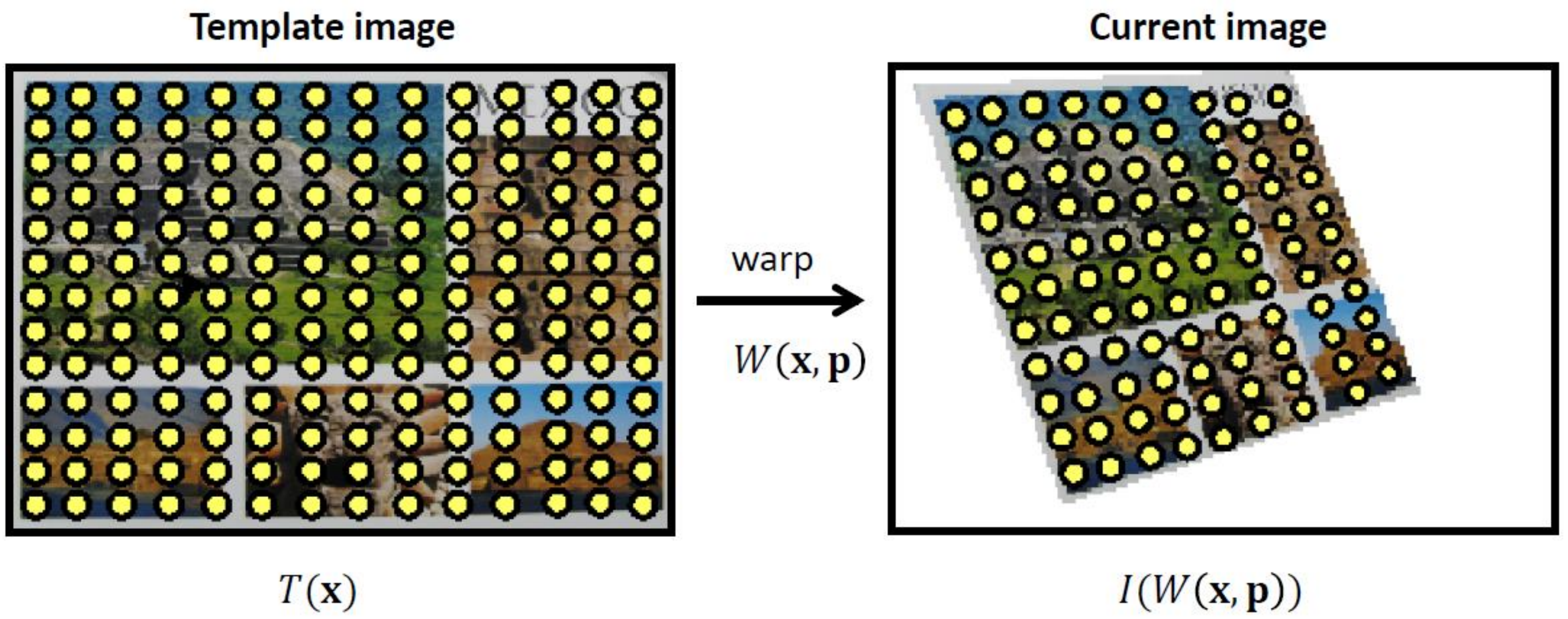
Gundimeda Venugopal, Professor of Practice, SCOPE

Template Tracking

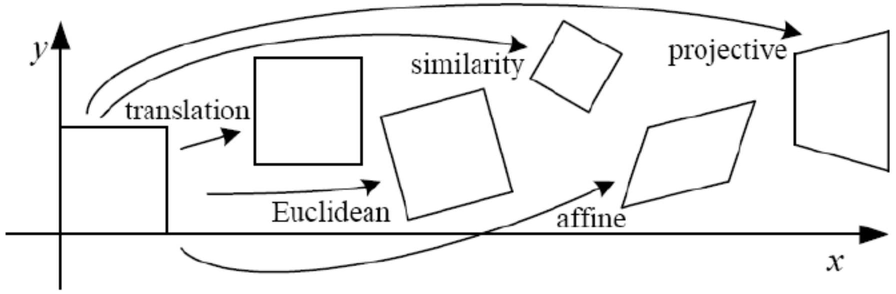
Goal: follow a template image in a video sequence by estimating the warp

Template Warping

- Given the template image $T(\mathbf{x})$
- Take all pixels from the template image $T(\mathbf{x})$ and warp them using the function $W(\mathbf{x}, \mathbf{p})$ parameterized in terms of parameters $\mathbf{p}$



Common 2D Transformations



- Translation

$$x' = x + a_1$$
$$y' = y + a_2$$

- Euclidean

$$x' = x\cos(a_3) - y\sin(a_3) + a_1$$
$$y' = x\sin(a_3) + y\cos(a_3) + a_2$$

- Affine

$$x' = a_1x + a_3y + a_5$$
$$y' = a_2x + a_4y + a_6$$

- Projective (homography)

$$x' = \frac{a_1x + a_2y + a_3}{a_7x + a_8y + 1}$$

$$y' = \frac{a_4x + a_5y + a_6}{a_7x + a_8y + 1}$$

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

$$x' = a_1x + a_2y + b_1$$
$$y' = a_3x + a_4y + b_2$$

Projective

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

Bi-quadratic

$$x' = a_1 + a_2x + a_3y + a_4x^2 + a_5y^2 + a_6xy$$
$$y' = a_7 + a_8x + a_9y + a_{10}x^2 + a_{11}y^2a_{12}xy$$

Bi-Linear

$$x' = a_1 + a_2x + a_3y + a_4xy$$
$$y' = a_5 + a_6x + a_7y + a_8xy$$

Pseudo-Perspective

$$x' = a_1 + a_2x + a_3y + a_4x^2 + a_5xy$$
$$y' = a_6 + a_7x + a_8y + a_4xy + a_5y^2$$

Displacement Models Parameterizations

Translation $x' = x + b_1$

$y' = y + b_2$

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

Rigid

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

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

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

Affine

$$x' = a_1 x + a_2 y + b_1$$

$$y' = a_3 x + a_4 y + b_2$$

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

Homogenous coordinates

Translation

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

Rigid

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

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

Affine

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

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

Displacement Models (Parameterizations)

Projective

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

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

$$W(\mathbf{x};\mathbf{p}) = \left(\frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}, \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}\right)$$

Projective

$$W(\mathbf{x};\mathbf{p}) = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ c_1 & c_2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Common 2D Transformations (using Matrices)

We denote the transformation W$(\mathbf{x}, \mathbf{p})$ and $\mathbf{p}$ the set of parameters $p = (a_1, a_2, \dots, a_n)$

- Translation

$$W(\mathbf{x},\mathbf{p}) = \begin{bmatrix} x + a_1 \\ y + a_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a_1 \\ 0 & 1 & a_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogeneous coordinates

- Euclidean

$$W(\mathbf{x},\mathbf{p}) = \begin{bmatrix} x\cos(a_3) - y\sin(a_3) + a_1 \\ x\sin(a_3) + y\cos(a_3) + a_2 \end{bmatrix} = \begin{bmatrix} \cos(a_3) & -\sin(a_3) & a_1 \\ \sin(a_3) & \cos(a_3) & a_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

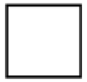
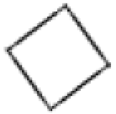
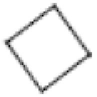
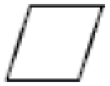
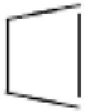
- Affine

$$W(\mathbf{x},\mathbf{p}) = \begin{bmatrix} a_1 x + a_3 y + a_5 \\ a_2 x + a_4 y + a_6 \end{bmatrix} = \begin{bmatrix} a_1 & a_3 & a_5 \\ a_2 & a_4 & a_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Projective (homography)

$$W(\tilde{\mathbf{x}},\mathbf{p}) = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Common 2D Transformations (using Matrices)

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

$$W(\mathbf{x},\mathbf{p}) = \begin{bmatrix} 1 & 0 & a_1 \\ 0 & 1 & a_2 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W(\mathbf{x},\mathbf{p}) = \begin{bmatrix} \cos(a_3) & -\sin(a_3) & a_1 \\ \sin(a_3) & \cos(a_3) & a_2 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W(\mathbf{x},\mathbf{p}) = a_4\begin{bmatrix} \cos(a_3) & -\sin(a_3) & a_1 \\ \sin(a_3) & \cos(a_3) & a_2 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W(\mathbf{x},\mathbf{p}) = \begin{bmatrix} a_1 & a_3 & a_5 \\ a_2 & a_4 & a_6 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W(\tilde{x},\mathbf{p}) = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Derivative and Gradient

- Function: $f(x)$
- Derivative: $f'(x) = \frac{df}{dx}$, where x is a scalar
- Function: $f(x_1, x_2, \ldots, x_n)$
- Gradient: $\nabla f(x_1, x_2, \ldots, x_n) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right)$

Jacobian

- $F(x_1, x_2, \ldots, x_n) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_m(x_1, x_2, \ldots, x_n) \end{bmatrix}$ is a vector-valued function

- The derivative in this case is called Jacobian $\frac{\partial F}{\partial \mathbf{x}}$:

$$\frac{\partial F}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}, \ldots, \frac{\partial f_1}{\partial x_n} \\ \vdots \\ \frac{\partial f_m}{\partial x_1}, \ldots, \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Carl Gustav Jacob Jacobi

10 December 1804—18 February 1851

- Made fundamental contributions to elliptic functions, dynamics, differential equations, and number theory.
- Jacobi was the first Jewish mathematician to be appointed professor at a German university.[2]
- In 1825 he obtained the degree of Doctor of Philosophy.
- He followed immediately with his Habilitation and at the same time converted to Christianity.
  - Now qualifying for teaching University classes, the 21 year old Jacobi lectured in 1825/26 on the theory of curves and surfaces at the University of Berlin.[4][5]
- Jacobi suffered a breakdown from overwork in 1843. He then visited Italy for a few months to regain his health.
- Jacobi died in 1851 from a smallpox infection.
- The crater Jacobi on the Moon is named after him.

