Computer Vision (CS763)

Teaching cameras to "see"

KLT Tracker

Arjun Jain

Most slides courtesy Ajit Rajwade

https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf

Feature Point Tracking-Problem Statement

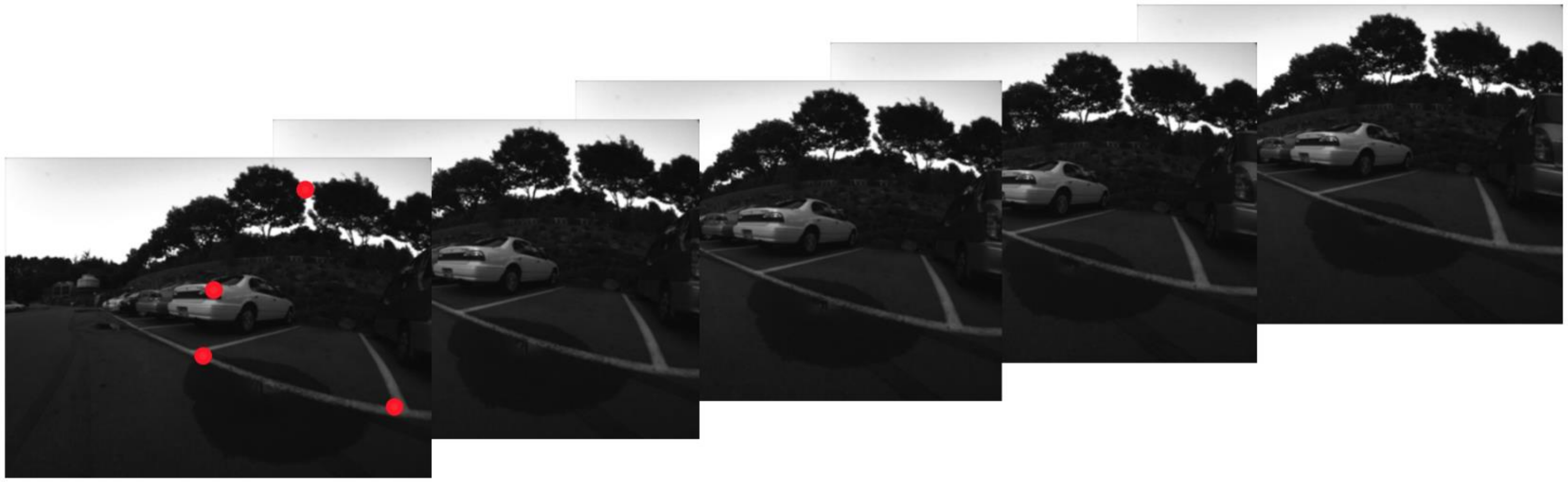
- Input: video sequence
- Output: A list of “good” feature points marked out in frame 1, and their positions in all subsequent video frames.

Feature Point Tracking-Problem Statement

Image sequence

Feature Point Tracking

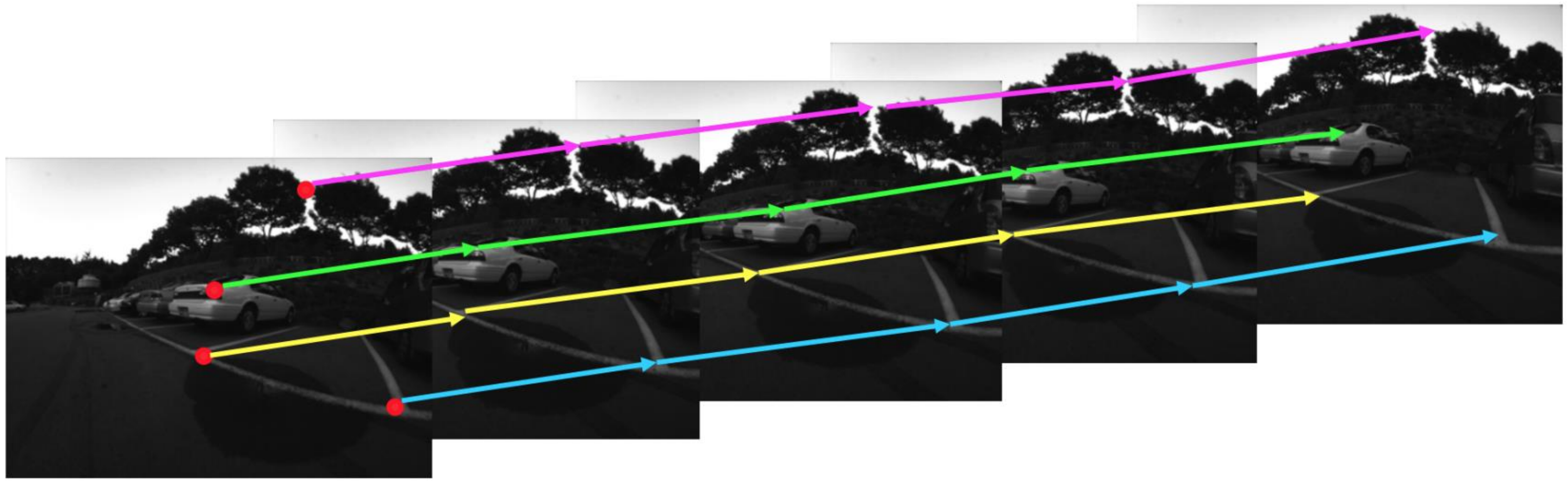
Feature point detection



Slide credit: Yonsei Univ.

