University of Zurich[UZH]

ROBOTICS & PERCEPTION GROUP

Vision Algorithms for Mobile Robotics

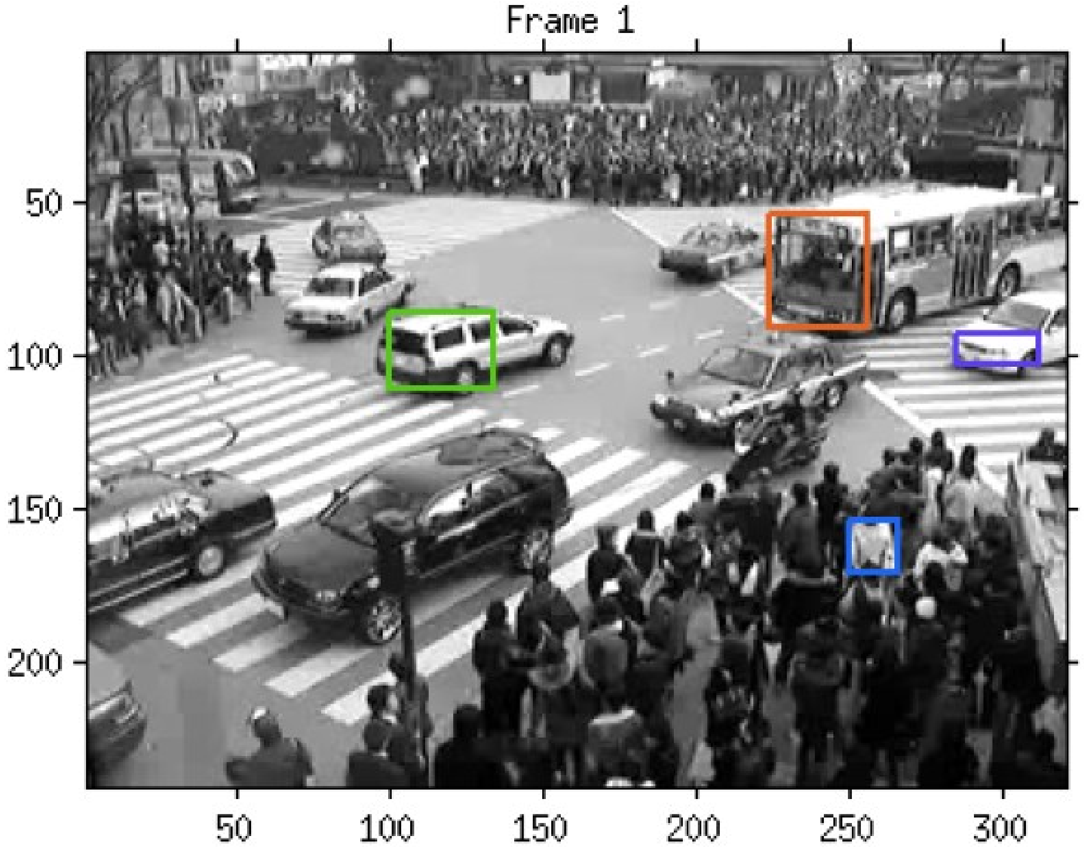
Lecture 11
Tracking

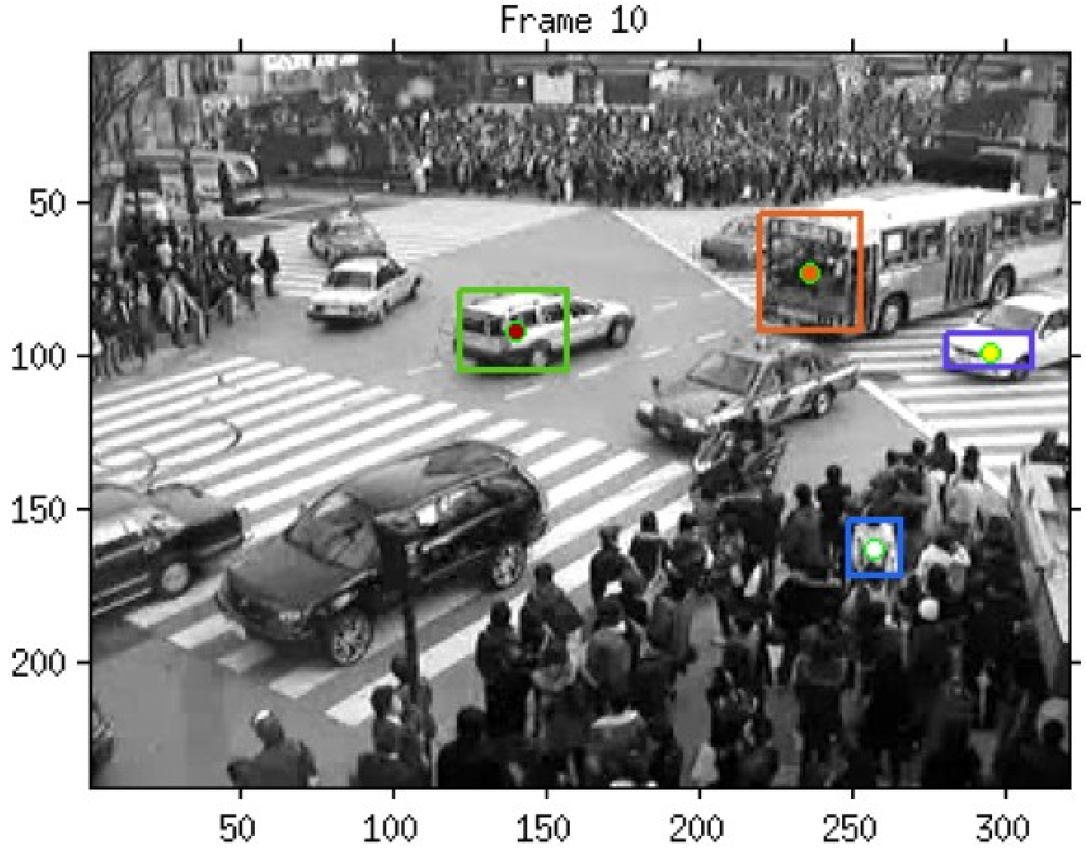
Davide Scaramuzza

http://rpg.ifi.uzh.ch

Lab Exercise – Today

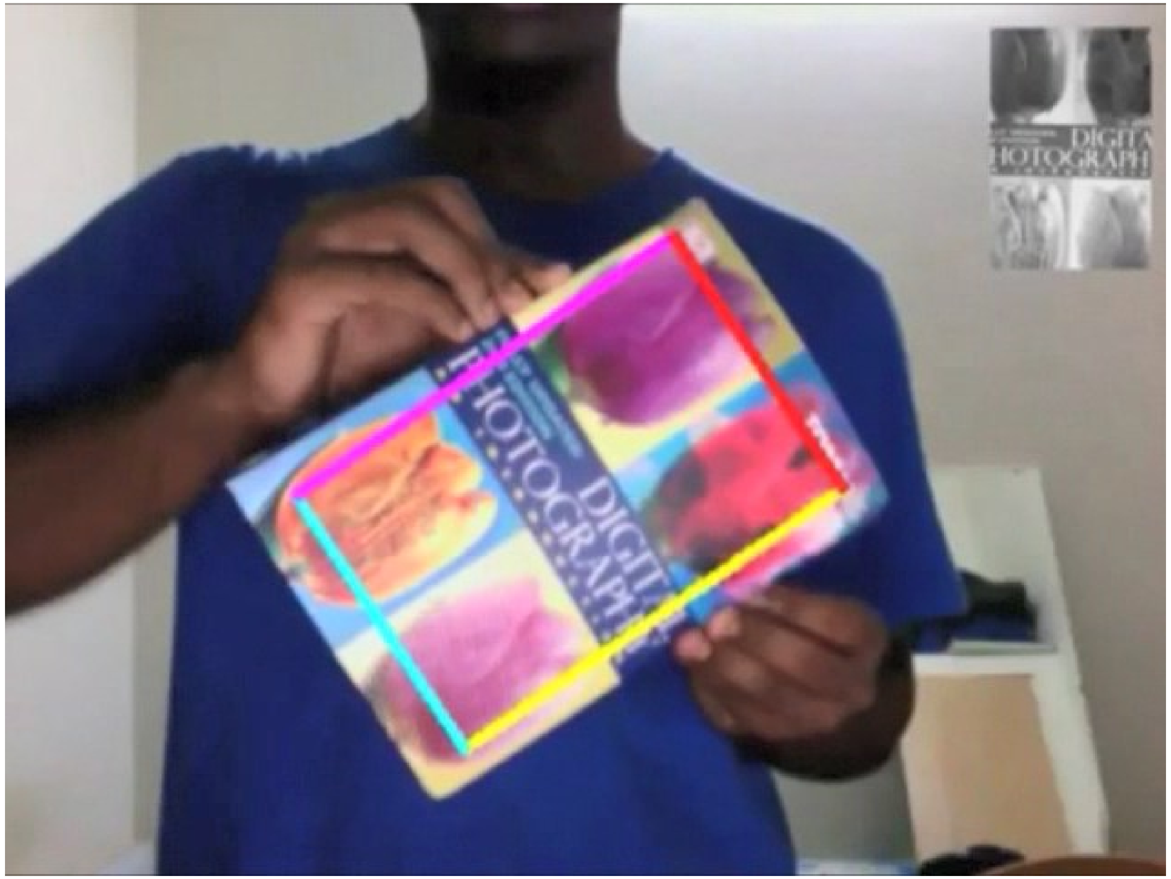
Implement the Kanade-Lucas-Tomasi (KLT) tracker