Displacement-model Jacobians ∇W_p

p is a set of parameters that control the transformation

$$p = (a_1, a_2, \dots, a_n)$$

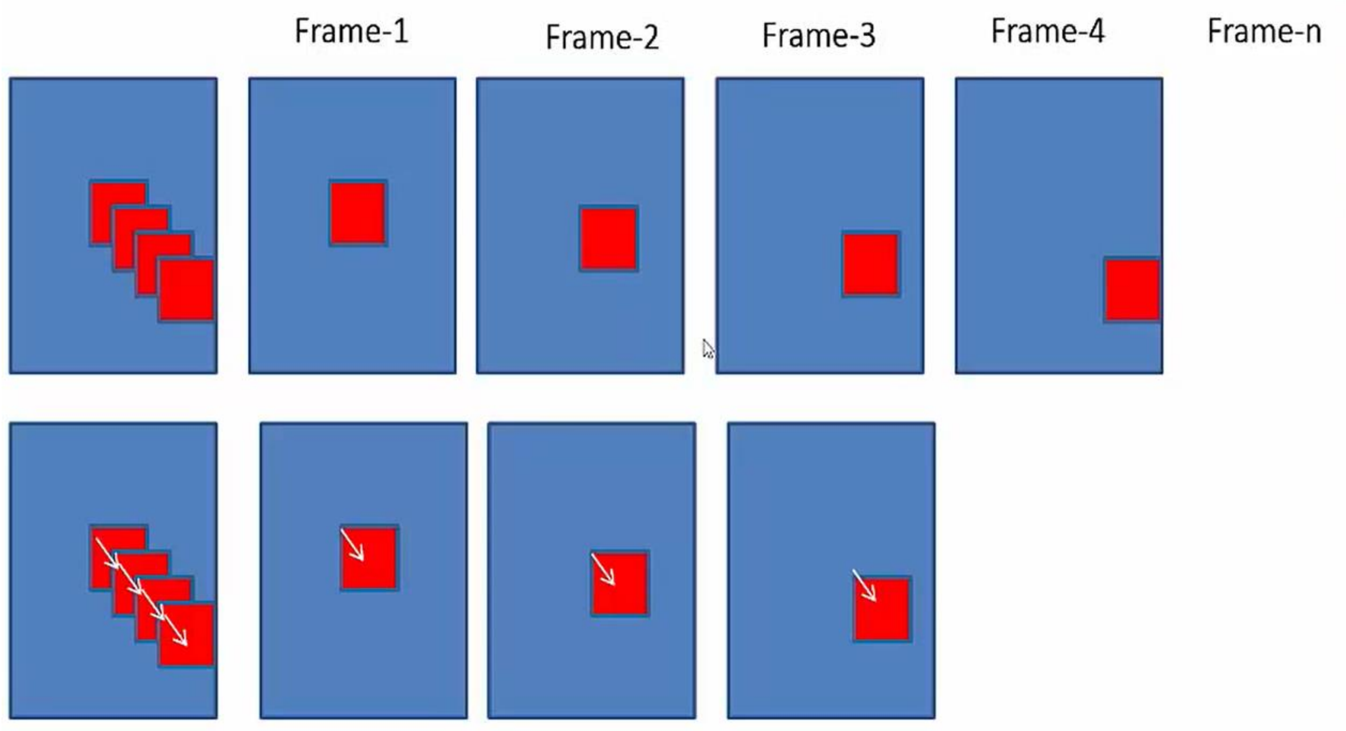
- Translation: $W(\mathbf{x}, \mathbf{p}) = \begin{bmatrix} x + a_1 \\ y + a_2 \end{bmatrix}$ $\quad \dfrac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} \dfrac{\partial W_1}{\partial a_1} & \dfrac{\partial W_1}{\partial a_2} \\ \dfrac{\partial W_2}{\partial a_1} & \dfrac{\partial W_2}{\partial a_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

- Euclidean: $W(\mathbf{x}, \mathbf{p}) = \begin{bmatrix} x\cos(a_3) - y\sin(a_3) + a_1 \\ x\sin(a_3) + y\cos(a_3) + a_2 \end{bmatrix}$ $\quad \dfrac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} 1 & 0 & -x\sin(a_3) - y\cos(a_3) \\ 0 & 1 & x\cos(a_3) - y\sin(a_3) \end{bmatrix}$

- Affine: $W(\mathbf{x}, \mathbf{p}) = \begin{bmatrix} a_1 x + a_3 y + a_5 \\ a_2 x + a_4 y + a_6 \end{bmatrix}$ $\quad \dfrac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$

KLT Tracker Intuition: Single Object Motion Tracker

The object is moving and Frame1, Frame2, Frame3, Frame4 and so on are the sequence of frames given.



Intuition

- For each frame, we are computing Optical flow for center pixel of the object.
  - Compute optical flow for center pixel of the object from frame1 to frame2 (a vector)
  - Compute optical flow for center pixel of the object from frame2 to frame3 (a vector)
  - Compute optical flow for center pixel of the object from frame3 to frame4 and so on (vector(s))
- Link all of them

This is a basic KLT Tracker (just translation)

Simple KLT Tracking 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.

Template Tracking: Problem Formulation

- The goal of template-based tracking is to find the set of warp parameters **p** such that:

$$I\big(W(\mathbf{x},\mathbf{p})\big) = T(\mathbf{x})$$

- This is solved by determining **p** that minimizes the Sum of Squared Differences:

$$SSD = \sum_{\mathbf{x}\in\mathbf{T}} \left[I\big(W(\mathbf{x},\mathbf{p})\big) - T(\mathbf{x}) \right]^2$$

- Uses the Gauss-Newton method for minimization, that is:
  - Applies a first-order approximation of the warp
  - Attempts to minimize the SSD iteratively

Problem Formulation + Derivation of the KLT algorithm

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 (Up to first order term)

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

Assumptions

- **No errors in the template image boundaries:** only the object to track appears in the template image

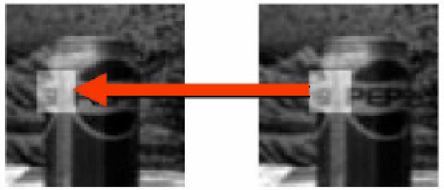
- **No occlusion**: the entire template is visible in the input image

- **Brightness constancy,**
- **Temporal consistency,**
- **Spatial coherency**

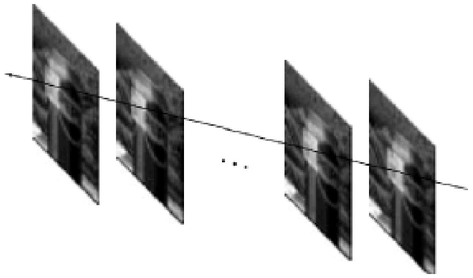
Assumptions

Assumptions:

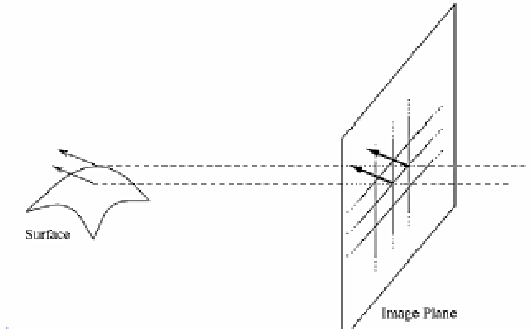
- **Brightness constancy**
  - The intensity of the pixels around the point to track does not change much between the two frames



- **Temporal consistency**
  - The motion displacement is small (1-2 pixels); however, this can be addressed using multi-scale implementations (see later)



- **Spatial coherency**
  - Neighboring pixels undergo similar motion (i.e., they all lay on the same 3D surface, i.e., no depth discontinuity)



Derivation of the KLT algorithm

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

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

$$\frac{\partial SSD}{\partial \Delta\mathbf{p}} = 0$$

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

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

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

Derivation of the KLT Algorithm

Notice that these are NOT matrix products but **pixel-wise** products!