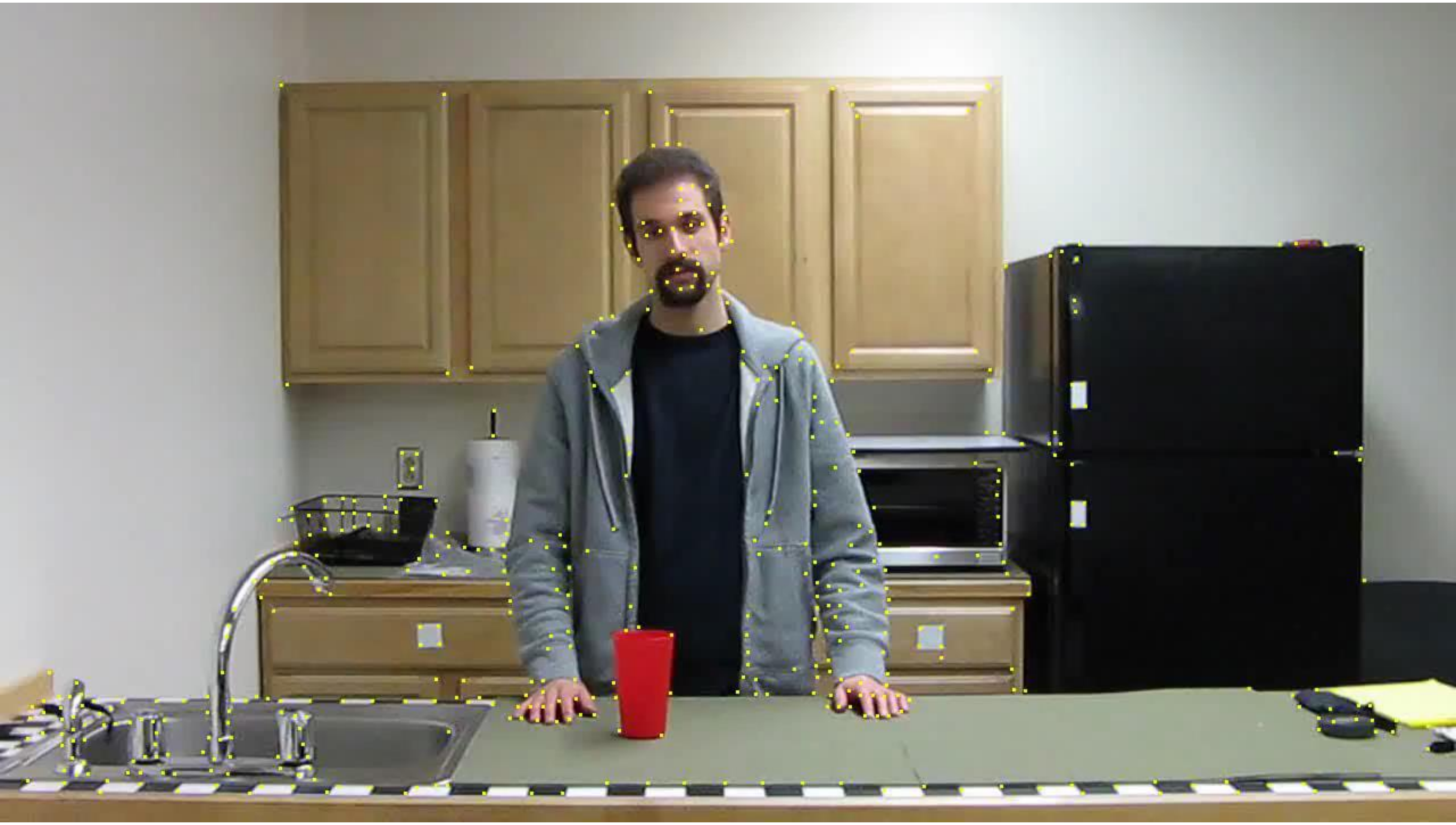
Feature Point Tracking

Feature point tracking

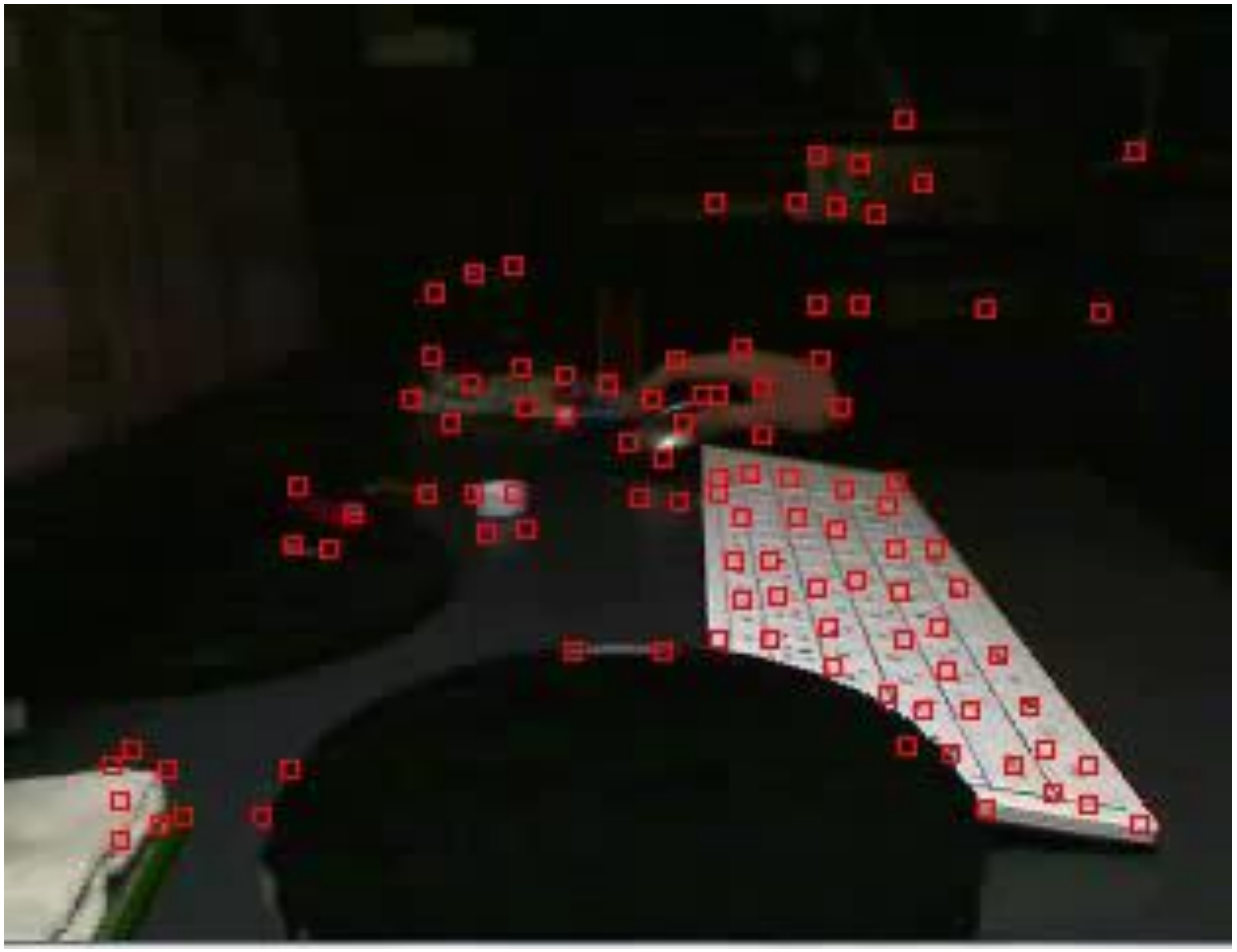


Slide credit: Yonsei Univ.