Template tracking

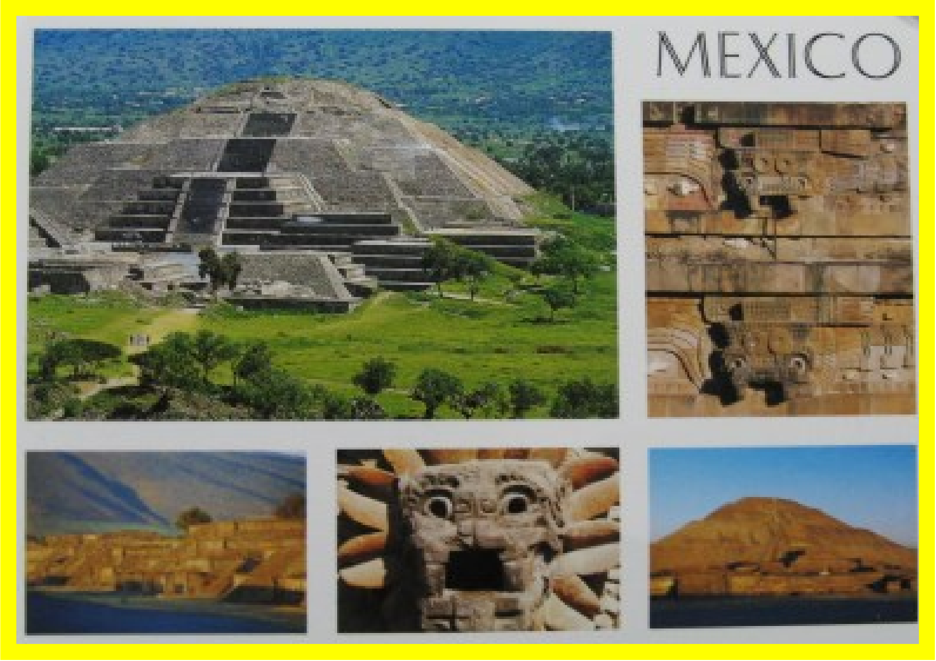
Goal: **follow** a template image in a video sequence



Problem Formulation

Goal: estimate the transformation W (warp) between a template T and the current image I

Template image T



$T(\mathbf{x})$

warp

$W(\mathbf{x}, \mathbf{p})$

Current image I

Common 2D Transformations (recall Lecture 03, slides 36-37)

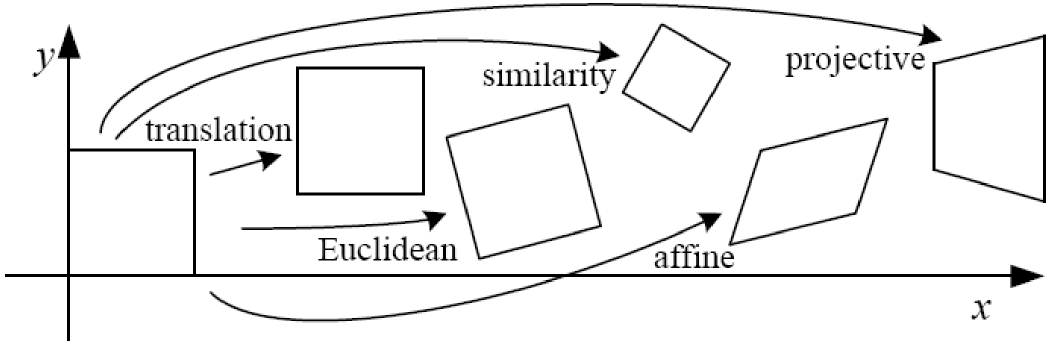
We denote the transformation $\mathrm{W}(\mathbf{x}, \mathbf{p})$ and **p** the set of parameters $\mathbf{p} = (a_1, a_2, \dots, a_n)$

- 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
$$x' = a_1 x + a_3 y + a_5$$
$$y' = a_2 x + a_4 y + a_6$$

- Projective (homography)
$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$
$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$



Two possible approaches

- Indirect methods (i.e., feature based)
  - ✓ Can cope with large frame-to-frame motions (large basin of convergence)
  - ✗ Slow due to costly feature extraction, matching, and outlier removal (e.g., RANSAC)

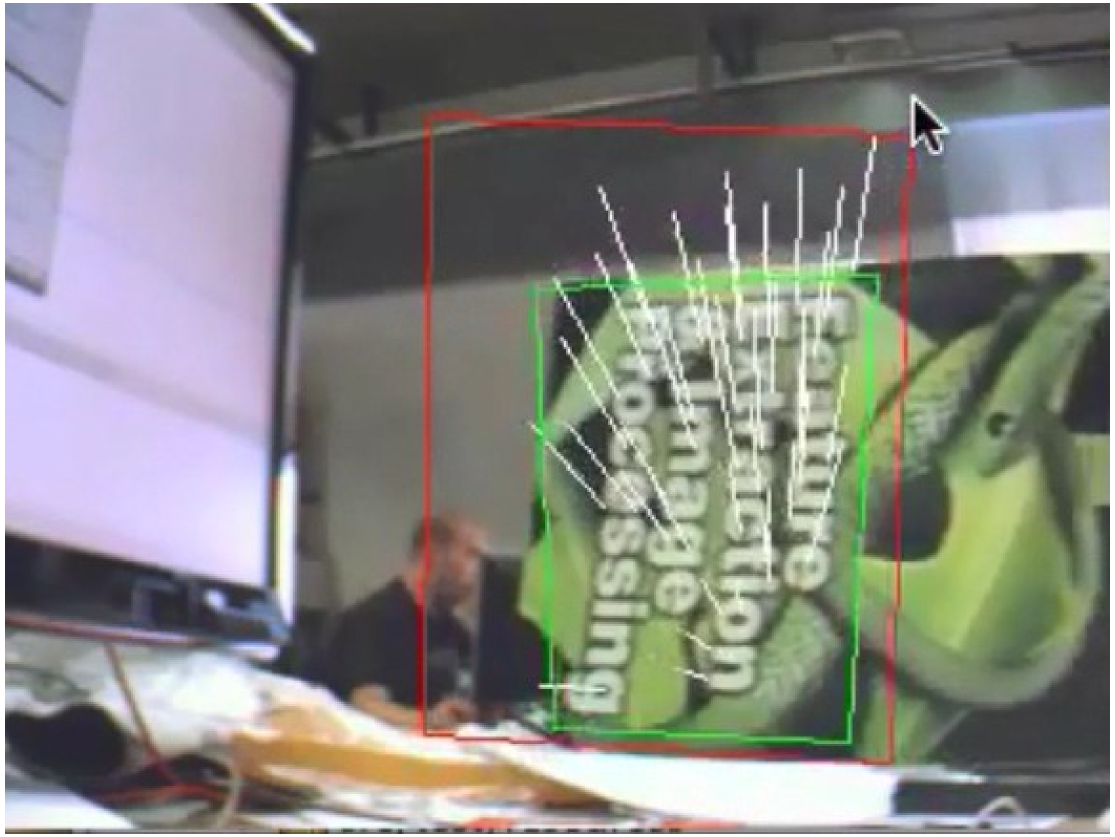
- Direct methods
  - ✓ All information in the image can be exploited (higher accuracy, higher robustness to motion blur and weak texture (i.e., weak gradients))
  - ✓ Increasing the camera frame-rate reduces computational cost per frame (no RANSAC needed)
  - ✗ Very sensitive to intial value $\rightarrow$ limited frame-to-frame motion (small basin of convergence)

