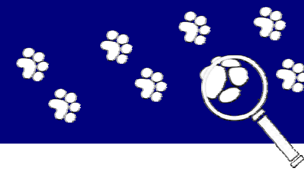
Good Features to Track



Jianbo Shi
Computer Science Department
Cornell University
Ithaca, NY 14853

Carlo Tomasi
Computer Science Department
Stanford University
Stanford, CA 94305

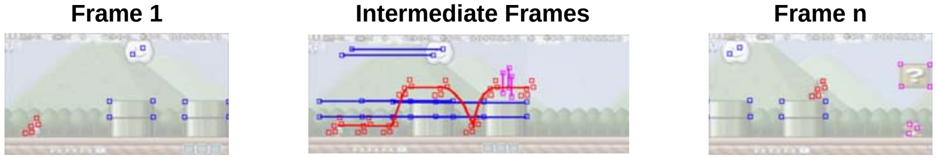
http://www.ces.clemson.edu/~stb/klt/shi-tomasi-good-features-cvpr1994.pdf

http://citeseer.ist.psu.edu/cache/papers/cs/2258/http:zSzzSzrobotics.stanford.eduzSz~birchzSzkltzSzshiCvpr94.pdf/shi94good.pdf

Problem Statement

- Given a set of sequential images, reliably track features across the sequence, while monitoring the quality of each feature

Frame 1 **Intermediate Frames** **Frame n**



Feature Detection → Feature Tracking System → Quality Measurement of Tracked Features

1 2 3

Keep/
Discard
Features

Paper Overview

- Feature Selection
  - Fundamental definition of the Harris corner method
- Tracking System
  - Anandan's Approach limited to only a pure translation model
- Ability to monitor the goodness of a feature throughout tracking process
  - Anandan's approach using full affine parameters (deformation and translation) to measure the dissimilarity between first and the current frame
  - Keep/Abandon features based on dissimilarity measure

- Detect occlusions, disocclusions, and features that do not have real-world correspondence
- Constraint: Inter-frame displacement is small

Terminology

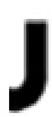
- Occlusions

Shape to Detect

Shape not occluded

Shape is occluded

- Disocclusions:
  - Areas occluded in original reference frame but visible in current view



Detect "J"

Detected "J"

Disocclusion

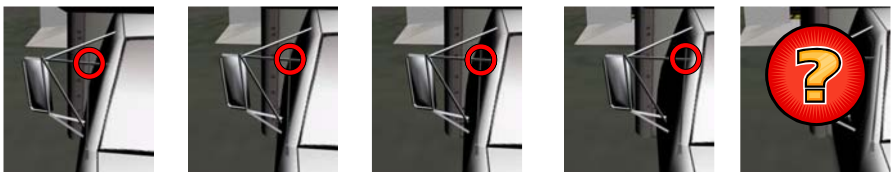
More Disocclusion

Terminology

- Non-real world points

Given Sequence

Antenna and mirror support bar create a feature which does not correlate to a real-world feature



- Feature Detection is unable to discern depth
- Need to monitor features to track reliably

Feature Selection

- Many feature selection options being debated in early 1990's
  - Most measure the amount of texturedness or cornerness in a window
  - Windows with high spatial frequency content
  - High standard deviation on the spatial intensity profile
  - Presence of zero crossings of the Laplacian of the image intensity
  - Regions where second-order derivatives are above a threshold
  - Corner detection
  - Even a window rich in texture can be a poor point to track
    - Non real-world point, occlusion/disocclusion, reflective surface, shadows, etc.
  - Tracking based solely on one of the above methods will most likely be unsuccessful and error-prone
- Paper proposes a fundamental definition for feature quality
  - i.e. Harris Corner Method
- Used for initial feature selection, not for further tracking

Feature Selection

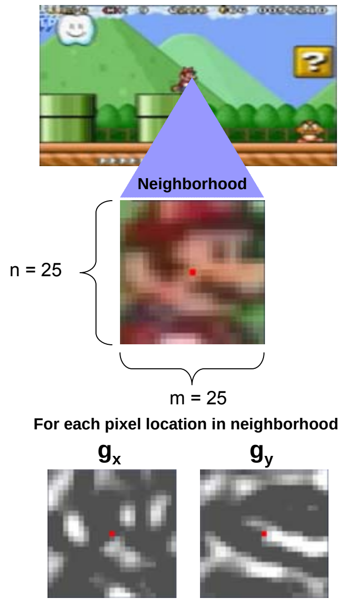
Basic Harris Corner Method

1. Given an image
2. Smooth image with Gaussian Filter
3. Compute derivatives $\{g_x\}$ and $\{g_y\}$ for smoothed image
4. Option: Smooth derivative images $\{g_x\}$ and $\{g_y\}$
5. For each pixel in the image space, compute the gradient moment matrix, using the n x m neighborhood of pixels (window) around current pixel.