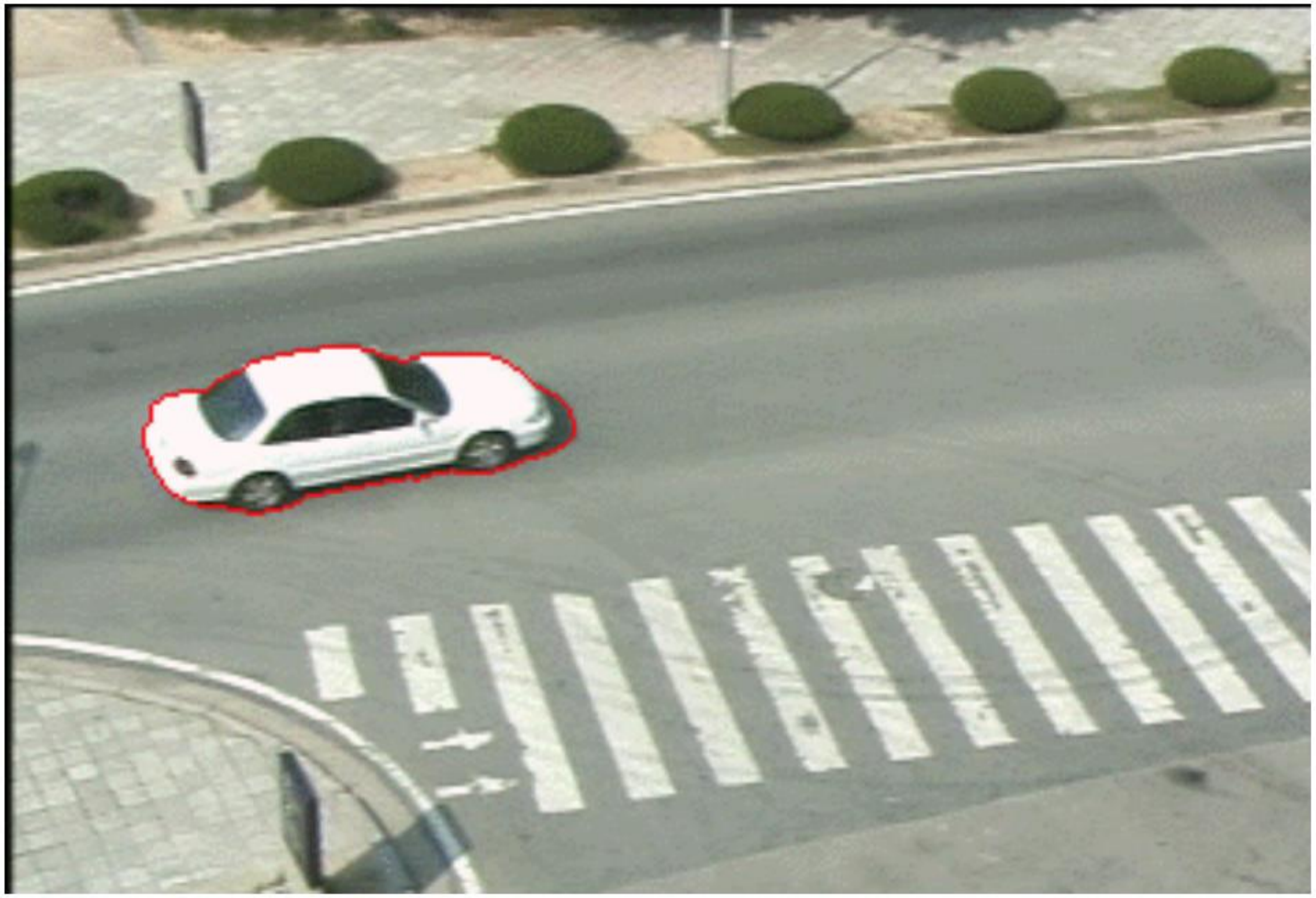
Feature Point Tracking



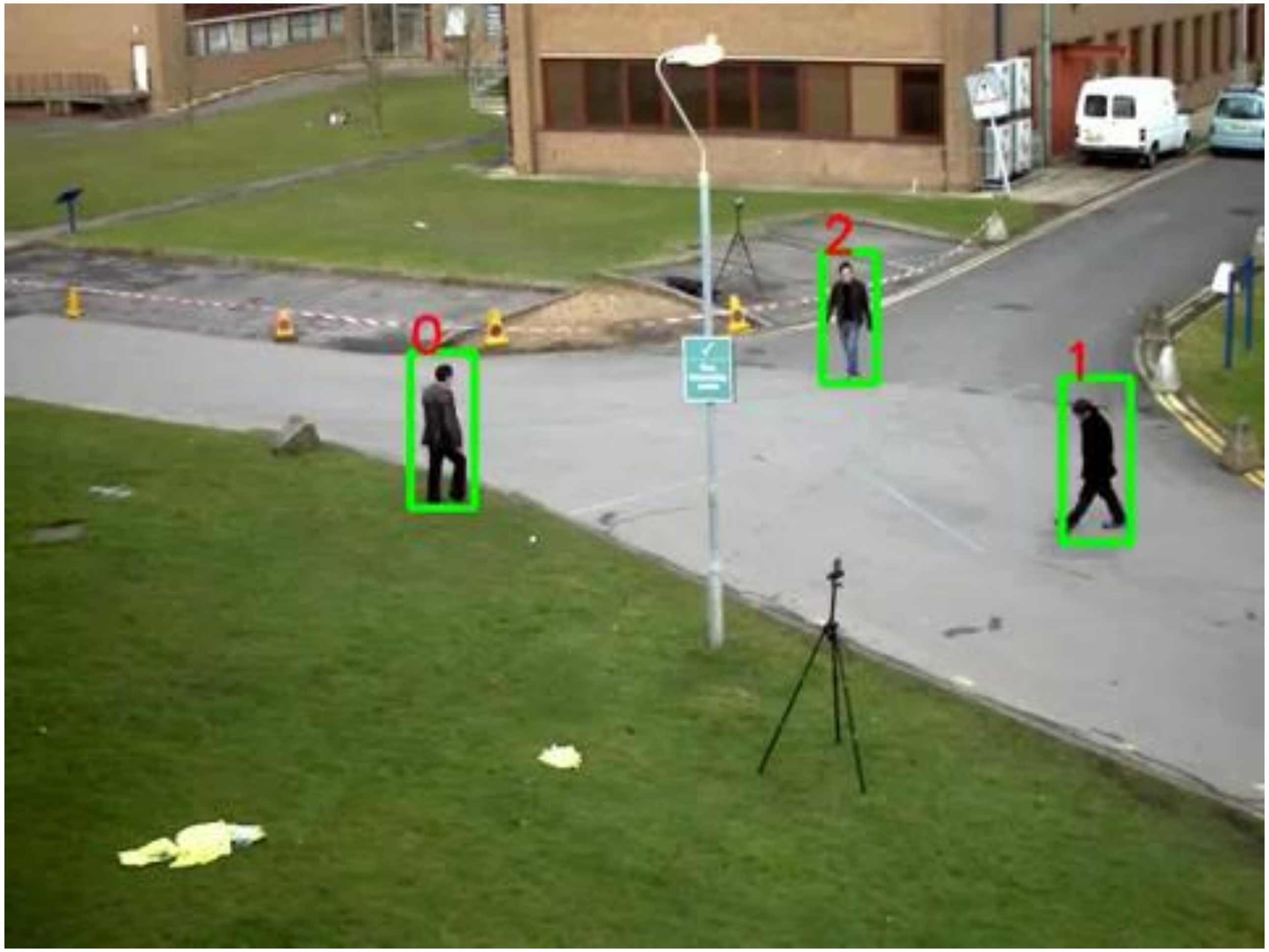
Feature Point Tracking



Other types of Tracking: 1. Object Tracking