Indirect methods work by detecting and matching features (i.e., feature based)

1. Detect and match features that are invariant to scale, rotation, view point changes (e.g., SIFT)
2. Geometric verification (RANSAC) (e.g., 4-point RANSAC for planar objects, or 5 or 8-point RANSAC for 3D objects)
3. Refine estimate by minimizing the sum of squared reprojection errors between the observed feature $\boldsymbol{f}^i$ in the current image and the warped corresponding feature $W(\mathbf{x}^i, \mathbf{p})$ from the template

$$\mathbf{p} = argmin_{\mathbf{p}} \sum_{i=1}^{N} \left\| W(\mathbf{x}^i, \mathbf{p}) - \boldsymbol{f}^i \right\|^2$$

- **Pros**: can cope with **large frame-to-frame motion** and **strong illumination changes**
- **Cons**: **computationally expensive**

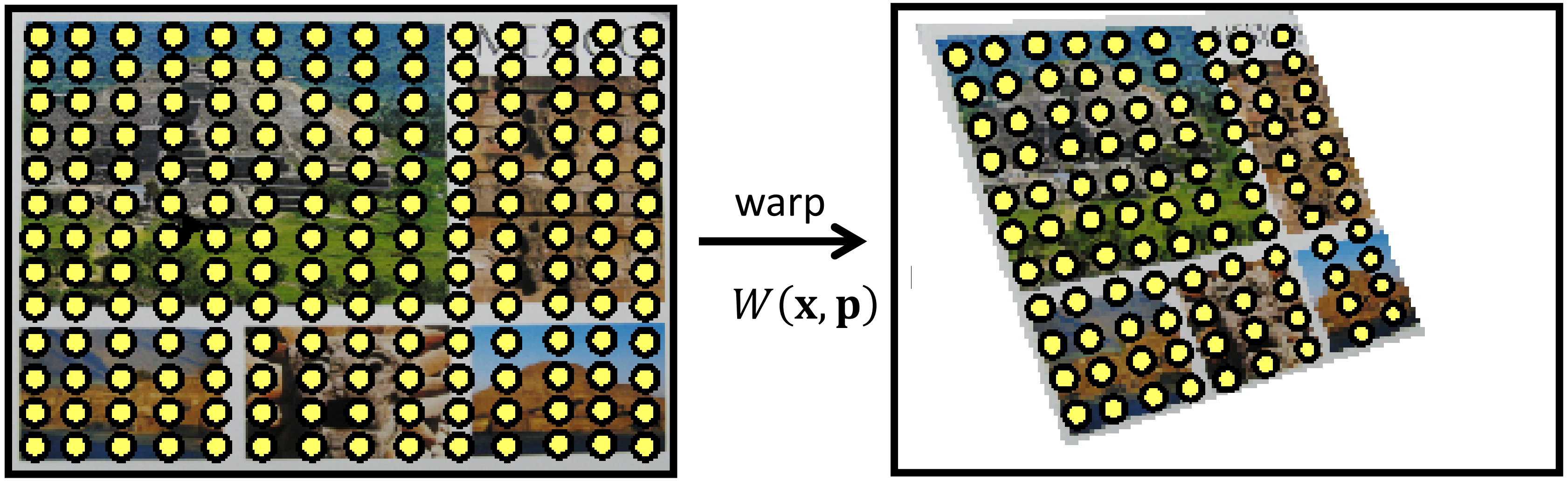


Direct methods work by directly processing pixel intensities (i.e. without features)

Goal: estimate the parameters **p** of the transformation $W(\mathbf{x}, \mathbf{p})$ that minimize the Sum of Squared Differences:

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

Template image T | **Current image** I



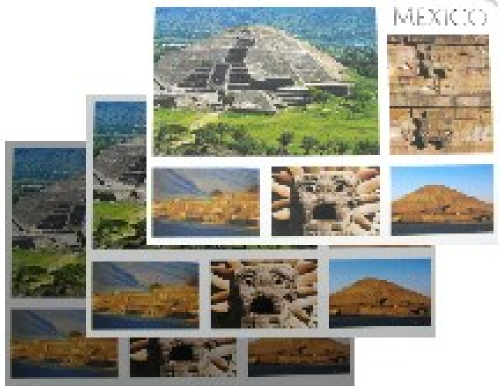
$T(\mathbf{x})$ | $I(W(\mathbf{x}, \mathbf{p}))$

* Every yellow dot in this image denotes a pixel

Assumptions

- **Brightness constancy**

  The intensity of the pixels to track does not change much over consecutive frames → **It does not cope with strong illumination changes**



- **Temporal consistency**

  Small frame-to-frame motion (1-2 pixels).

  → **It does not cope with large frame-to-frame motion.** However, this can be addressed using coarse-to-fine multi-scale implementations (see later)

- **Spatial coherency**

  All pixels in the template undergo the same transformation (i.e., they all lay on the same 3D surface)
  → **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

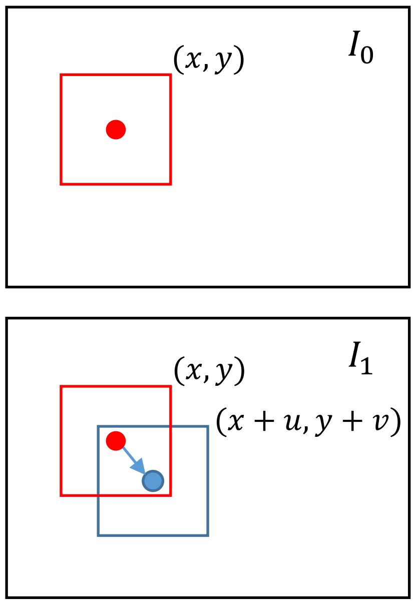
The Kanade-Lucas-Tomasi (KLT) tracker

- Simplified case: pure translation
- General case

Lucas, Kanade, *An iterative image registration technique with an application to stereo vision.* Proceedings of Imaging Understanding Workshop, 1981. PDF.
Tomasi, Kanade, *Detection and Tracking of Point Features*, Carnegie Mellon University Technical Report CMU-CS-91-132, 1991. PDF.
Baker, Matthews, Lucas-Kanade 20 Years On: A Unifying Framework, International Journal of Computer Vision, 2004. PDF.

KLT tracking applied to pure translation

Consider the reference patch centered at (x, y) in image I_0 and the shifted patch centered at $(x + u, y + v)$ in image I_1. The patch has size Ω. We want to find the motion vector (u, v) that minimizes the Sum of Squared Differences (SSD):



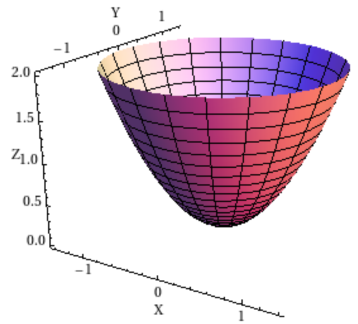
$$SSD(u, v) = \sum_{x,y \in \Omega} \left(I_0(x, y) - \underbrace{I_1(x + u, y + v)}\right)^2$$

$$I_1(x + u, y + v) \cong I_1(x, y) + I_x u + I_y v$$

$$\Rightarrow SSD(u, v) \cong \sum_{x,y \in \Omega} \left(I_0(x, y) - I_1(x, y) - I_x u - I_y v\right)^2$$

$$\Rightarrow SSD(u, v) \cong \sum_{x,y \in \Omega} \left(\Delta I - I_x u - I_y v\right)^2$$

This is a simple quadratic function in two variables (u, v)



KLT tracking applied to pure translation

$$\Rightarrow SSD(u, v) \cong \sum_{x,y \in \Omega} (\Delta I - I_x u - I_y v)^2$$