$$M = \int\int_W Zwdxdy \quad \text{where,} \quad Z = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix}$$

W = window (neighborhood) = n x m = i.e. 5 x 5, 25 x 25, etc.
w = 1, OR a 2D Gaussian weighting scheme

OR, $$M = \begin{bmatrix} \sum_i^n \sum_j^m g_x^2 w & \sum_i^n \sum_j^m g_x g_y w \\ \sum_i^n \sum_j^m g_x g_y w & \sum_i^n \sum_j^m g_y^2 w \end{bmatrix}$$



Feature Selection

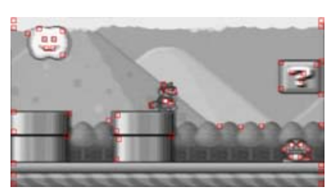
6. Compute the two Eigen values for the gradient moment matrix M
   - Two requirements must be upheld for the matrix M
   1. Above the Noise Level
      - Both Eigen values must be large
   2. Well-Conditioned
      - Eigen values cannot differ by several orders of magnitude
7. Select the minimum Eigen value

$$\min(\lambda_1, \lambda_2) > \lambda_{Threshold}$$

   - Smaller Eigen value meets noise-level-criterion
   - Well-conditioned because intensity variations are bounded by image intensity range (i.e. 0-255).
8. Store the minimum Eigen value for each pixel in the image
9. Apply a type of Non-Maximum Suppression to the Eigen values
10. Threshold Suppressed Eigen value space to reduce amount of detected interest points

Alternative Computation to 6,7:
R = det(M) + k trace(M)^2 > Threshold

λ_1	λ_2	Texturedness
Small	Small	Constant intensity profile (nothing)
Small	Large	Unidirectional texture pattern (edge)
Large	Small	Unidirectional texture pattern (edge)
Large	Large	Corner, salt-and-pepper texture, (texture can be tracked reliably)



What is Next?

- Feature Selection used for initial detection only
- How to Track?
- Affine Motion Model
  - Last Semester Project: Anandan’s Approach

Starting Image **Warped Image** **Goal Image**

Rotated and Enlarged

  - Inter-frame displacement is relatively small
  - Brightness constancy constraint
  - Uses
    - Image registration
    - Mosaics/Panoramic views
    - Morphing technology
    - Tracking (uses pure translation of affine motion model)
    - Measuring quality of tracked feature (complete affine model)
  - Authors apply Anandan’s approach to neighborhood around features

Affine Motion Model

- Affine model for one pixel

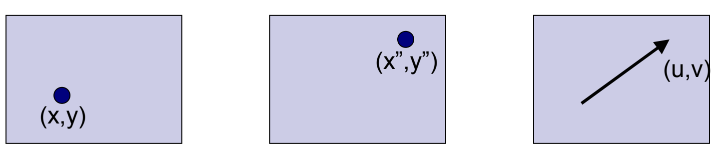


image at time t image at time t+1

u = x” – x
v = y” - y

Affine motion:

$$u(x,y) = a_1 x + a_2 y + b_1$$
$$v(x,y) = a_3 x + a_4 y + b_2$$

Affine motion parameters:

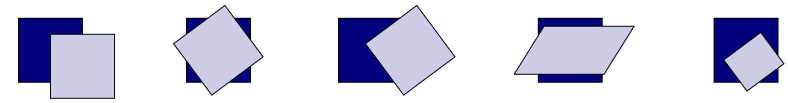
$$\{a_1, a_2, b_1, a_3, a_4, b_2\}$$

Affine Transformation:

$$x'' - x = a_1 x + a_2 y + b_1 \qquad y'' - y = a_3 x + a_4 y + b_2$$
$$x'' = (a_1 + 1)x + a_2 y + b_1 \qquad y'' = a_3 x + (a_4 + 1)y + b_2$$

Affine Motion Model

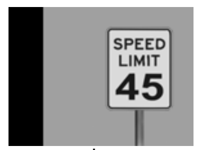
- Affine model handles translation, rotation, rigid rotation and translation, affine, and shear