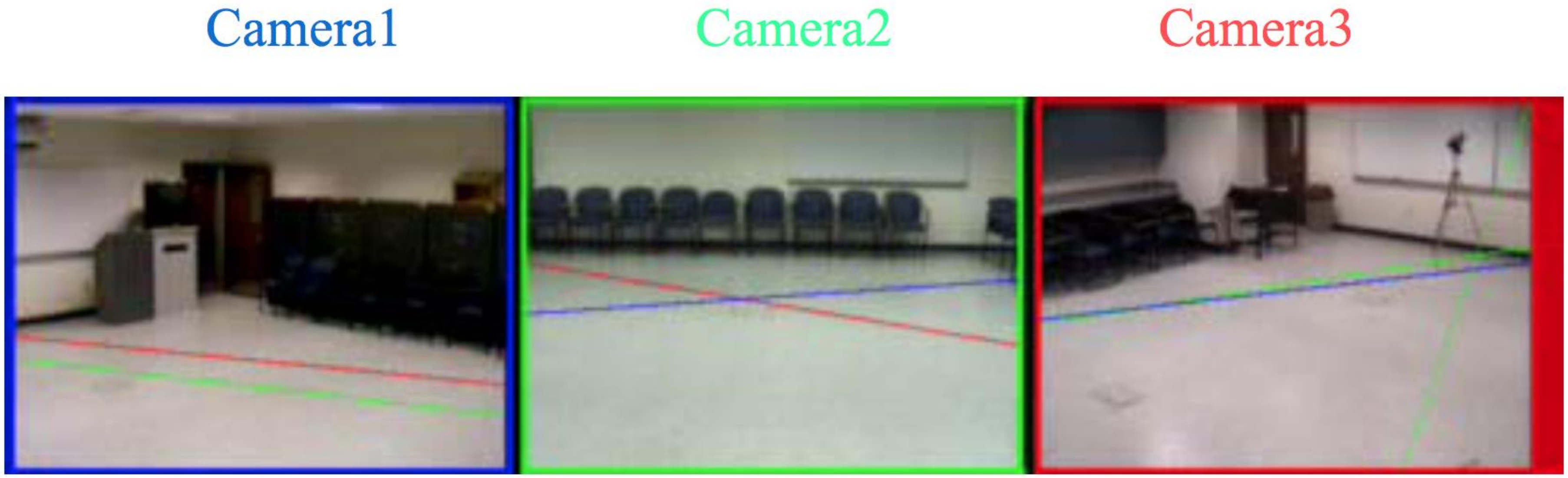


2. Multiple Object Tracking



3. Tracking with a Fixed Camera

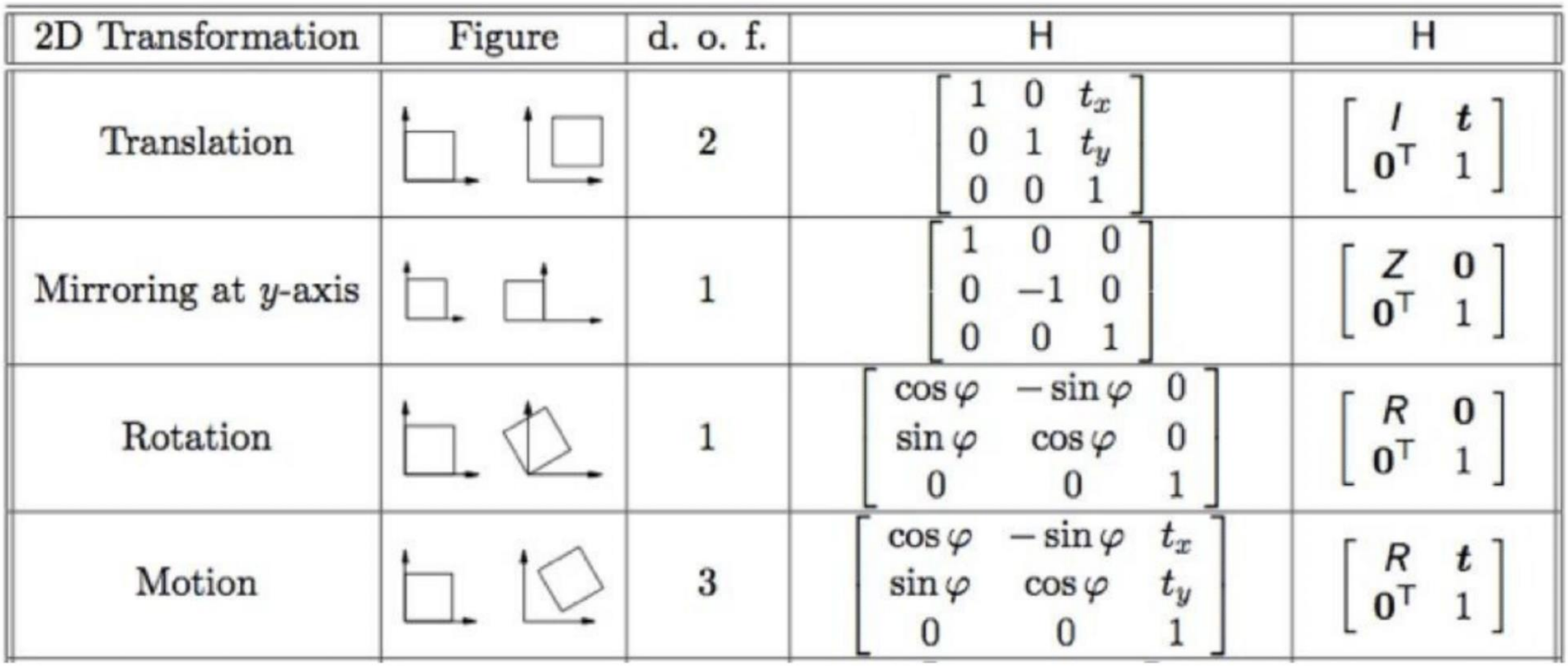
4. Multiple Fixed & Overlapping Cameras Tracking