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

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

Second moment matrix (Hessian) of the warped image

What does H look like when the warp is a pure translation?

H Matrix for Translation Motion

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

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

$$\nabla I = \begin{bmatrix} I_x & I_y \end{bmatrix} \qquad \frac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\left[\nabla I \frac{\partial W}{\partial \mathbf{p}}\right]^T = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \qquad \nabla I \frac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} I_x & I_y \end{bmatrix}$$

$$H = \sum_{\mathbf{x}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Interestingly, this is a Harris Detector (Corner detection)

KLT Algorithm

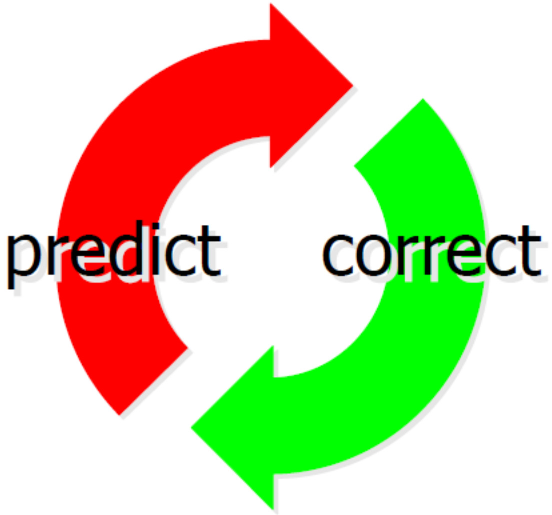
$$\Rightarrow \Delta\mathbf{p} = H^{-1} \sum_{\mathbf{x}\in\mathrm{T}} \left[\nabla I \frac{\partial W}{\partial \mathbf{p}}\right]^{\mathrm{T}} \left[T(\mathbf{x}) - I\big(W(\mathbf{x},\mathbf{p})\big)\right]$$

1. Warp $I(\mathbf{x})$ with $W(\mathbf{x},\mathbf{p}) \rightarrow I\big(W(\mathbf{x},\mathbf{p})\big)$
2. Compute the error: subtract $I\big(W(\mathbf{x},\mathbf{p})\big)$ from $T(\mathbf{x})$
3. Compute **warped** gradients: $\nabla I = [I_x, I_y]$, evaluated at $W(\mathbf{x},\mathbf{p})$
4. Evaluate the Jacobian of the warping: $\frac{\partial W}{\partial \mathbf{p}}$
5. Compute steepest descent: $\nabla I \frac{\partial W}{\partial \mathbf{p}}$
6. Compute Inverse Hessian: $H^{-1} = \left[\sum_{\mathbf{x}\in\mathrm{T}} \left[\nabla I \frac{\partial W}{\partial \mathbf{p}}\right]^{\mathrm{T}} \left[\nabla I \frac{\partial W}{\partial \mathbf{p}}\right]\right]^{-1}$
7. Multiply steepest descend with error: $\sum_{\mathbf{x}\in\mathrm{T}} \left[\nabla I \frac{\partial W}{\partial \mathbf{p}}\right]^{\mathrm{T}} \left[T(\mathbf{x}) - I\big(W(\mathbf{x},\mathbf{p})\big)\right]$
8. Compute $\Delta\mathbf{p}$
9. Update parameters: $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$
10. Repeat until $\Delta\mathbf{p} < \boldsymbol{\varepsilon}$

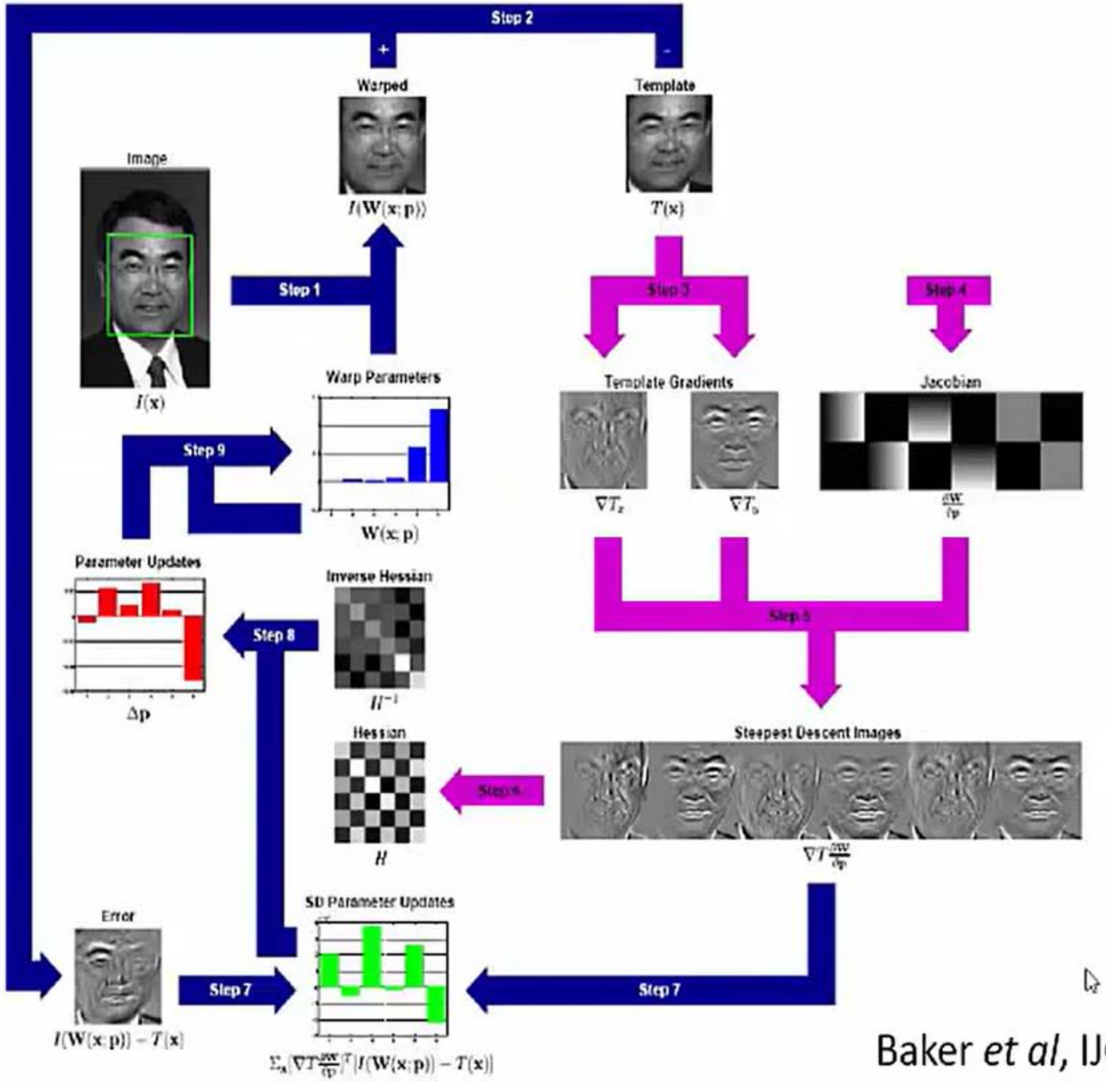
KLT predict-correct cycle

Lucas-Kanade follows a **predict-correct cycle**

- A **prediction** $I(W(\mathbf{x}, \mathbf{p}))$ of the warped image is computed from an initial estimate
- The **correction** parameter $\Delta\mathbf{p}$ is computed as a function of the error $T(\mathbf{x}) - I(W(\mathbf{x}, \mathbf{p}))$ between the prediction and the template
- The larger this error, the larger the correction applied