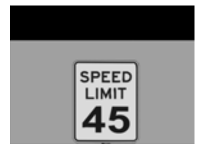
translation rotation rigid shear affine

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

original I



$- \leftarrow b_1 \rightarrow +$

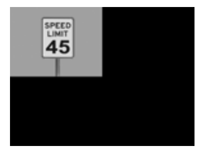


$-$ $\uparrow$ b_2 $\downarrow$ $+$

a_3 positive

a_3 negative

a_1 & a_4 positive



a_1 & a_4 negative

Affine Motion Model

$$u(x,y) = a_1 x + a_2 y + b_1$$
$$v(x,y) = a_3 x + a_4 y + b_2$$

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

where, $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ $\mathbf{u}(\mathbf{x}) = \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix}$

$$\mathbf{a}^{\mathrm{T}} = \begin{bmatrix} a_1 & a_2 & b_1 & a_3 & a_4 & b_2 \end{bmatrix}$$

$$\mathbf{X}(\mathbf{x}) = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}$$

$$\boxed{\mathbf{u}(\mathbf{x}) = \mathbf{X}(\mathbf{x})\mathbf{a}}$$

Affine Motion Model

- Optical Flow Equation

$$I_x u + I_y v = -I_t \rightarrow \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t \rightarrow \Delta I^T \mathbf{u} = -I_t$$

- Energy Functional

$$E(\mathbf{u}) = \sum_W \left(I_t + \Delta I^T \mathbf{u} \right)^2 \qquad E(\mathbf{a}) = \sum_W \left(I_t + \Delta I^T \mathbf{X} \mathbf{a} \right)^2$$

- Minimize energy by taking derivative and setting it equal to zero

Affine Motion Model

$$E(\mathbf{a}) = \sum_W \left(I_t + \Delta I^T \mathbf{X} \mathbf{a} \right)^2$$

$$\frac{\partial E}{\partial \mathbf{a}} = 2 \sum_W \left(\Delta I^T \mathbf{X} \right)^T \left(I_t + \Delta I^T \mathbf{X} \mathbf{a} \right) = 0$$

$$\sum_W \mathbf{X}^{\mathbf{T}} \Delta I \Delta I_t + \sum_W \mathbf{X}^{\mathbf{T}} \Delta I \Delta I^T \mathbf{X} \mathbf{a} = 0$$

$$\sum_W \mathbf{X}^{\mathbf{T}} \Delta I \Delta I^T \mathbf{X} \mathbf{a} = -\sum_W \mathbf{X}^{\mathbf{T}} \Delta I \Delta I_t$$

Affine Motion Model

$$\underbrace{\sum_W \mathbf{X}^{\mathbf{T}} \Delta I \Delta I^T \mathbf{X}}_{\mathbf{K}} \mathbf{a} = -\underbrace{\sum_W \mathbf{X}^{\mathbf{T}} \Delta I \Delta I_t}_{\mathbf{L}}$$

$$\mathbf{K}_{6x6}\mathbf{a}_{6x1} = \mathbf{L}_{6x1} \longrightarrow \mathbf{a} = \mathbf{K}^{-1}\mathbf{L}$$

- Update previous **a** with new **a**
  - Concatenation procedure
- Iteratively solve for affine parameters **a** until updates do not change or some iteration limit is reached

Affine Motion Model

- Author's method similar to Anandan's
  - Affine Motion

$$\delta = D\mathbf{x} + \mathbf{d} \qquad D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \qquad \mathbf{d} = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

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

  - Affine Transformation
    - A point **x** in the first image, I, moves to a point A**x**+**d** in the second image J, where A = **1** + D and **1** is the 2 x 2 identity matrix

$$J\left(A\mathbf{x} + \mathbf{d}\right) = I\left(\mathbf{x}\right) \qquad (2)$$

Tracking

- Given two images I and J
- Tracking means computing D and **d**
- Quality of computation depends on
  - Size of feature window
  - Texturedness inside the feature window
  - Amount of camera/object motion between frames
- When window is small, or when inter-frame motion is small, D is harder to estimate
  - Variations of motion within window are small
  - D is not reliable
- However, small windows are preferred for tracking
  - Less likely to straddle depth discontinuity
- Therefore, a pure translational model is used for tracking
  - D is assumed to be zero

$$\delta = \mathbf{d}$$

Two Models of Image Motion

1. Affine Model (D + **d**)
2. Pure Translation Model (**d**)

- Use Pure Translation for tracking
  - Higher reliability
  - Higher accuracy
  - Inter-frame motion tends to be small
  - Less computations
- Use Affine Motion to monitor quality of features
  - Between first and current frame
  - Not computed every frame! Every n^{th} frame