5. Tracking with a Moving Camera

Motion Models

2D Motion Models



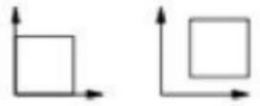
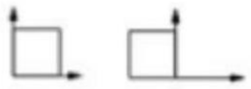
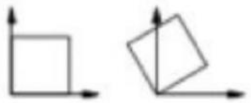
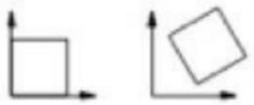
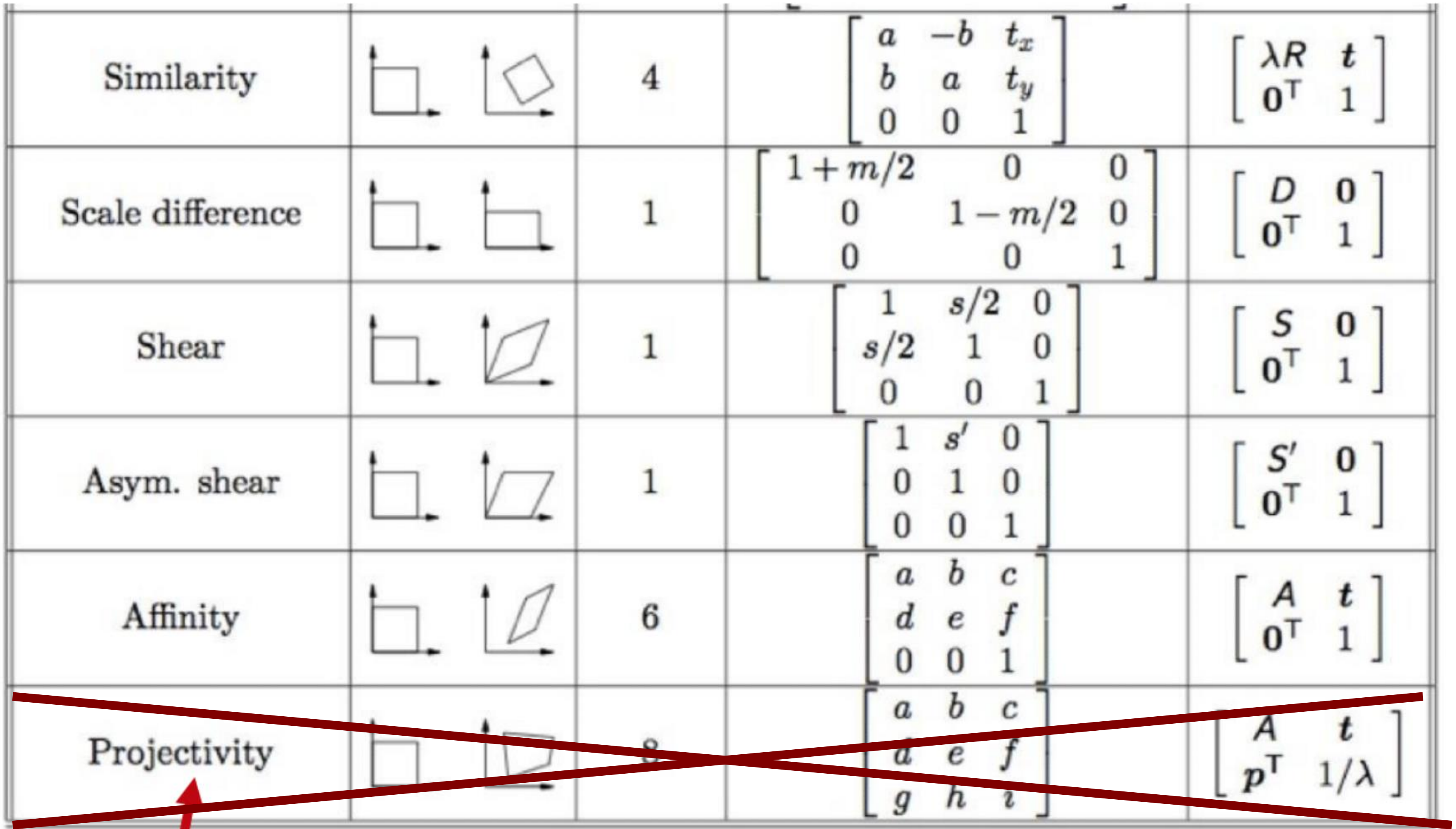
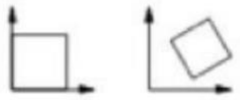
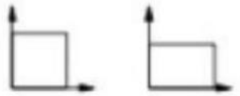
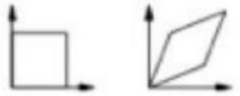
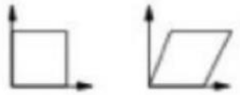
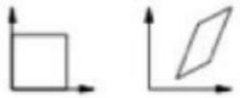
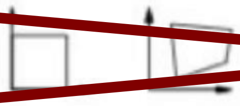
2D Transformation	Figure	d. o. f.	H	H
Translation		2	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} I & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$
Mirroring at y-axis		1	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} Z & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$
Rotation		1	$\begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$
Motion		3	$\begin{bmatrix} \cos\varphi & -\sin\varphi & t_x \\ \sin\varphi & \cos\varphi & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$

Image courtesy: Schindler

2D Motion Models

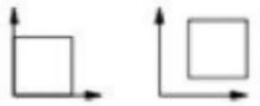
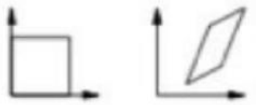


Similarity		4	$\begin{bmatrix} a & -b & t_x \\ b & a & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \lambda R & t \\ 0^\top & 1 \end{bmatrix}$
Scale difference		1	$\begin{bmatrix} 1+m/2 & 0 & 0 \\ 0 & 1-m/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} D & 0 \\ 0^\top & 1 \end{bmatrix}$
Shear		1	$\begin{bmatrix} 1 & s/2 & 0 \\ s/2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} S & 0 \\ 0^\top & 1 \end{bmatrix}$
Asym. shear		1	$\begin{bmatrix} 1 & s' & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} S' & 0 \\ 0^\top & 1 \end{bmatrix}$
Affinity		6	$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} A & t \\ 0^\top & 1 \end{bmatrix}$
~~Projectivity~~		~~8~~	$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$	$\begin{bmatrix} A & t \\ p^\top & 1/\lambda \end{bmatrix}$

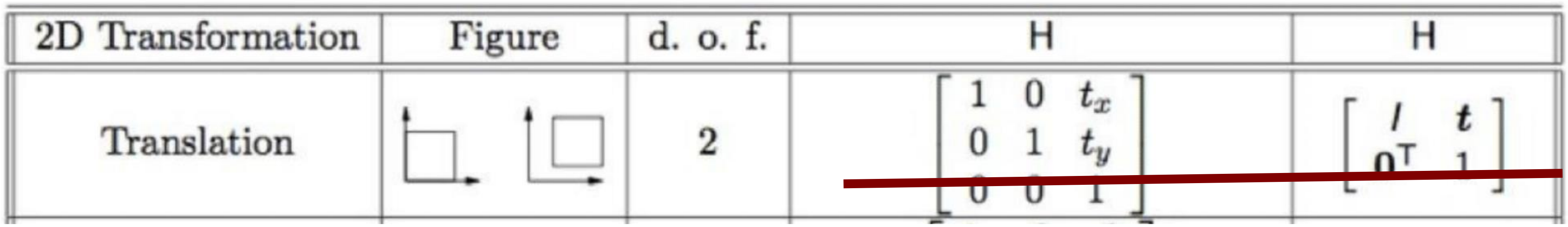
2D Homography

Image courtesy: Schindler

2D Motion Models

2D Transformation	Figure	d. o. f.	H	H
Translation		2	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ \cancel{0} & \cancel{0} & \cancel{1} \end{bmatrix}$	$\begin{bmatrix} I & t \\ \cancel{0^T} & \cancel{1} \end{bmatrix}$
Affinity		6	$\begin{bmatrix} a & b & c \\ d & e & f \\ \cancel{0} & \cancel{0} & \cancel{1} \end{bmatrix}$	$\begin{bmatrix} A & t \\ \cancel{0^T} & \cancel{1} \end{bmatrix}$