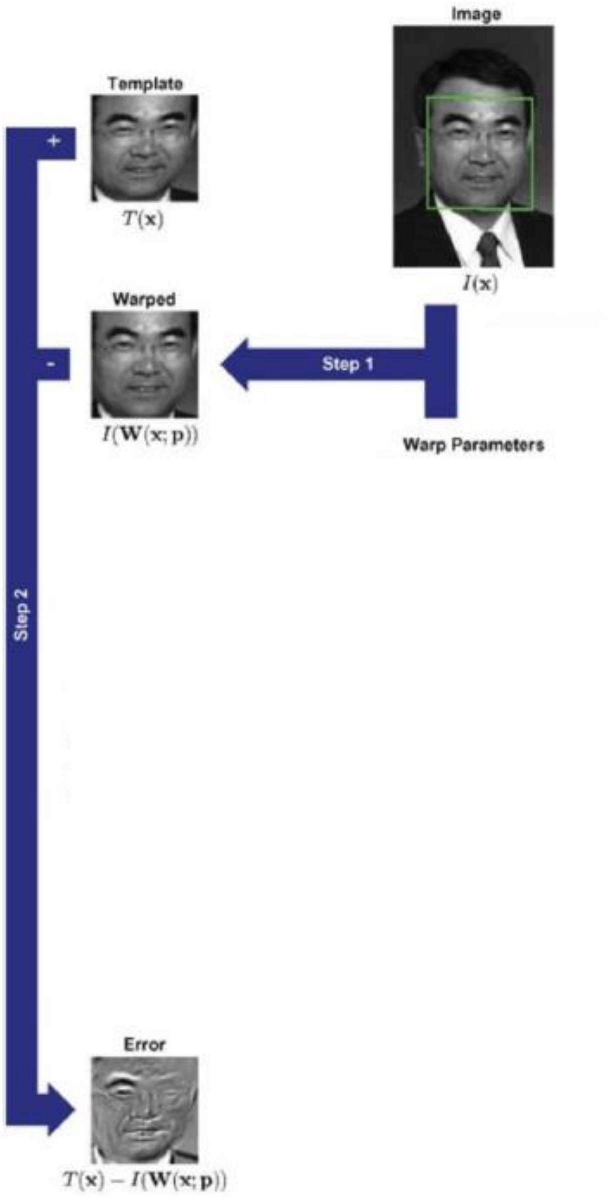
Finding Alignment



Baker *et al*, IJCV, 2004.

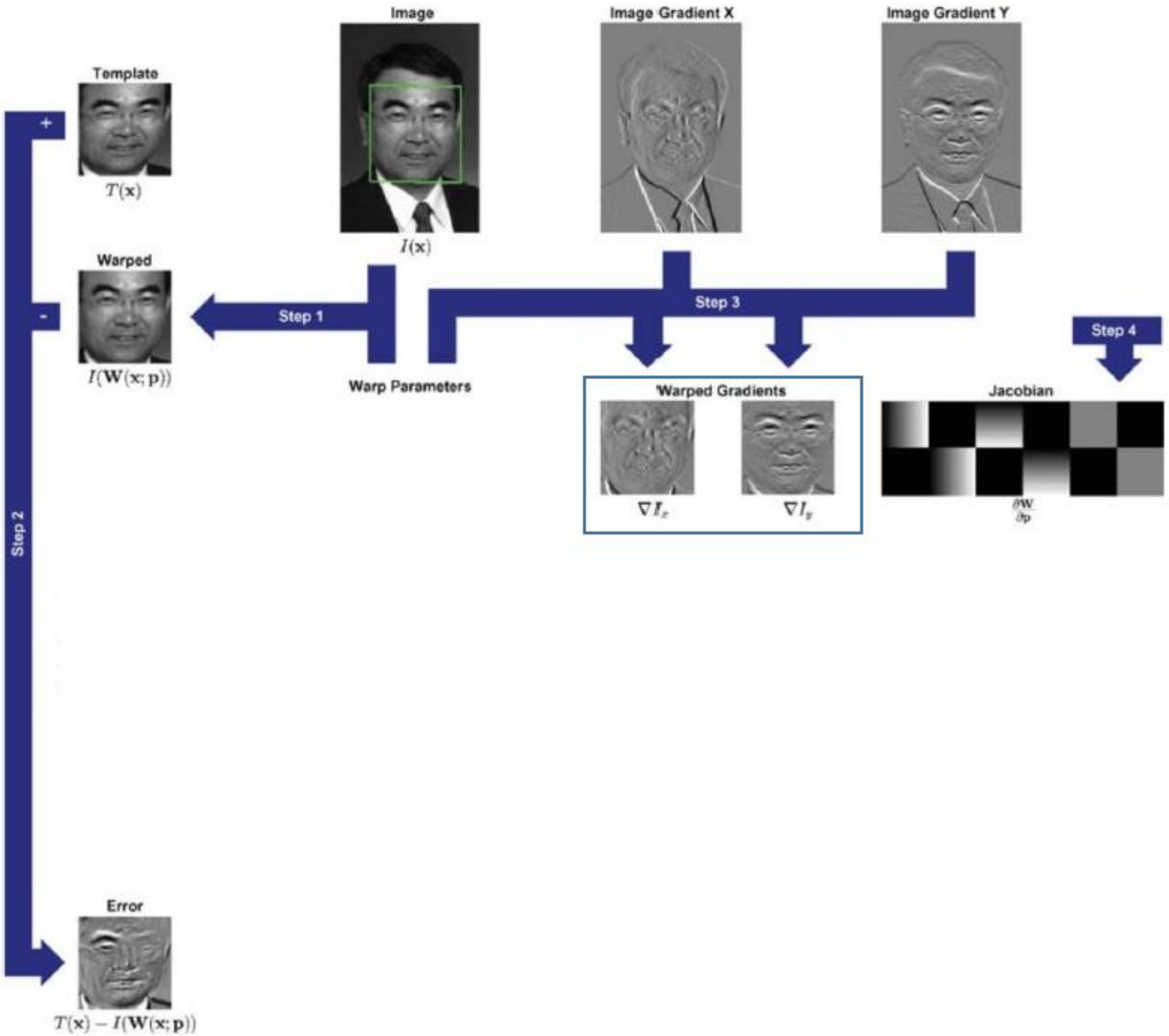
KLT Algorithm Steps (1, 2)

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



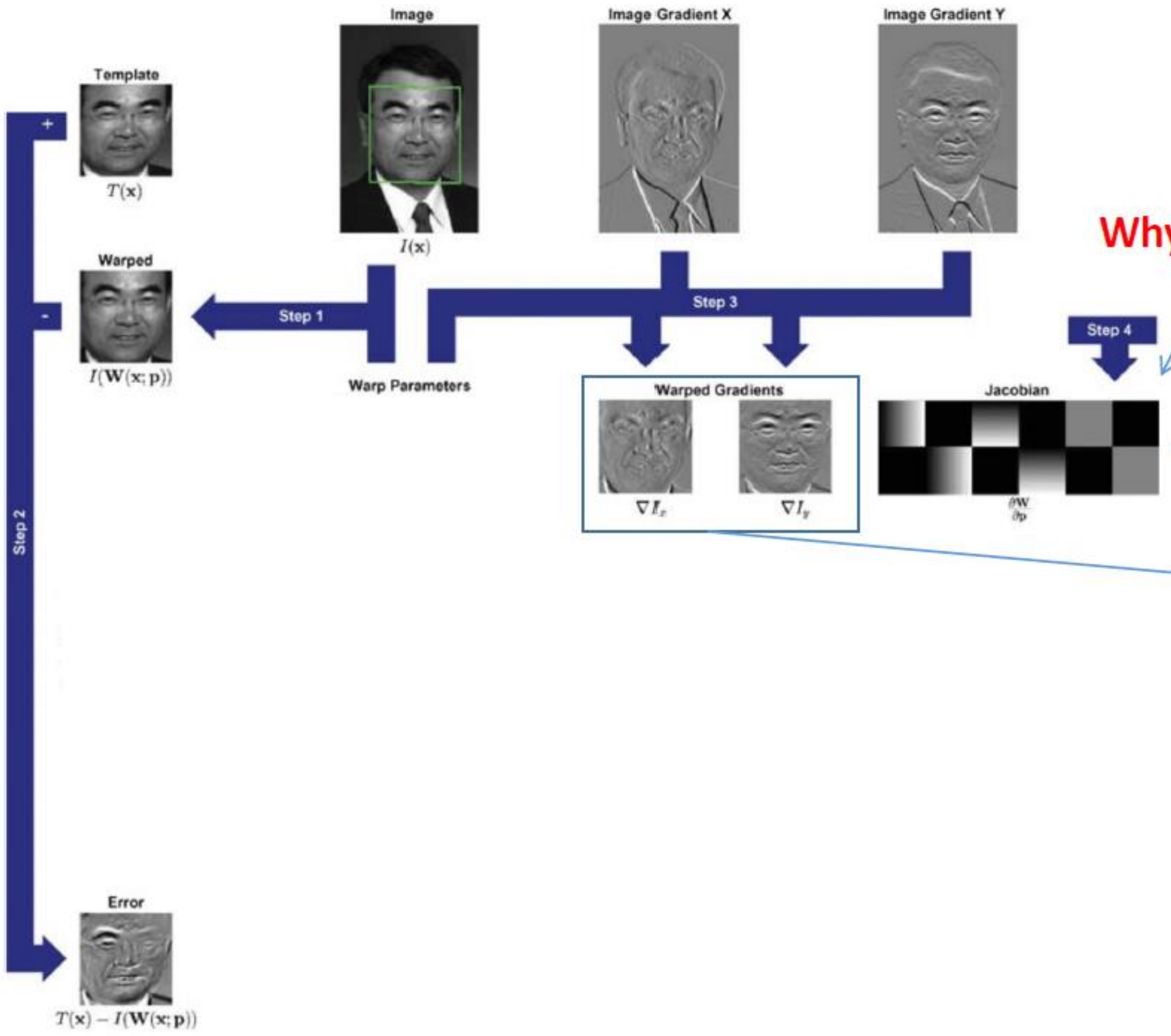
KLT Algorithm steps (3,4)

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



KLT Algorithm steps (3,4)

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



Why does it look like that?