Computing Image Motion

- Both motion models measure **dissimilarity** between frames
  - Find an A and **d** that minimizes this dissimilarity
  - Increasing number of iterations for model can improve dissimilarity parameter

dissimilarity,

$$\varepsilon = \int\int_W \left[J\left(A\mathbf{x}+\mathbf{d}\right) - I\left(\mathbf{x}\right) \right]^2 w\left(\mathbf{x}\right) d\mathbf{x} \quad (3)$$

  - W = window (neighborhood) = n x m = i.e. 5 x 5, 25 x 25, etc.
  - *w* = 1, OR a 2D Gaussian weighting scheme

- To minimize (3), take derivative and set equal to zero
- Linearize result by a truncated Taylor series
  - Due to this truncation, method must be solved iteratively

Computing Image Motion

- Linearization yields,

$$T_{6x6}\mathbf{z}_{6x1} = \mathbf{a}_{6x1} \quad (5)$$

Affine motion Dissimilarity

where **z** is comprised of affine parameters, D and **d**

$$\mathbf{z}^T = \left[d_{xx} \quad d_{yx} \quad d_{xy} \quad d_{yy} \quad d_x \quad d_y \right]$$

and **a** is the error vector,

$$\mathbf{a} = \int\int_W \left[I\left(\mathbf{x}\right) - J\left(\mathbf{x}\right) \right] \begin{bmatrix} xg_x \\ xg_y \\ yg_x \\ yg_y \\ g_x \\ g_y \end{bmatrix} w\left(\mathbf{x}\right) d\mathbf{x}$$

This method of calculation requires two images and is therefore not used

Computing Image Motion

- T can be computed from one image

$$T = \iint_W \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} w(\mathbf{x})\, d\mathbf{x} \quad (6)$$

$$U = \begin{bmatrix} x^2 g_x^2 & x^2 g_x g_y & xy g_x^2 & xy g_x g_y \\ x^2 g_x g_y & x^2 g_y^2 & xy g_x g_y & xy g_y^2 \\ xy g_x^2 & xy g_x g_y & y^2 g_x^2 & y^2 g_x g_y \\ xy g_x g_y & xy g_y^2 & y^2 g_x g_y & y^2 g_y^2 \end{bmatrix} \qquad Z = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix}$$

$$V^T = \begin{bmatrix} x g_x^2 & x g_x g_y & y g_x^2 & y g_x g_y \\ x g_x g_y & x g_y^2 & y g_x g_y & y g_y^2 \end{bmatrix}$$