To minimize it, we differentiate it with respect to (u, v) and equate it to zero:

$$\frac{\partial SSD}{\partial u} = 0 \,, \; \frac{\partial SSD}{\partial v} = 0$$

$$\frac{\partial SSD}{\partial u} = 0 \;\Rightarrow\; -2 \sum I_x (\Delta I - I_x u - I_y v) = 0$$

$$\frac{\partial SSD}{\partial v} = 0 \;\Rightarrow\; -2 \sum I_y \left(\Delta I - I_x u - I_y v\right) = 0$$

KLT tracking applied to pure translation

$$\sum I_x(\Delta I - I_x u - I_y v) \quad = 0$$

$$\sum I_y(\Delta I - I_x u - I_y v) \quad = 0$$

- Linear system of two equations in two unknowns
- We can write them in matrix form:

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

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x \Delta I \\ \sum I_y \Delta I \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}^{-1} \begin{bmatrix} \sum I_x \Delta I \\ \sum I_y \Delta I \end{bmatrix}$$

KLT tracking applied to pure translation

$$\sum I_x(\Delta I - I_x u - I_y v) = 0$$

$$\sum I_y(\Delta I - I_x u - I_y v) = 0$$

- Linear system of two equations in two unknowns
- We can write them in matrix form:

Haven't we seen this matrix already?

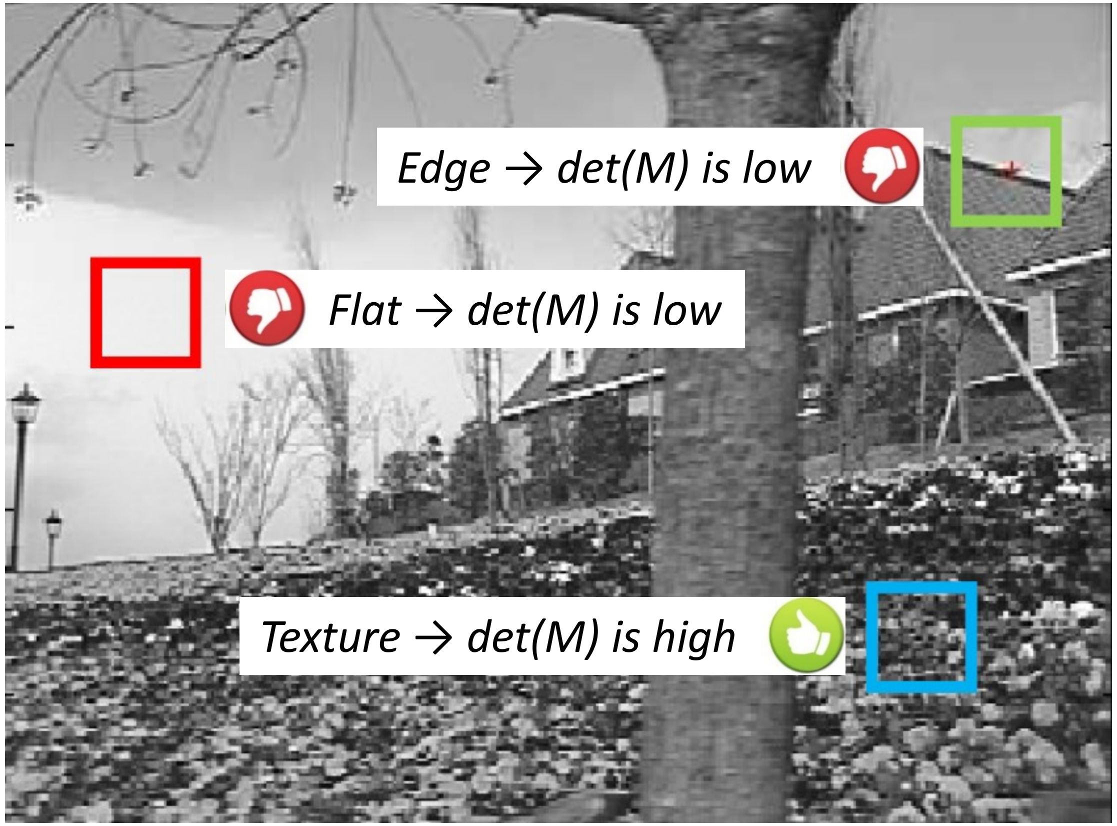
This is the M matrix of the Harris detector (Lecture 05)

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x \Delta I \\ \sum I_y \Delta I \end{bmatrix} \Rightarrow \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}^{-1} \begin{bmatrix} \sum I_x \Delta I \\ \sum I_y \Delta I \end{bmatrix}$$

KLT tracking applied to pure translation

For M to be invertible, $\det(M)$ should be non zero, which means that its eigenvalues should be large (i.e., not a flat region, not an edge) $\rightarrow$ in practice, it **should be a corner or more generally contain texture**!

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$



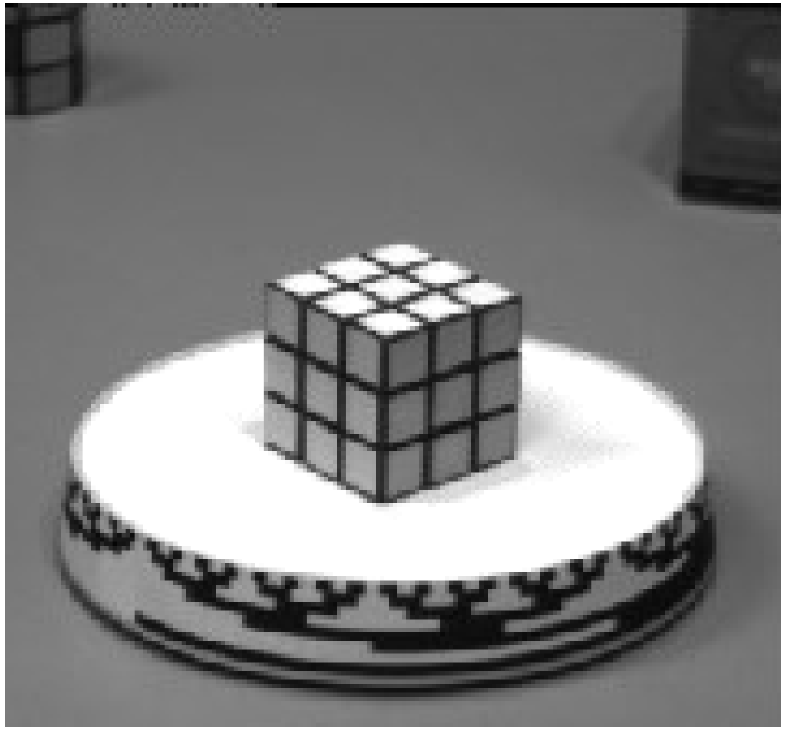
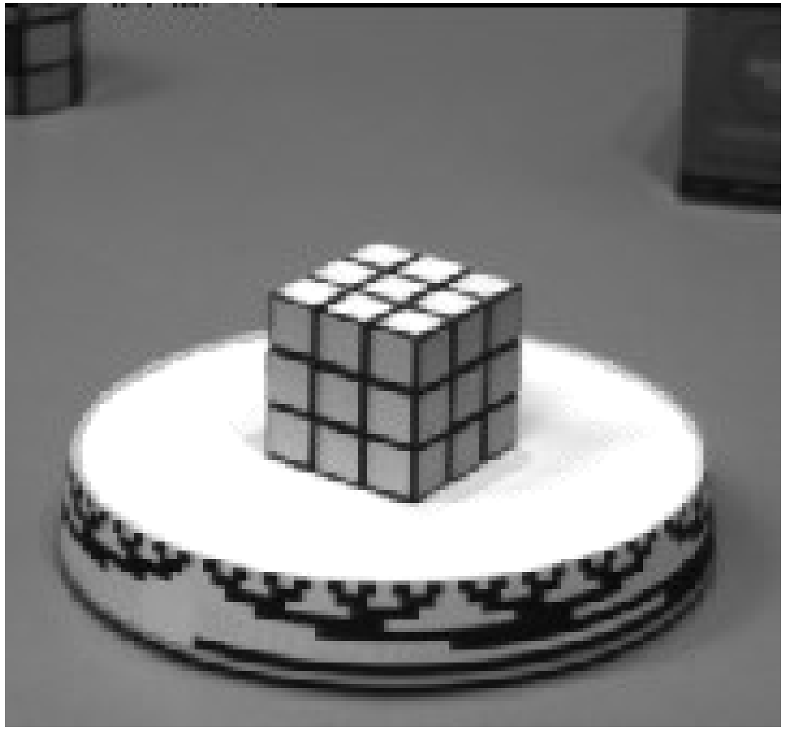
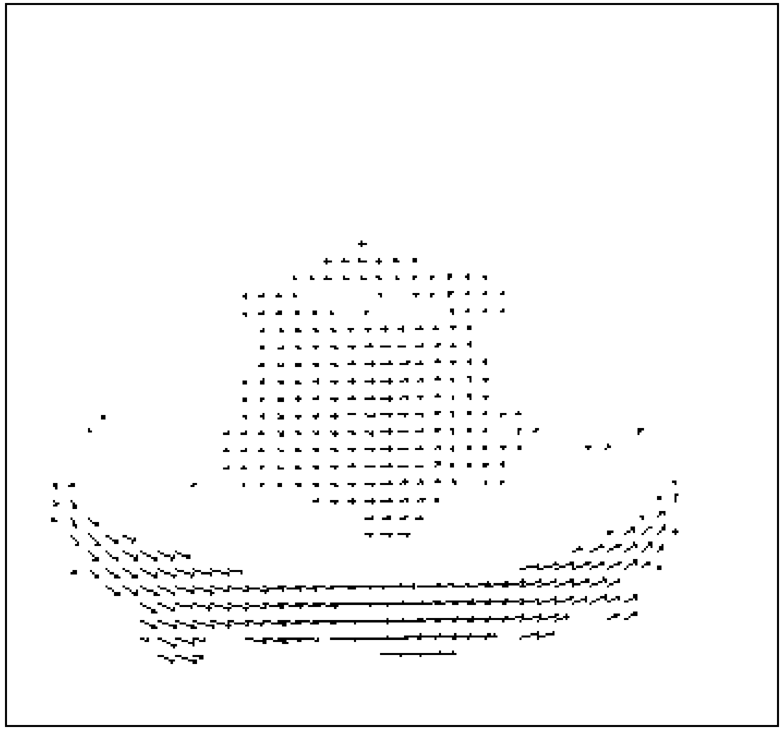
Application to Corner Tracking