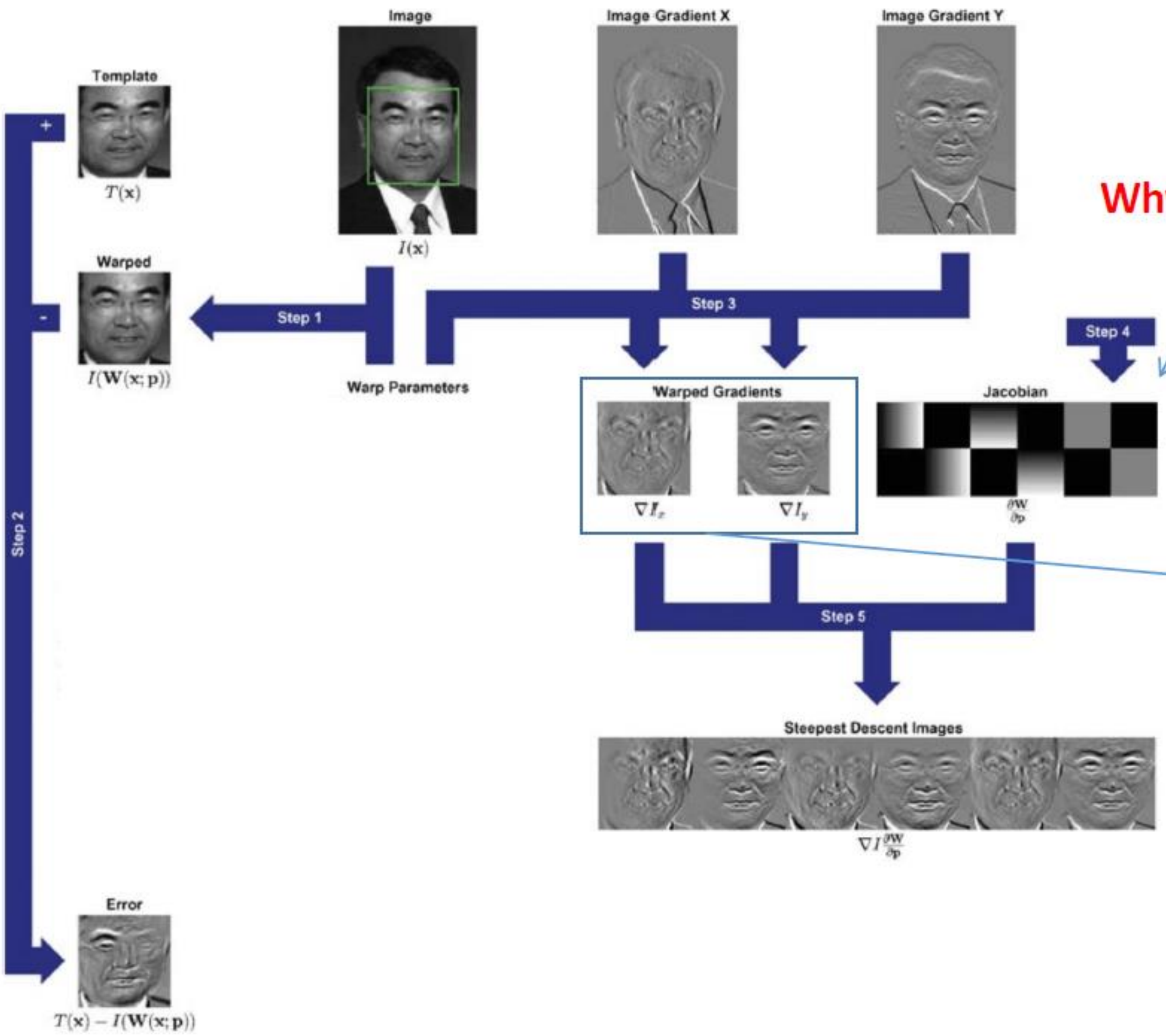
$$\frac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$$

What is the size?

$2n \times 6n$

$n \times 2n$

KLT Algorithm steps (5)



Why does it look like that? (slide 48)

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

What is the size?

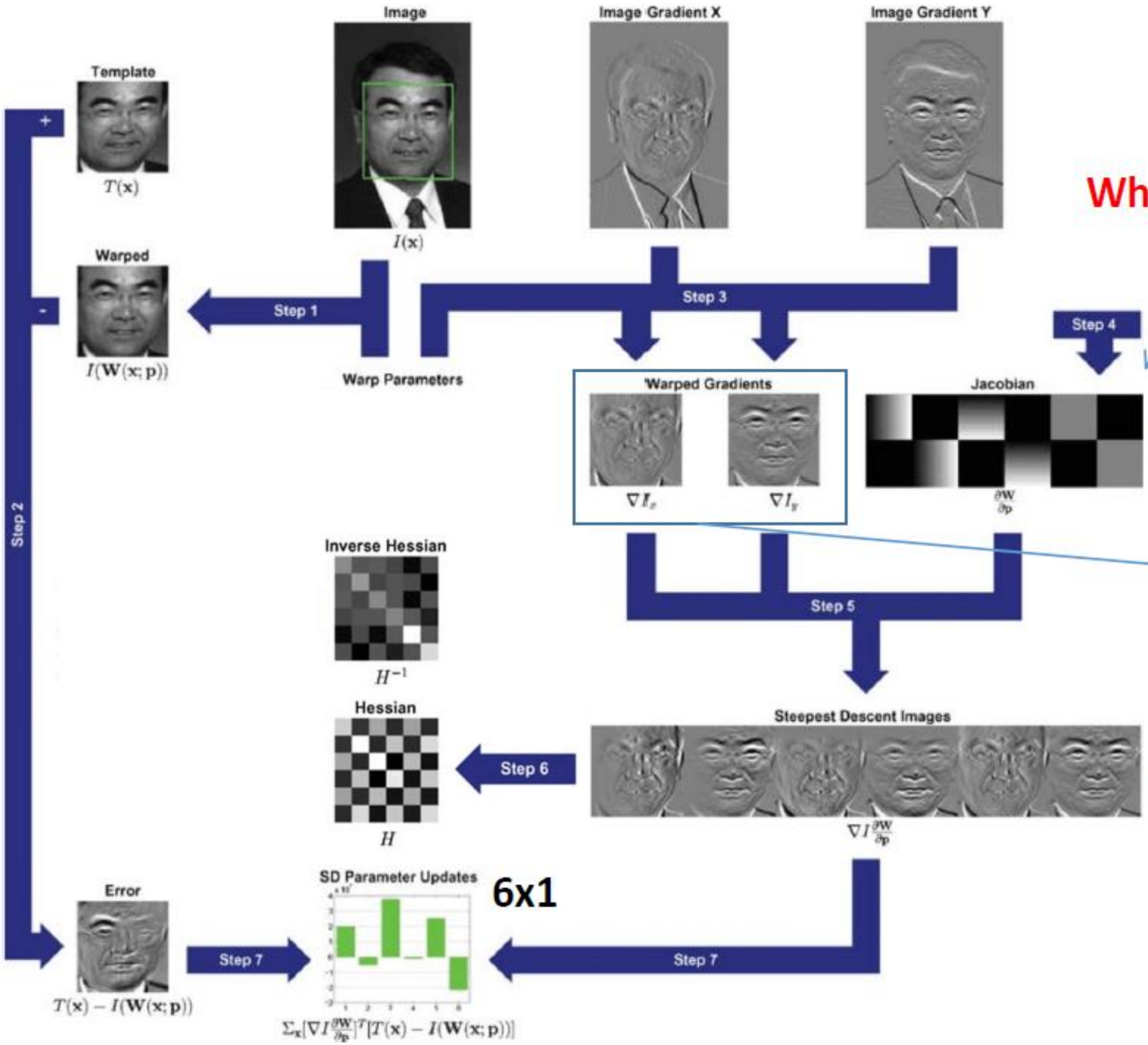
$2n \times 6n$

$n \times 2n$

What's its size?

$n \times 6n$

KLT Algorithm steps (6, 7)



6x1

Why does it look like that? (slide 48)

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

What is the size?