D and **d** interaction in matrix V

∴ errors in D seep into **d**

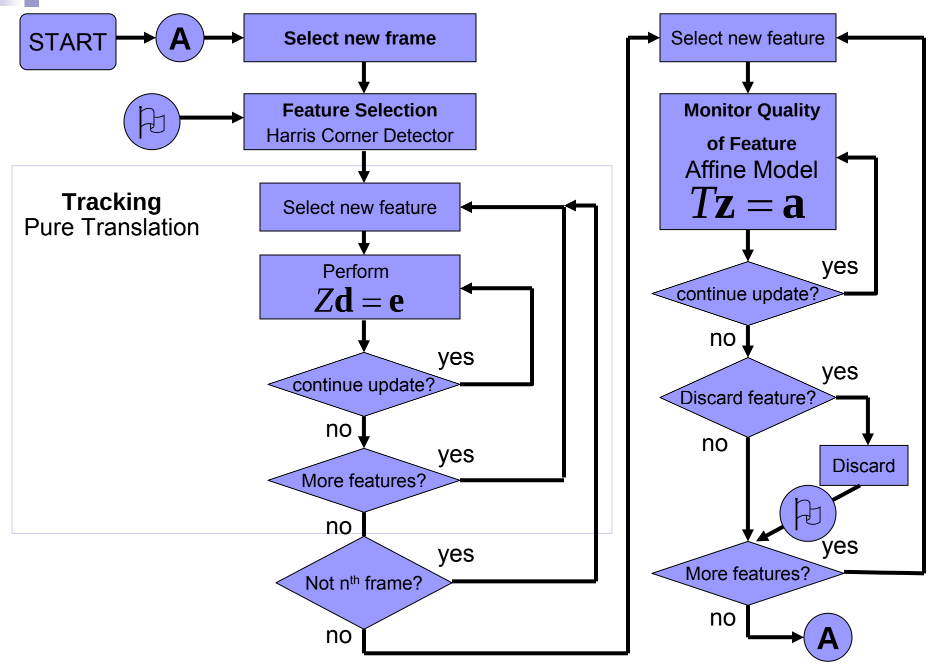
Computing Image Motion

- For Pure Translation Model

$$Z\mathbf{d} = \mathbf{e} \quad (7)$$ Pure Translation Dissimilarity

$$Z = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} \qquad \mathbf{d} = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

- Same Z used to compute Eigen values in corner detector
- Derivation by Stan Birchfield (developed KLT program)
  - *Derivation of Kanade-Lucas-Tomasi Tracking Equation* (1997)



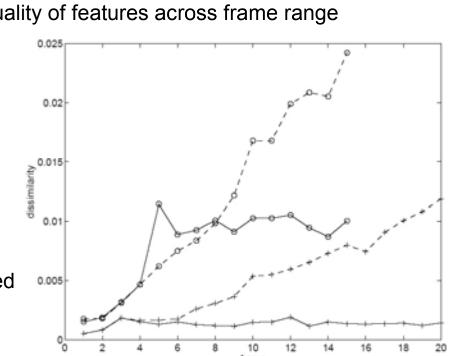
Dissimilarity

- Not all features are good to track & some features are only good to track for a while
- **Dissimilarity** indicates possible change in feature (becomes a bad feature)
- Typical video spans a large number of frames
  - Pure translational model good for inter-frame tracking
  - Pure translation dissimilarity measure not good across a large number of frames
  - Affine dissimilarity better measures the quality of features across frame range

Example 1: Woody Allen's *Manhattan*

1st frame 11th frame 21st frame

Sign mostly translates, but does increase size by 15%



Crosses (+) = Example 1

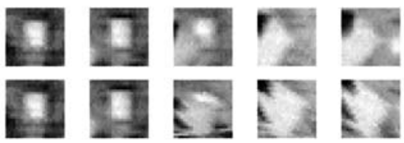
Dashed line = Pure Translation

Solid Line = Affine Transformation

Dissimilarity

Example 2: Woody Allen's *Manhattan*

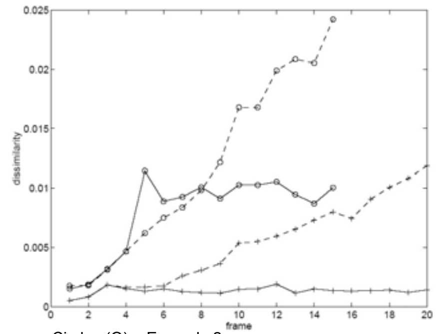
1st frame 5th frame 15th frame



Tracked

Affine warping

1 3 5 10 15



Circles (O) = Example 2

Dashed line = Pure Translation

Solid Line = Affine Transformation

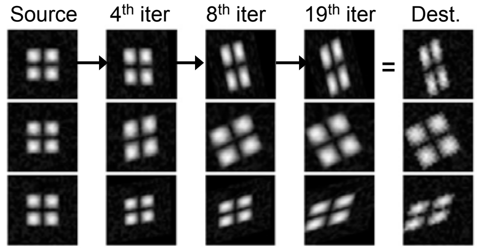
- Glass window becomes occluded in middle frame
- Dissimilarity spike in affine transformation curve at frame 5 indicates occlusion
- Affine warping tries to deform traffic sign into a window

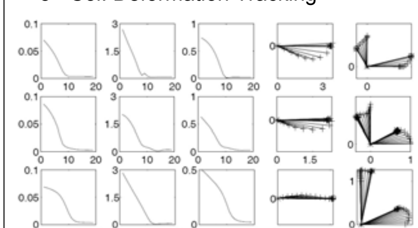
Convergence

- Dissimilarity looked at an entire sequence of frames
  - Many affine dissimilarity measurements computed
- Convergence: comparing the first and current frames
  - Fitting current frame (source) to first frame (destination)
  - One dissimilarity measurement
  - Iterative method
  - Leftmost column: source
  - Rightmost column: destination
    - 16% Gaussian noise added
  - Middle cols: after 4, 8, & 19 iterations

- 1st Col: Dissimilarity
- 2nd Col: Displacement Error (in pixels)
- 3rd Col: Deformation Error
  - Horizontal axis: iteration number
- 4th Col: Displacement Tracking
- 5th Col: Deformation Tracking

Source 4th iter 8th iter 19th iter Dest.