Color encodes motion direction

When does it fail?

Application to Optical Flow

What if you track every single pixel in the image?

Application to Optical Flow

Application to Optical Flow

Application to Optical Flow

The Kanade-Lucas-Tomasi (KLT) tracker

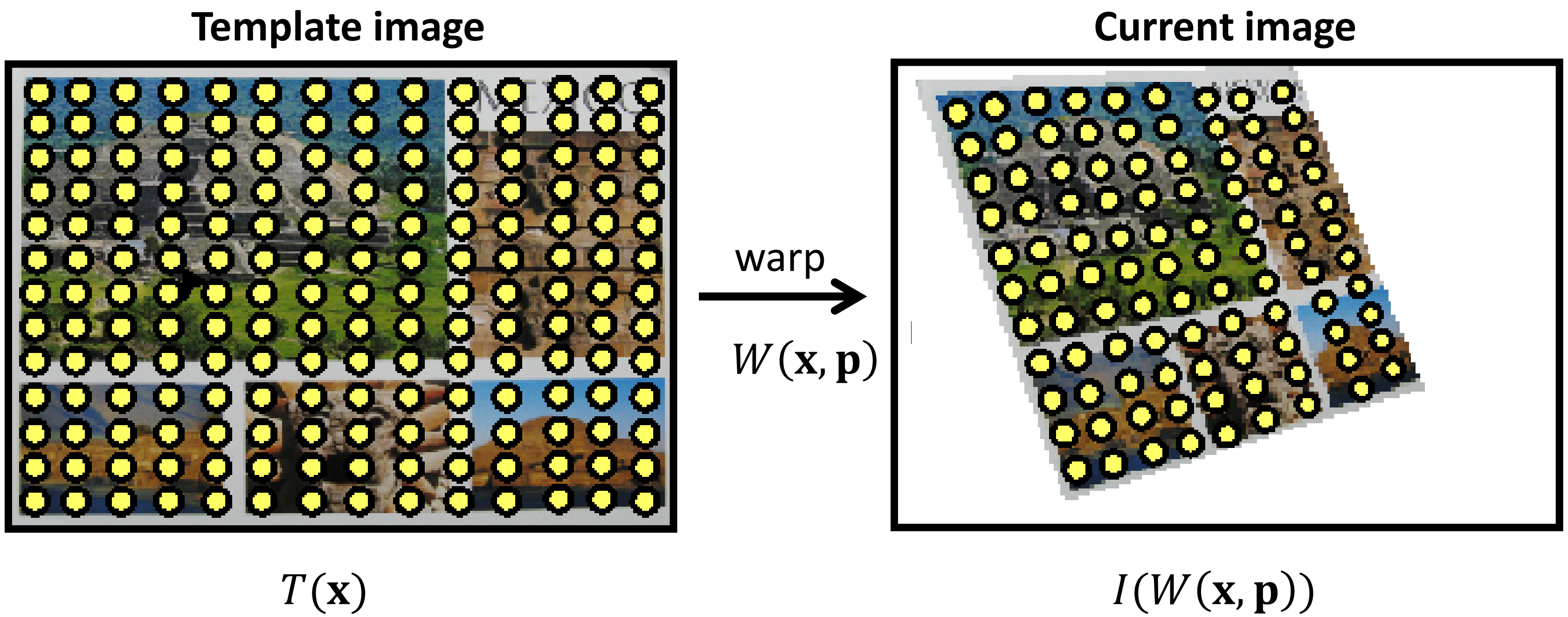
- Simplified case: pure translation
- General case

Lucas, Kanade, *An iterative image registration technique with an application to stereo vision.* Proceedings of Imaging Understanding Workshop, 1981. PDF.
Tomasi, Kanade, *Detection and Tracking of Point Features*, Carnegie Mellon University Technical Report CMU-CS-91-132, 1991. PDF.

KLT applied to generic warps

Goal: estimate the parameters $\mathbf{p}$ of the transformation $W(\mathbf{x}, \mathbf{p})$ that minimize the SSD:

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



* Every yellow dot in this image denotes a pixel

KLT applied to generic warps

Goal: estimate the parameters $\mathbf{p}$ of the transformation $W(\mathbf{x}, \mathbf{p})$ that minimize the SSD:

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

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

KLT applied to generic warps

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

- Assume that an initial estimate of $\mathbf{p}$ is known. Then, we want to find the increment $\Delta\mathbf{p}$ that minimizes

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

- First-order Taylor approximation of $I\big(W(\mathbf{x}, \mathbf{p} + \Delta\mathbf{p})\big)$ yelds to:

$$I\big(W(\mathbf{x}, \mathbf{p} + \Delta\mathbf{p})\big) \cong I\big(W(\mathbf{x}, \mathbf{p})\big) + \nabla I \frac{\partial W}{\partial \mathbf{p}} \Delta\mathbf{p}$$

$\nabla I = [I_x, I_y]$ = Image gradient evaluated at $W(\mathbf{x}, \mathbf{p})$

Jacobian of the warp $W(\mathbf{x}, \mathbf{p})$

KLT applied to generic warps

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

- By replacing $I\big(W(\mathbf{x}, \mathbf{p} + \Delta\mathbf{p})\big)$ with its 1st order approximation, we get

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

- How do we minimize it?
- We differentiate SSD with respect to $\Delta\mathbf{p}$ and we equate it to zero, i.e., $\frac{\partial SSD}{\partial \Delta\mathbf{p}} = 0$

KLT applied to generic warps

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

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

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

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

KLT applied to generic warps

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

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

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

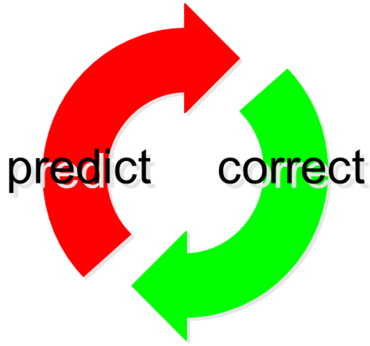
Second moment matrix (Hessian) of the warped image

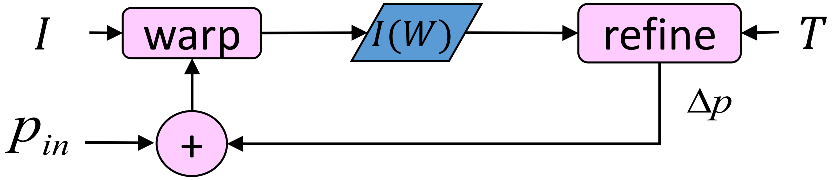
What does H look like when the warp is a pure translation?