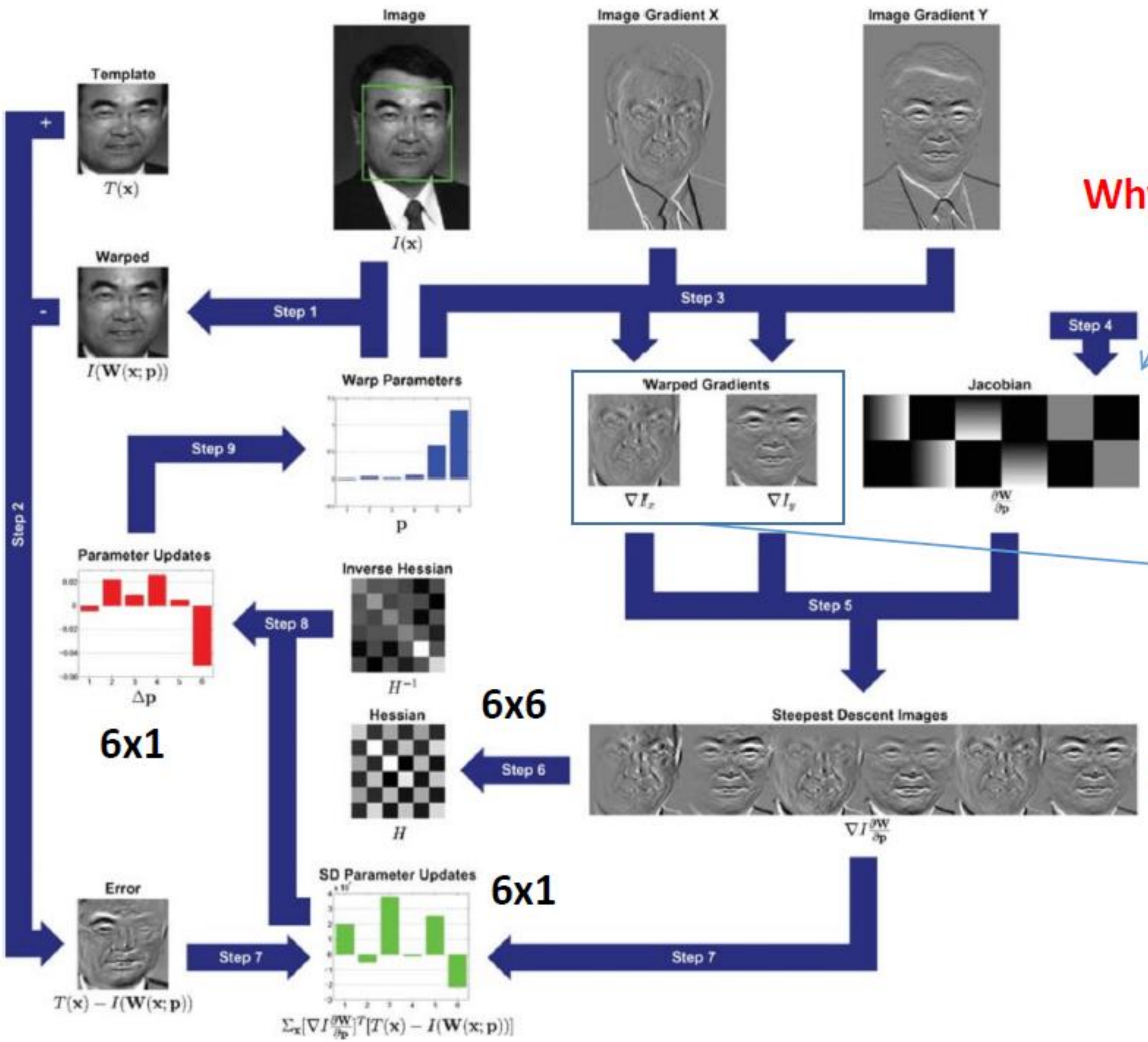
$2n \times 6n$

$n \times 2n$

What's its size?

$n \times 6n$

KLT Algorithm steps (8,9)



Why does it look like that? (slide 48)

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

What is the size?

$$2n \times 6n$$

$$n \times 2n$$

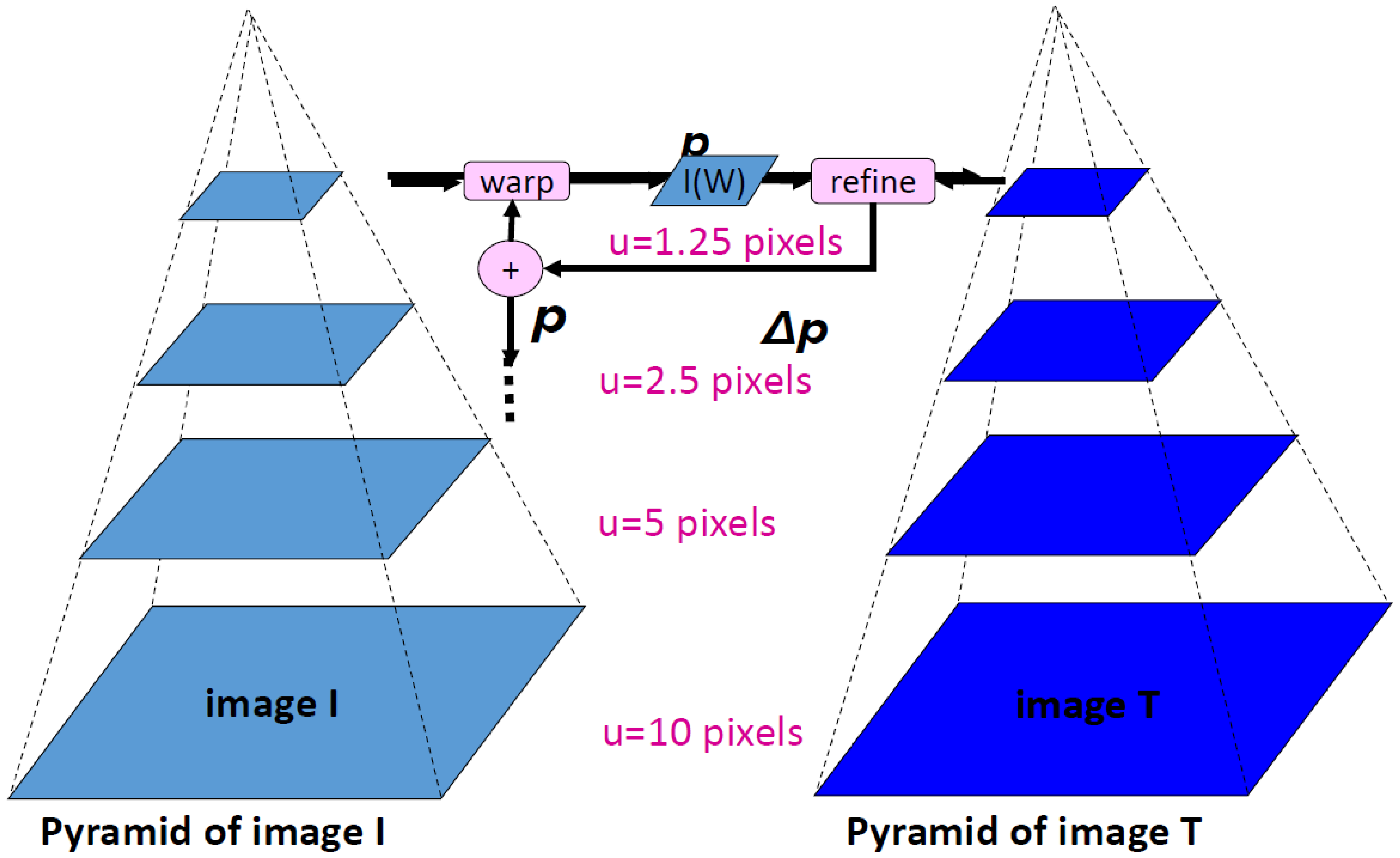
What's its size?