2D Motion Models

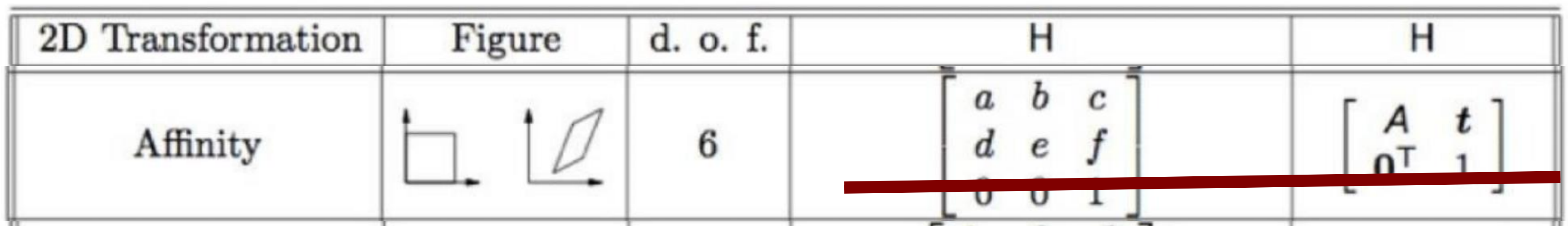


2D Transformation	Figure	d. o. f.	H	H
Translation		2	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix}$

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

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

2D Motion Models



2D Transformation	Figure	d. o. f.	H	H
Affinity		6	$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$

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

$$W = \begin{pmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$p = \begin{pmatrix} a_1 & a_2 & b_1 & a_3 & a_4 & b_2 \end{pmatrix}^T$$

Derivatives: Jacobian

Suppose $f : \mathbf{R}^n \to \mathbf{R}^m$ is differentiable. Its *derivative* or *Jacobian* at a point $x \in \mathbf{R}^n$ is denoted $Df(x) \in \mathbf{R}^{m\times n}$, defined as

$$(Df(x))_{ij} = \left.\frac{\partial f_i}{\partial x_j}\right|_x, \quad i = 1,\ldots,m, \quad j = 1,\ldots,n.$$

$$\begin{bmatrix} \frac{\partial f(x)_1}{\partial x_1} & \cdots & \frac{\partial f(x)_1}{\partial x_n} \\ \vdots & & \\ \frac{\partial f(x)_m}{\partial x_1} & \cdots & \frac{\partial f(x)_m}{\partial x_n} \end{bmatrix}^{m \times n}$$

Derivatives: Gradient

For $f : \mathbf{R}^n \to \mathbf{R}$, the *gradient* at $x \in \mathbf{R}^n$ is denoted $\nabla f(x) \in \mathbf{R}^n$, and it is defined as $\nabla f(x) = Df(x)^T$, the transpose of the derivative. In terms of partial derivatives, we have

$$\nabla f(x)_i = \left.\frac{\partial f}{\partial x_i}\right|_x, \quad i = 1, \ldots, n.$$

$$\left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right)^{1 \times n}$$

Taylor Series: Vector Valued f

Suppose $f : \mathbf{R}^n \to \mathbf{R}^m$ is differentiable. Its *derivative* or *Jacobian* at a point $x \in \mathbf{R}^n$ is denoted $Df(x) \in \mathbf{R}^{m\times n}$, defined as

$$(Df(x))_{ij} = \left.\frac{\partial f_i}{\partial x_j}\right|_x, \quad i = 1,\ldots,m, \quad j = 1,\ldots,n.$$

The first order Taylor expansion of f at (or near) x is given by

$$\hat{f}(y) = f(x) + Df(x)(y - x).$$

When $y - x$ is small, $f(y) - \hat{f}(y)$ is very small. This is called the *linearization* of f at (or near) x.

Taylor Series: Vector Valued f

The first order Taylor expansion of f at (or near) x is given by

$$\hat{f}(y) = f(x) + Df(x)(y - x).$$

As an example, consider $n = 3$, $m = 2$, with

$$f(x) = \begin{bmatrix} x_1 - x_2^2 \\ x_1 x_3 \end{bmatrix}.$$

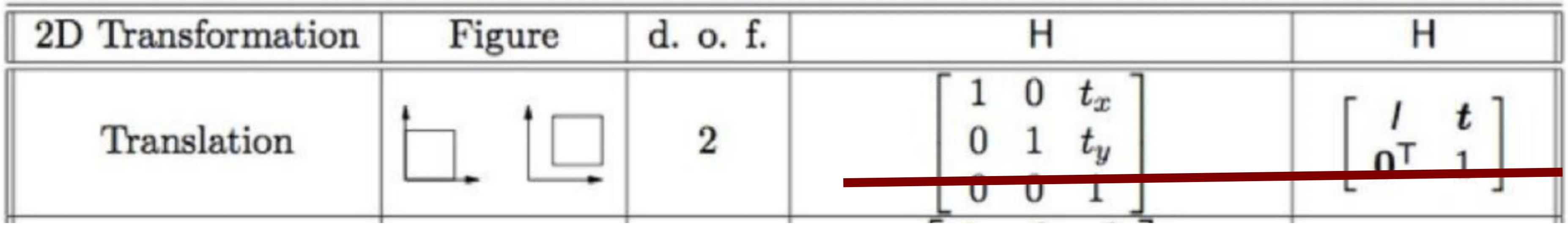
Its derivative at the point x is

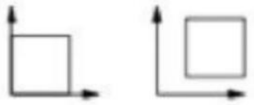
$$Df(x) = \begin{bmatrix} 1 & -2x_2 & 0 \\ x_3 & 0 & x_1 \end{bmatrix},$$

and its first order Taylor expansion near $x = (1, 0, -1)$ is given by

$$\hat{f}(y) = \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \left(y - \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \right).$$

2D Motion Models



2D Transformation	Figure	d. o. f.	H	H
Translation		2	$\begin{bmatrix}1 & 0 & t_x\\0 & 1 & t_y\\0 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}I & \boldsymbol{t}\\0^T & 1\end{bmatrix}$