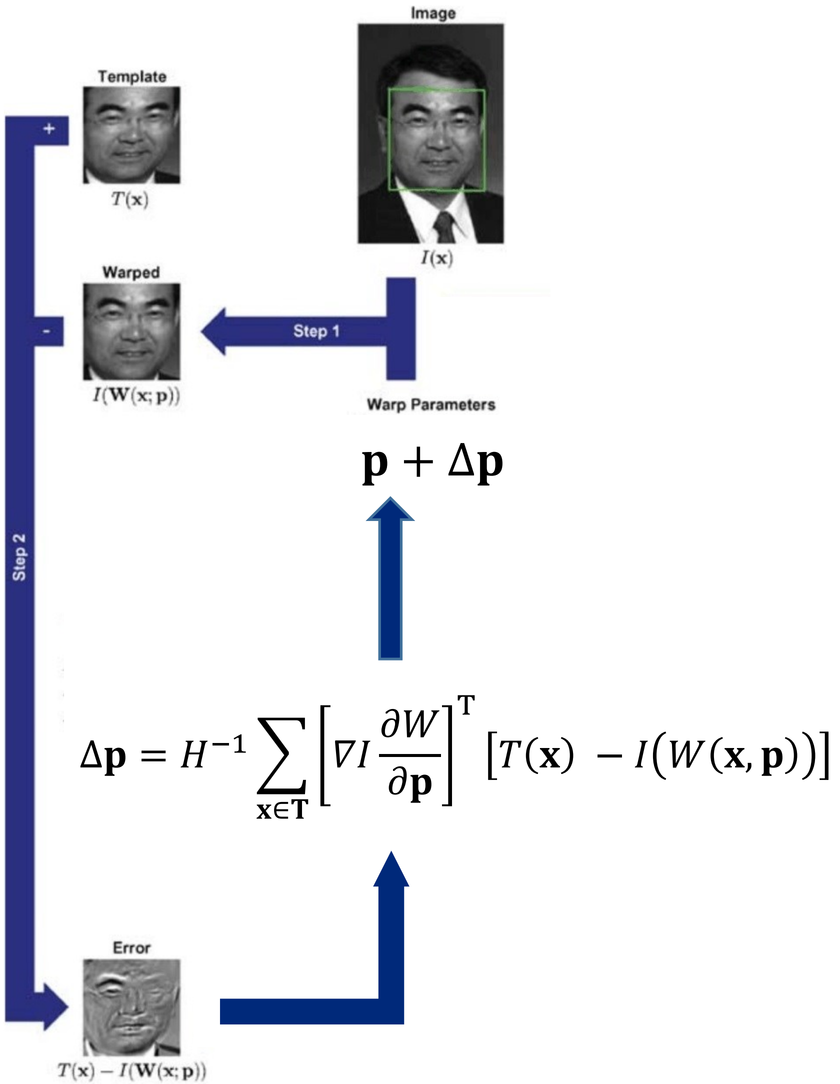
KLT algorithm: Discussion

KLT algorithm is iterated until convergence by following a **predict-correct cycle**

1. A **prediction** $I(W(\mathbf{x},\mathbf{p}))$ of the warped image is computed from an initial estimate of $\mathbf{p}$
2. The **correction** parameter $\Delta\mathbf{p}$ is then 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
3. Steps 1 & 2 are iterated until the error is smaller than a threshold and the output parameters are used as input for the next frame



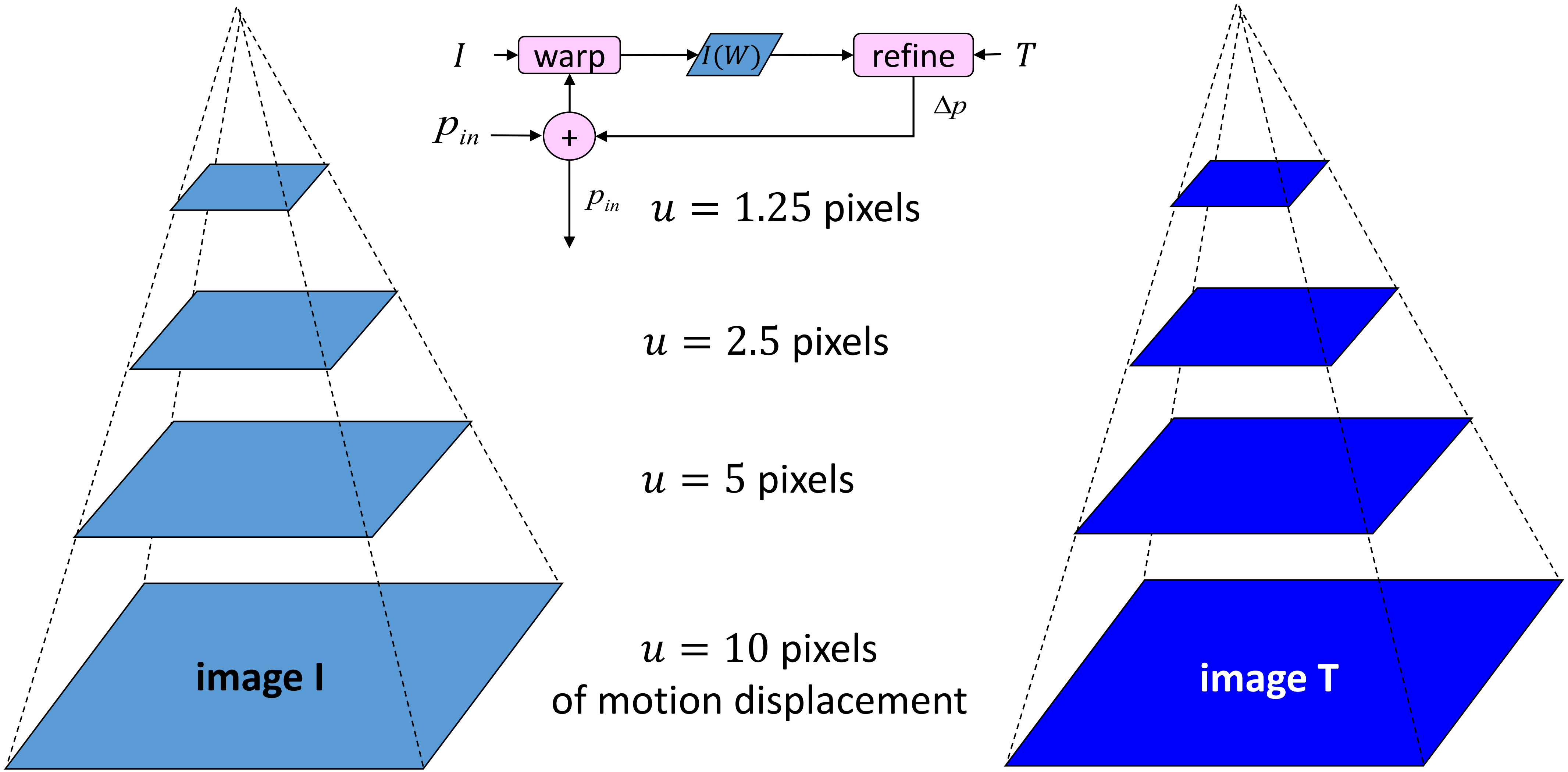




KLT algorithm: Discussion

- 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: **Coarse-to-fine implementations** (see next slide)
  - Too **poor texture**
    → Solution: **increase the aperture** (see next slide)
  - **Deviations from mathematical warp model**: object deformations, illumination changes, etc.
    → Solution: **Update the template with the last image**: problem: drift
  - **Occlusions**
  - **Template background**