$$n \times 6n$$

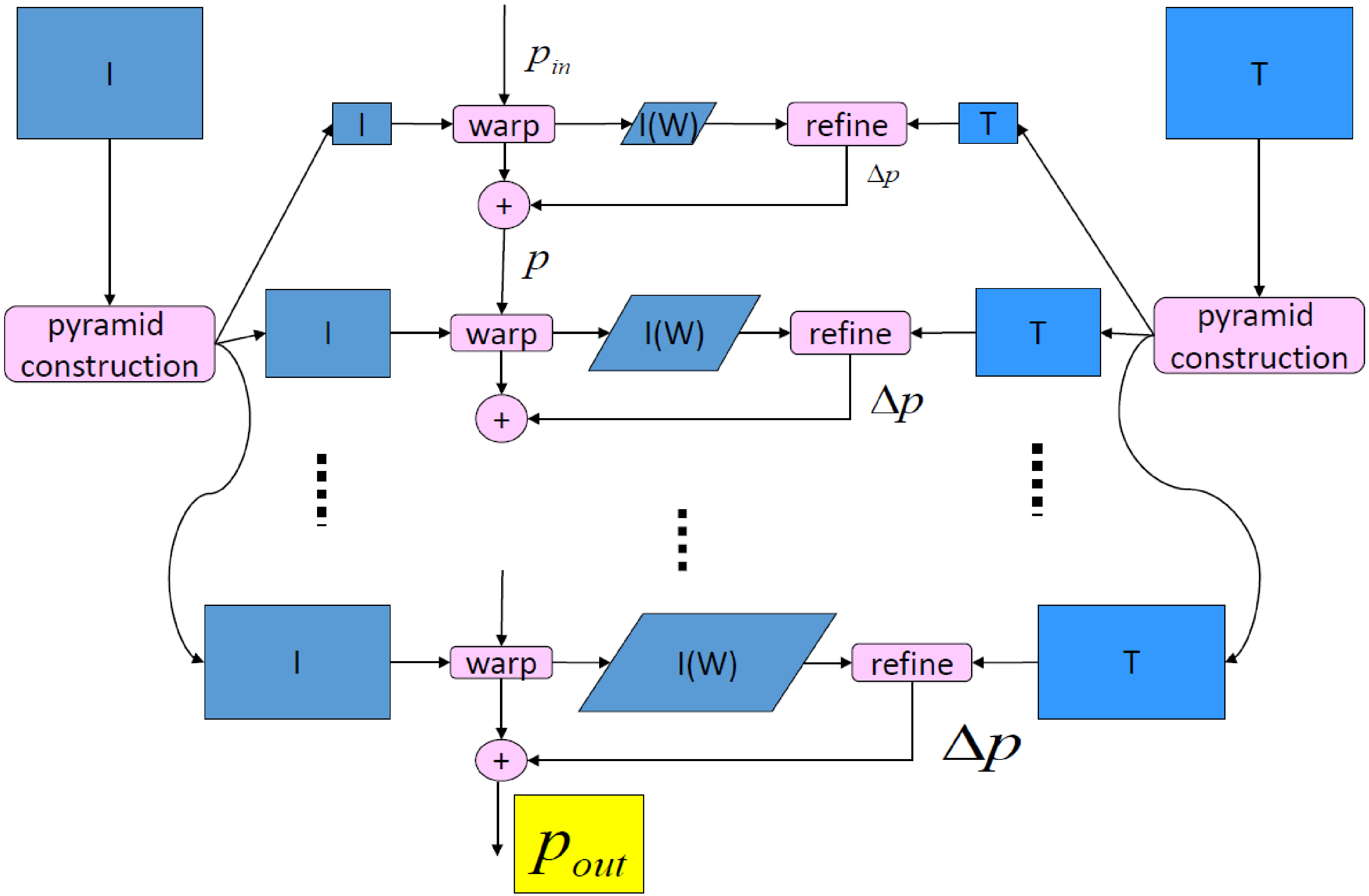
KLT Algorithm: Discussion

- How to get the initial estimate **p**?
- When does the Lucas-Kanade fail?
  - If the initial estimate is too far, then the linear approximation does not longer hold -> solution?
    - Pyramidal implementations (see next slide)
- Other problems:
  - Deviations from the mathematical model: object deformations, illumination changes, etc.
  - Occlusions
  - Due to these reasons, tracking may drift -> solution?
    - Update the template with the last image

Coarse to Fine Estimation

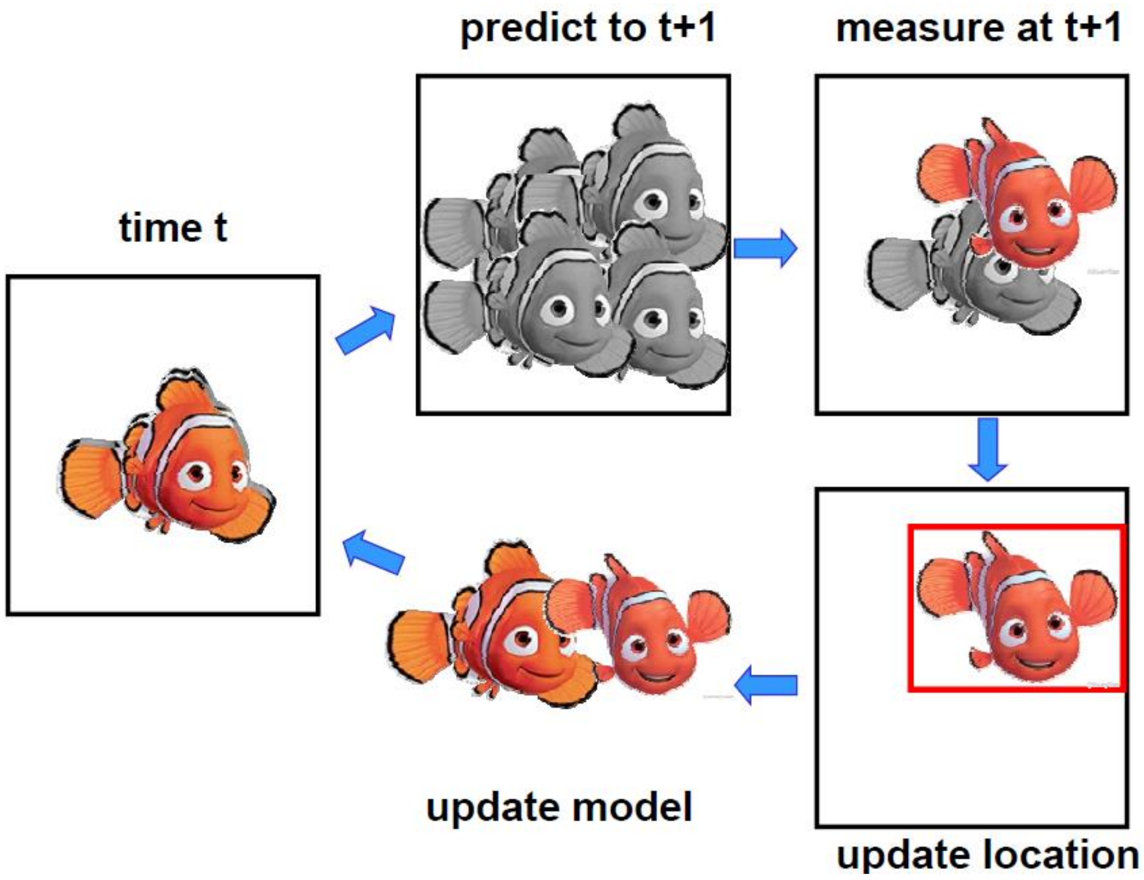


Coarse to Fine Estimation



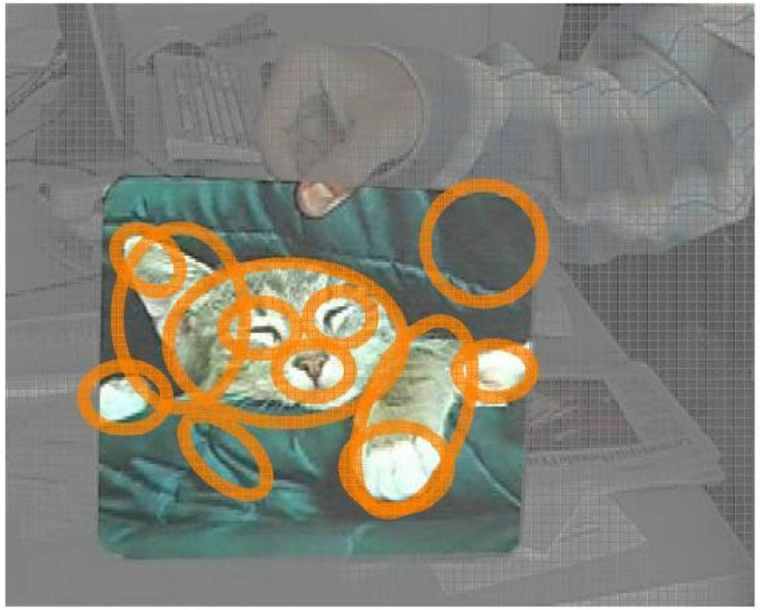
Generalisation of KLT

- The same concept (predict/correct) can be applied to tracking of 3D object (in this case, what is the transformation to estimate? What is the template?)
- In order to deal with wrong prediction, it can be implemented in a **Particle-Filter** fashion (using multiple hypotheses that need to be validated)