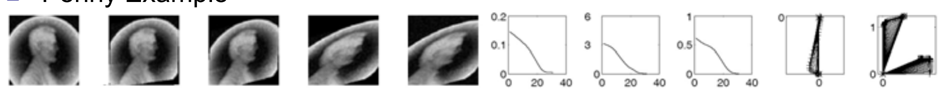
Convergence

- Comparisons for previous slide

	True Deformation	Computed Deformation
1	$\begin{bmatrix} 1.409 & -0.342 \\ 0.342 & 0.563 \end{bmatrix}$	$\begin{bmatrix} 1.393 & -0.334 \\ 0.338 & 0.569 \end{bmatrix}$
2	$\begin{bmatrix} 0.658 & -0.342 \\ 0.342 & 0.658 \end{bmatrix}$	$\begin{bmatrix} 0.670 & -0.343 \\ 0.319 & 0.660 \end{bmatrix}$
3	$\begin{bmatrix} 0.809 & 0.253 \\ 0.342 & 1.232 \end{bmatrix}$	$\begin{bmatrix} 0.802 & 0.235 \\ 0.351 & 1.227 \end{bmatrix}$

	True Translation	Computed Translation
1	$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 3.0785 \\ -0.0007 \end{bmatrix}$
2	$\begin{bmatrix} 2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 2.0920 \\ 0.0155 \end{bmatrix}$
3	$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 3.0591 \\ 0.0342 \end{bmatrix}$

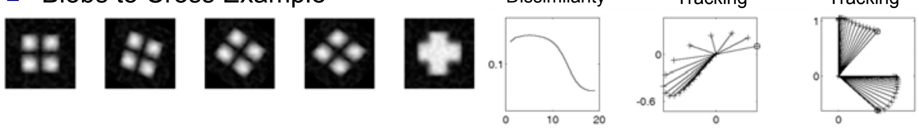
- Penny Example



Source 4th iter 8th iter 19th iter Dest.

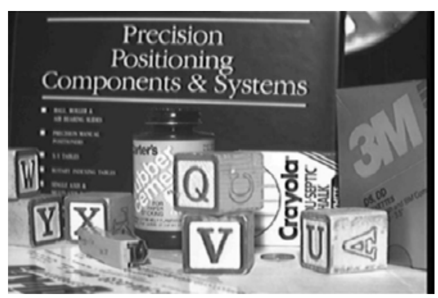
- Blobs to Cross Example

Dissimilarity Displacement Tracking Deformation Tracking



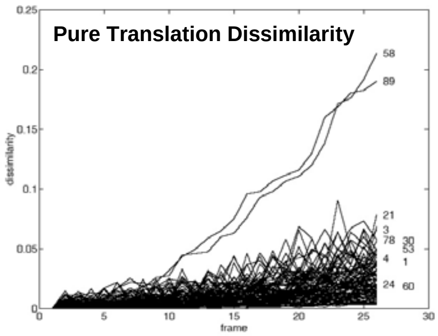
Monitoring Features

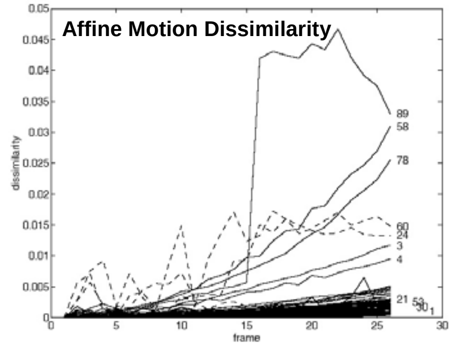
- Real world image sequence
  - 26 frame sequence
  - Camera moves forward
  - Objects become larger
  - Due to depth issue, the following will occur
    - Occlusions
    - Disocclusions
    - Non-real points
  - 102 features selected
  - Limited # features by prohibiting overlapping feature windows during feature selection process



Monitoring Features

- Pure translation is sufficient for inter-frame tracking
  - Not for monitoring
  - All features, except two, have comparable dissimilarities
  - No way to distinguish good from bad features





- Affine Motion Dissimilarity
  - Good for monitoring
  - Seven features have high dissimilarity, thus bad and are discarded
  - Thick band of curves at bottom represents all good features (keep)

KLT Demo