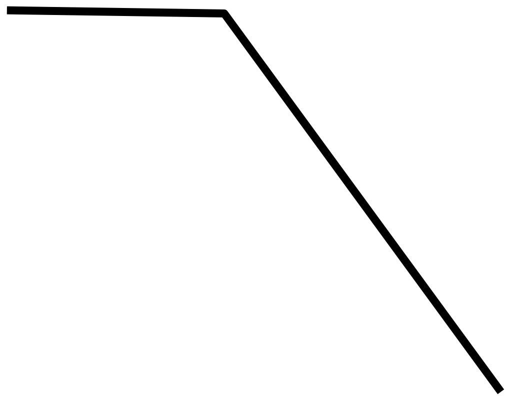
Coarse-to-fine estimation



Pyramid of image I **Pyramid of image T**

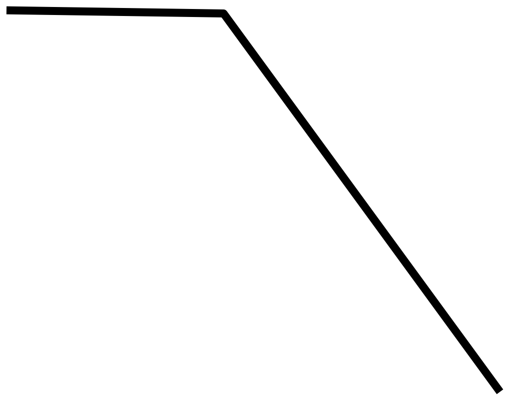
Aperture Problem

- Consider the motion of the following corner



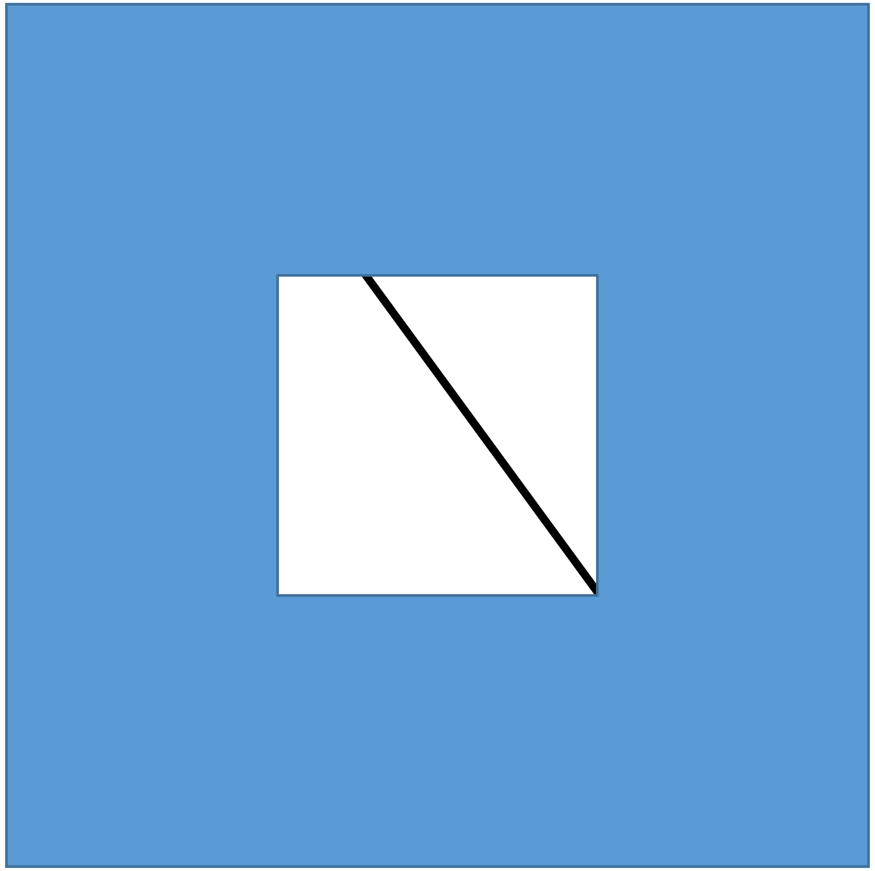
Aperture Problem

- Consider the motion of the following corner



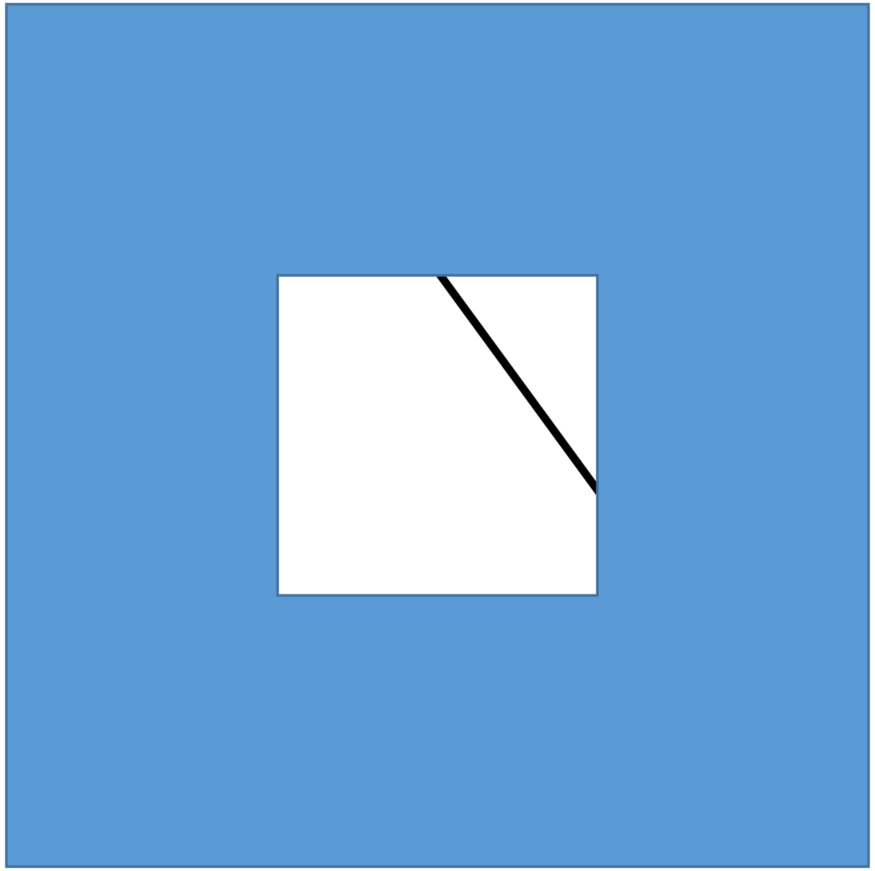
Aperture Problem

- Now, look at the local brightness changes **through a small aperture**



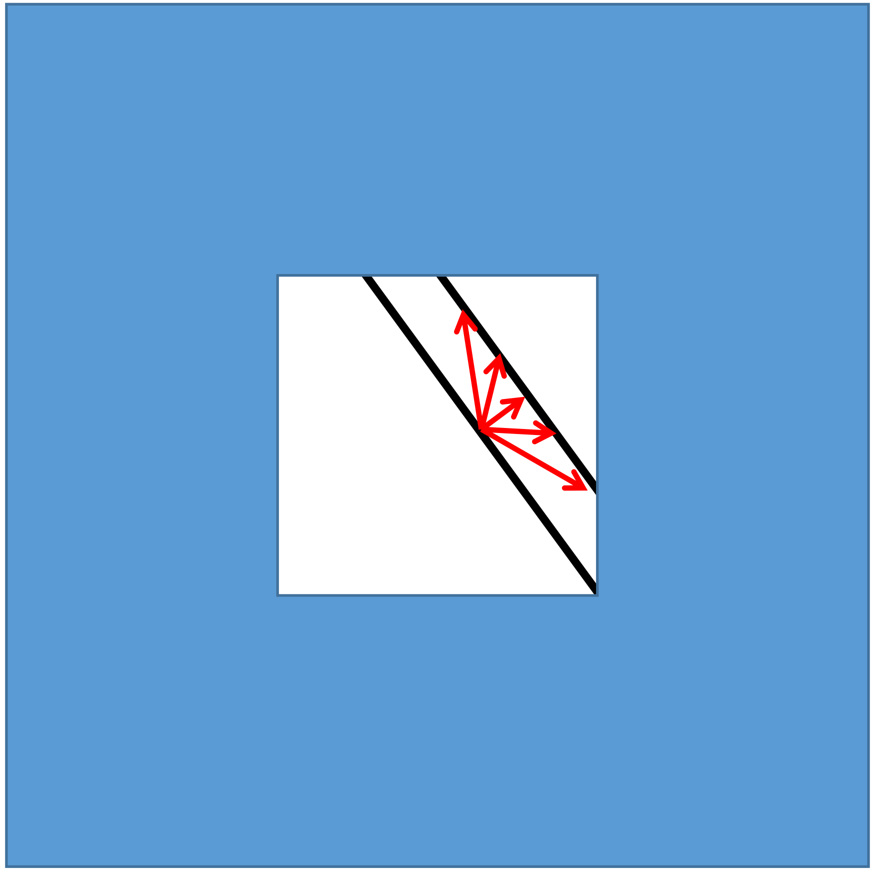
Aperture Problem

- Now, look at the local brightness changes **through a small aperture**



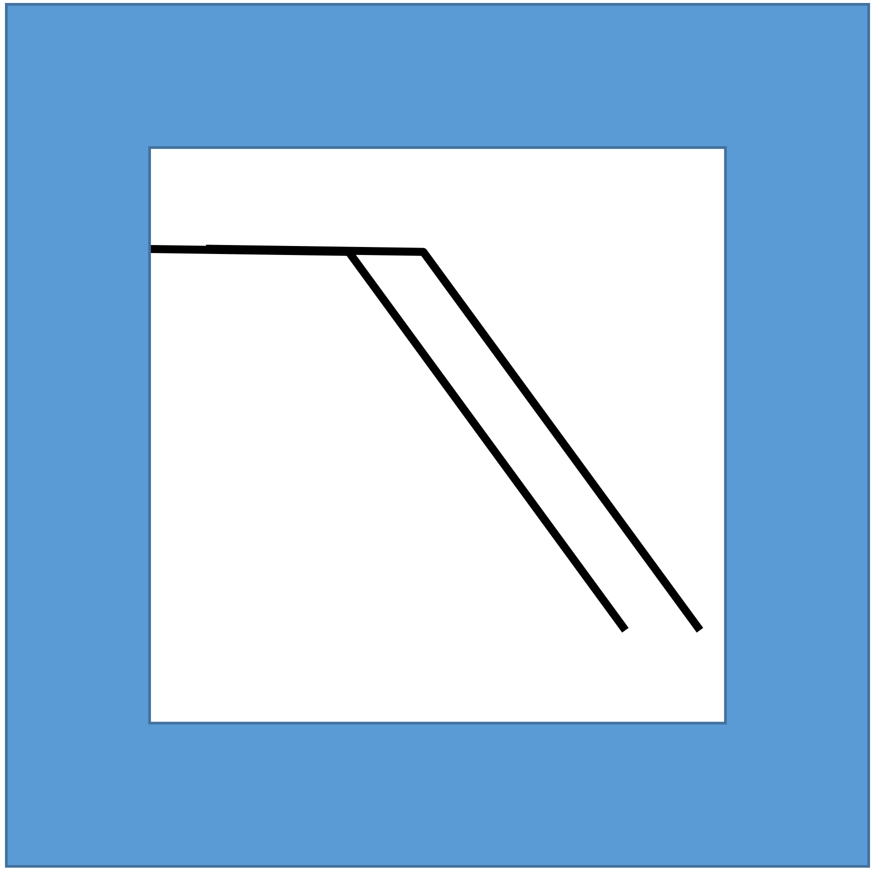
Aperture Problem

- Now, look at the local brightness changes **through a small aperture**
- We **cannot always determine** the motion direction → **Infinite motion solutions** may exist!
- Solution?



Aperture Problem

- Now, look at the local brightness changes **through a small aperture**
- We **cannot always determine** the motion direction → **Infinite motion solutions** may exist!
- Solution?
  - Increase aperture size!

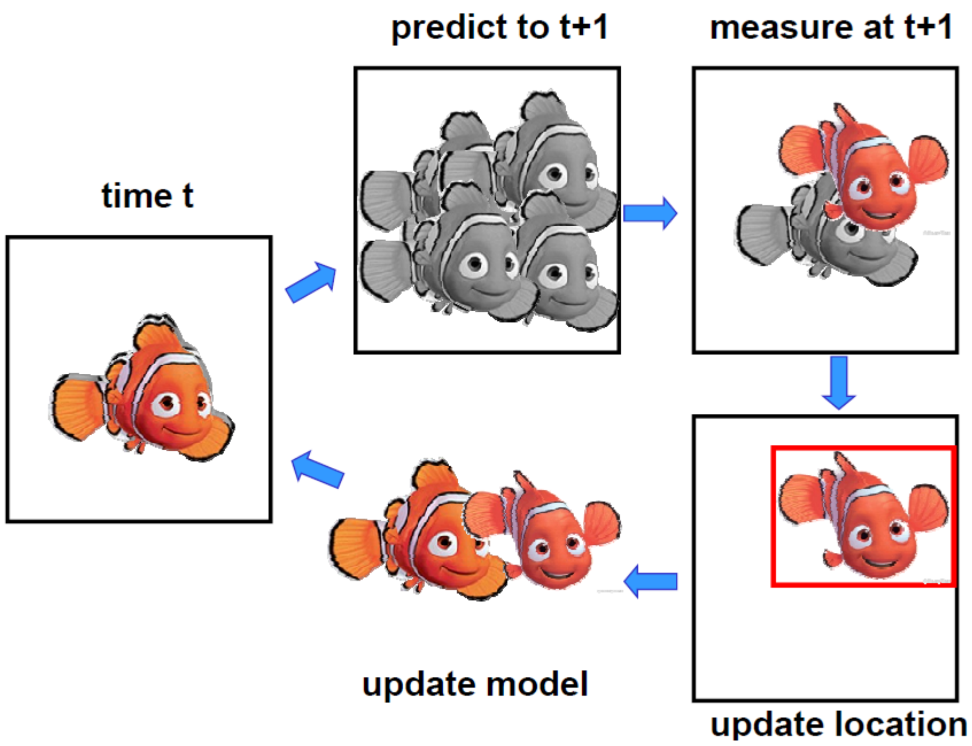


Generalization of KLT

- The same concept (predict/correct) can be applied to tracking of 3D object (in this case, what is the transformation to etimate? What is the template?)

Generalization of KLT

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



Math Refresher

Common 2D Transformations in Matrix form

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

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

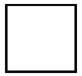
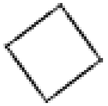
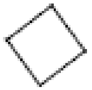
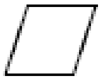
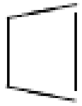
- Affine

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

- 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 in Matrix form

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