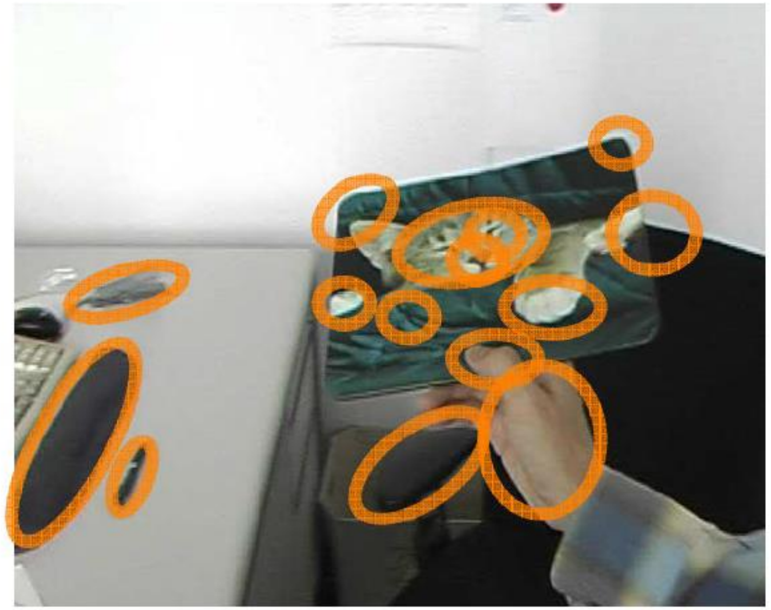


Tracking by detection of local image features (1)

- Step 1: Keypoint detection and matching
  - invariant to scale, rotation, or perspective



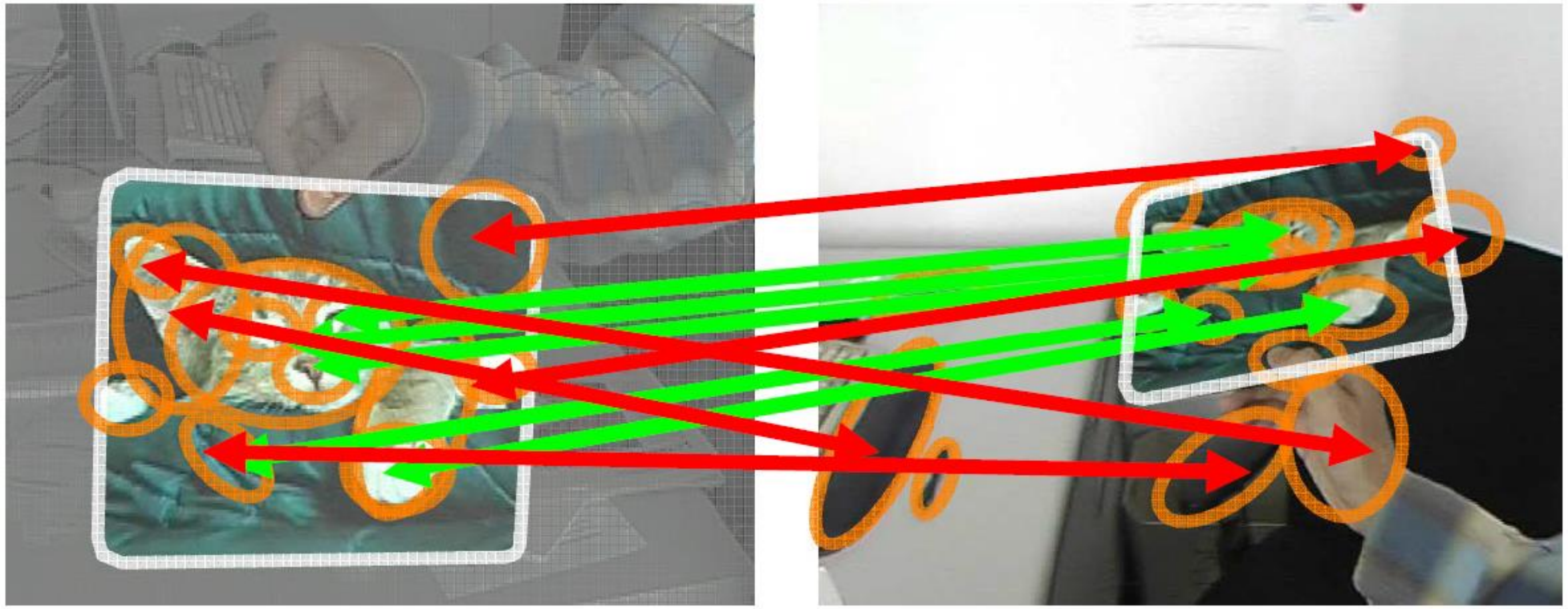
Template image with the object to detect



Current test image

Tracking by detection of local image features (2)

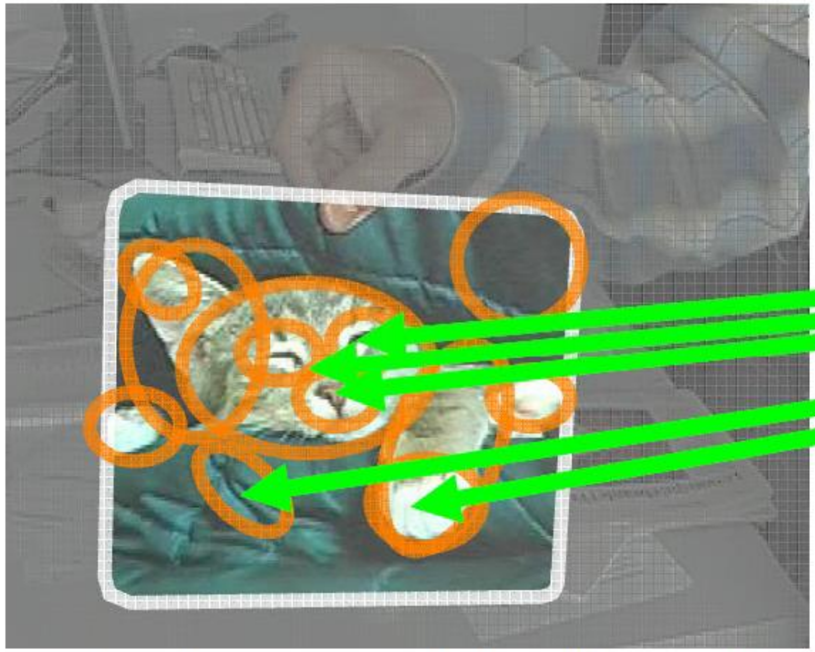
- Step 1: Keypoint detection and matching
  - invariant to scale, rotation, or perspective



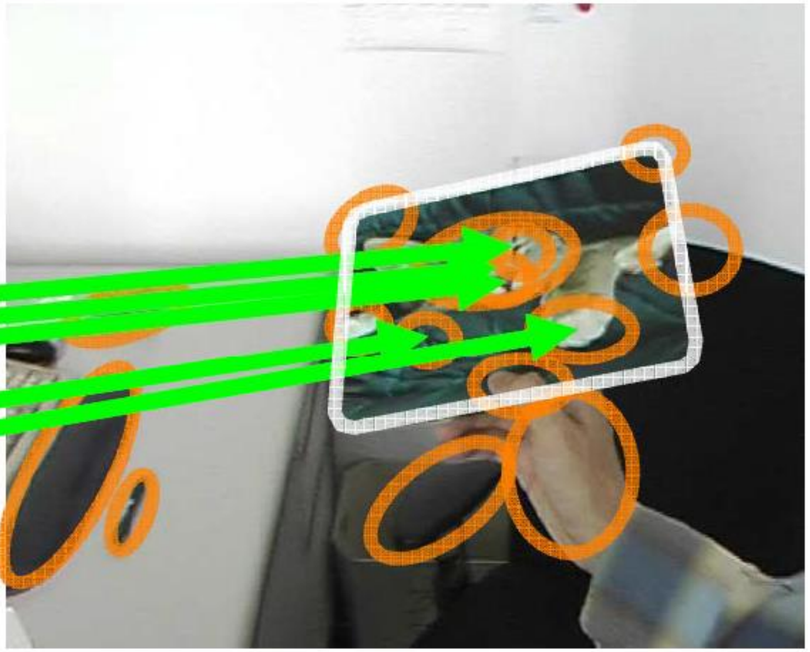
Template image with the object to detect   Current test image

Tracking by detection of local image features (3)

- Step 1: Keypoint detection and matching
  - invariant to scale, rotation, or perspective
- Step 2: Geometric verification (RANSAC) (e.g., 4-point RANSAC for planar objects, or 5 or 8-point RANSAC for 3D objects)



Template image with the object to detect



Current test image

References

- UCF Computer Vision Channel: Lecture 10 – KLT – 2014
- Image analysis: motion: feature tracking by Hany Farid, Professor at UC Berkeley
- https://rpg.ifi.uzh.ch/docs/teaching/2020/11_tracking.pdf