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

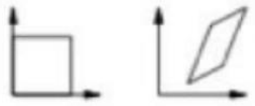
$$\begin{pmatrix}x'\\y'\end{pmatrix} = \begin{pmatrix}1 & 0 & b_1\\0 & 1 & b_2\end{pmatrix}\begin{pmatrix}x\\y\\1\end{pmatrix} \quad W(\boldsymbol{x};\boldsymbol{p}) = \begin{pmatrix}1 & 0 & b_1\\0 & 1 & b_2\end{pmatrix}\begin{pmatrix}x\\y\\1\end{pmatrix}$$

$$p = \begin{pmatrix}b_1\\b_2\end{pmatrix}$$

Taking the partial derivative of W with respect to $\boldsymbol{p}$ we get:

$$\frac{\partial W}{\partial \boldsymbol{p}}(\boldsymbol{x};\boldsymbol{p}) = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$$

2D Motion Models

2D Transformation	Figure	d. o. f.	H	H
Affinity		6	$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$

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

$$W = \begin{pmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$
$$p = (a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2)^T$$

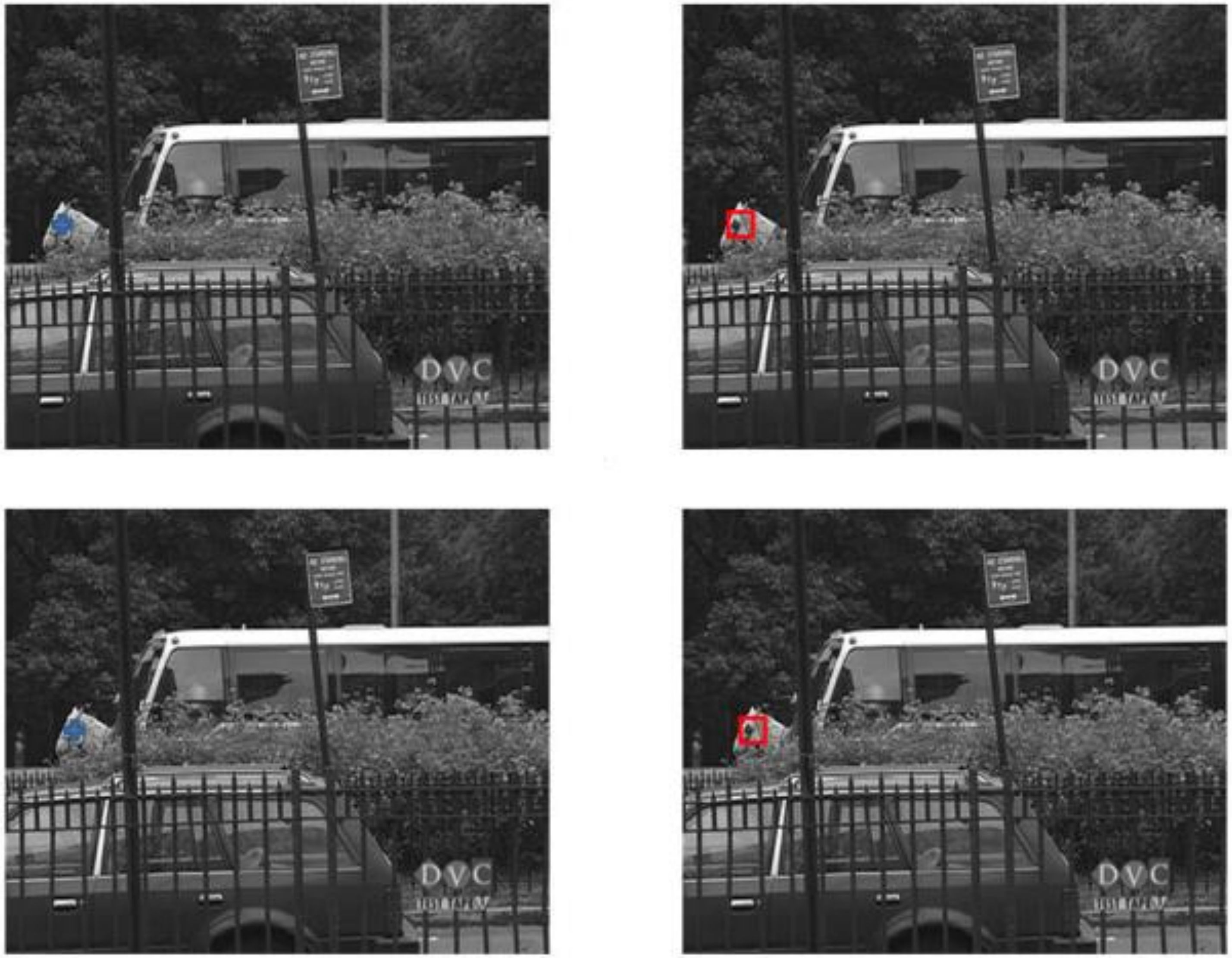
Jacobian for affine motion is the following:

$$\frac{\partial W}{\partial \boldsymbol{p}}(\boldsymbol{x};\boldsymbol{p}) = \begin{pmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{pmatrix}$$

Motion Estimation

- $T(\cdot)$ is the template, $I(\cdot)$ is the image. Takes *x*,*y* and gives us the intensity at *x*,*y*.
- Consider $T(\boldsymbol{x}) = I(W(\boldsymbol{x}_t; \boldsymbol{p}))$ where W is given by translation (warp the template)
- We want to estimate **p**
- But estimating **p** at individual pixels is unreliable due to image noise and the aperture problem.
- So consider a small patch of pixels around **x**. Estimate (and assume) a common **p** for the entire window.

Using a Patch instead of a Point



Motion Estimation

- We want to find the transform **p** that minimizes the error of the feature description around $\boldsymbol{x} = W(\boldsymbol{x}_t; \boldsymbol{p})$ (your hypothesis for where the feature's new location is) in the next frame. In other words, the below expression wrt $\boldsymbol{p}$

$$\sum_x [I(W(x;p)) - T(x)]^2$$

- Assume that you have an initial estimate $\boldsymbol{p_0}$ for $\boldsymbol{p}$ and $\boldsymbol{p} = \boldsymbol{p_0} + \boldsymbol{\Delta p}$

Motion Estimation

$$E = \sum_x \left[I\left(W(x;p)\right) - T(x)\right]^2 = \sum_x \left[I\left(W(x;p_0 + \Delta p)\right) - T(x)\right]^2$$

- Using the Taylor approximation, we see that this error term is roughly equal to

$$E \approx \sum_x \left[I\left(W(x;p_0)\right) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x)\right]^2$$

- To minimize this term, we take the derivative with regard to Δp and set it equal to 0, then solve for Δp.