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

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

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

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

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

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 (1804-1851)

Displacement-model Jacobians ∇W_p

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

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

- Euclidean: $W(\mathbf{x}, \mathbf{p}) = \begin{bmatrix} xcos(a_3) - ysin(a_3) + a_1 \\ xsin(a_3) + ycos(a_3) + a_2 \end{bmatrix}$ $\quad \frac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} 1 & 0 & -xsin(a_3) - ycos(a_3) \\ 0 & 1 & xcos(a_3) - ysin(a_3) \end{bmatrix}$

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

Readings

- Baker, Matthews, *Lucas-Kanade 20 Years On: A Unifying Framework*, International Journal of Computer Vision, 2004. PDF.

Understanding Check

Are you able to answer the following questions?

- What is the problem formulation of tracking?
- Difference between direct and indirect methods and their pros and cons
- Can you illustrate tracking methods using point features?
- Are you able to explain the underlying assumptions behind direct methods, derive their mathematical expression for the case of pure rotation and the meaning of the M matrix?
- When is the M matrix invertible and when not?
- What is optical flow?
- Are you able to describe the working principle of KLT for a generic warp?
- What functional does KLT minimize?
- What is the Hessian matrix and for which warping function does it coincide to that used for pure translation?
- Can you list Lukas-Kanade failure cases and how to overcome them?
- How do we get the initial guess?
- Can you illustrate the coarse-to-fine Lucas-Kanade implementation?
- What is the aperture problem and how can we overcome it?