Motion Estimation

$$E \approx \sum_x \left[I\left(W(x;p_0)\right) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right]^2$$

- To minimize this term, we take the derivative with regard to Δp and set it equal to 0, then solve for Δp

$$\frac{\partial E}{\partial p} \approx \sum_x \left[I\left(W(x;p_0)\right) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right] \left[\nabla I \frac{\partial W}{\partial p}\right] = \mathbf{0}$$

1xd 1x1 1x2 2xd dx1 1x1 1xd

d is the dimension of *p*

Motion Estimation

- We have Δp as the only unknown in the following equation:

$$\frac{\partial E}{\partial p} \approx \sum_x \left[I\big(W(x;p_0)\big) + \nabla I \frac{\partial W}{\partial p}\,\Delta p \; - T(x) \right] [\nabla I \; \frac{\partial W}{\partial p}] = \mathbf{0}$$

$$\sum_x \left[\nabla I \frac{\partial W}{\partial p}\,\Delta p \; \nabla I \frac{\partial W}{\partial p} \right] = \sum_x \left[T(x) \; - I\big(W(x;p_0)\big) \right] \left[\nabla I \; \frac{\partial W}{\partial p} \right]$$

1x1 1xd

$$\sum_x \left[\Delta p^T \left(\nabla I \frac{\partial W}{\partial p} \right)^T \; \nabla I \frac{\partial W}{\partial p} \right] = \sum_x \left[T(x) \; - I\big(W(x;p_0)\big) \right] \left[\nabla I \; \frac{\partial W}{\partial p} \right]$$

Motion Estimation

- We have Δp as the only unknown in the following equation:

$$\sum_x \left[\Delta p^T \left(\nabla I \frac{\partial W}{\partial p}\right)^T \nabla I \frac{\partial W}{\partial p}\right] = \sum_x [T(x) - I(W(x; p_0))] \left[\nabla I \frac{\partial W}{\partial p}\right]$$

$$\Delta p^T \sum_x \left[\left(\nabla I \frac{\partial W}{\partial p}\right)^T \nabla I \frac{\partial W}{\partial p}\right] = \sum_x [T(x) - I(W(x; p_0))] \left[\nabla I \frac{\partial W}{\partial p}\right]$$

$$\Delta p^T = \left\{\sum_x [T(x) - I(W(x; p_0))] \left[\nabla I \frac{\partial W}{\partial p}\right]\right\} * H^{-1}$$

Where, $H = \sum_x \left[\left(\nabla I \frac{\partial W}{\partial p}\right)^T \nabla I \frac{\partial W}{\partial p}\right]$ Structure Tensor

Motion Estimation

- To minimize this term, we take the derivative with regard to $\boldsymbol{p_0}$ and set it equal to 0, then solve for $\boldsymbol{p_0}$

$$\Delta p^T = \left\{ \sum_x \left[T(x) - I\big(W(x;p_0)\big) \right] \left[\nabla I \frac{\partial W}{\partial p} \right] \right\} * H^{-1}$$

- By iteratively setting $\boldsymbol{p} = \boldsymbol{p_0} + \boldsymbol{\Delta p}$, we can eventually converge on an accurate, error-minimizing value of $\boldsymbol{p}$, which tells us what the transform is.

Link to Harris Corner Detection

- For translation Motion $\frac{\partial W}{\partial p}(x;p) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

- Reminder, gradient vector for the image template

$$\nabla I(x,y) = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} & \frac{\partial I(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} I_x & I_y \end{bmatrix}$$ Gradient (row-vector)

- So for translation motion model, we can easily verify

$$\sum_x [\nabla I \frac{\partial W}{\partial p}]^T [\nabla I \frac{\partial W}{\partial p}] = H = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$

Good Features to Track

Challenges

- Figure out which features can be tracked
- Efficiently track across frames – Some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)
- Drift: small errors can accumulate as appearance model is updated
- Points may appear or disappear: need to be able to add/delete tracked points

Good Features to Track

- One that can be tracked well.

- A point whose surrounding patch has nice properties!

$$H = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$

- Structure tensor H should be invertible

Good Features to Track

- Let the two eigenvalues of H be $\lambda_1, \lambda_2 (\lambda_1 > \lambda_2)$
- If the patch has constant intensity, $\lambda_1 = \lambda_2 = 0$. H is a null matrix.
- If the patch is noisy but otherwise has constant intensity, $\lambda_1 \approx \lambda_2 \approx 0$. H is ill-conditioned.
- If the patch has constant but large gradient, then $\lambda_1 \gg 0, \lambda_2 0$. H has rank 1.
- If the patch has constant but large gradient and some noise, then $\lambda_1 \gg 0, \lambda_2 \approx 0$. H has rank 2 but is ill-conditioned.

Code at:

http://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2015/eigvals_structure_tensor.m

Good Features to Track

- If the patch contains a strong edge against a constant intensity background, then $\lambda_1 \gg 0, \lambda_2 \approx 0$. H has rank 2 but is ill-conditioned.
- If the patch is rich in texture with significant intensity, then $\lambda_1 \gg 0, \lambda_2 \gg 0$. This also occurs at corner points. **H has rank 2 and is well conditioned. It allows for reliable motion estimation**.
- The minor eigen-value gives a good idea of the "goodness" of the feature point for tracking. Larger is better.

Code at:

http://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2015/eigvals_structure_tensor.m

Good Features to Track

- Pick feature points whose minor eigen-value exceeds some threshold τ
- How to pick τ?
- Take a validation set consisting of uniform intensity patches and another of textured patches.
- The minor eigen values will have a huge difference. That gives you a good estimate of τ

Code at:

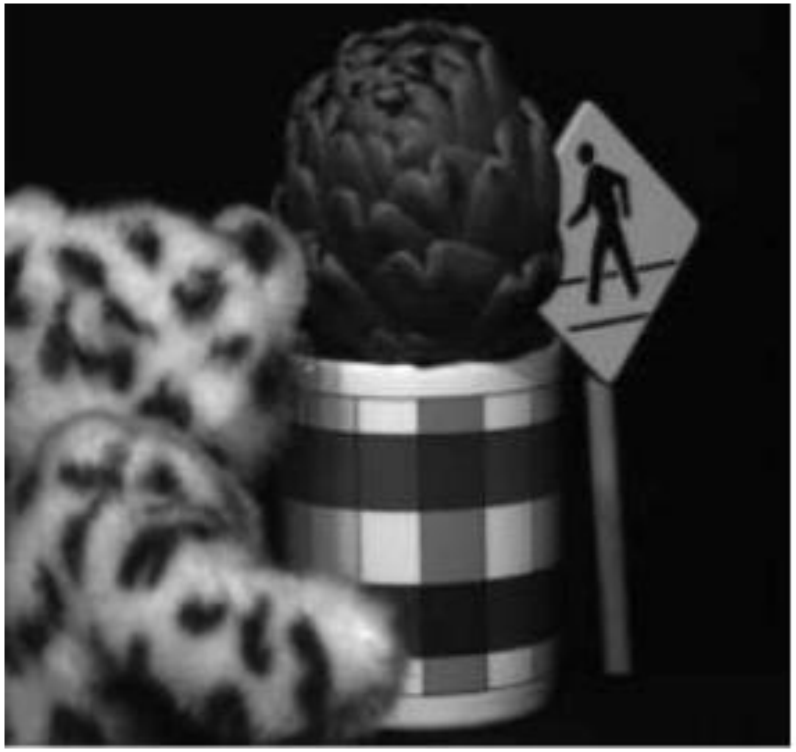
http://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2015/eigvals_structure_tensor.m

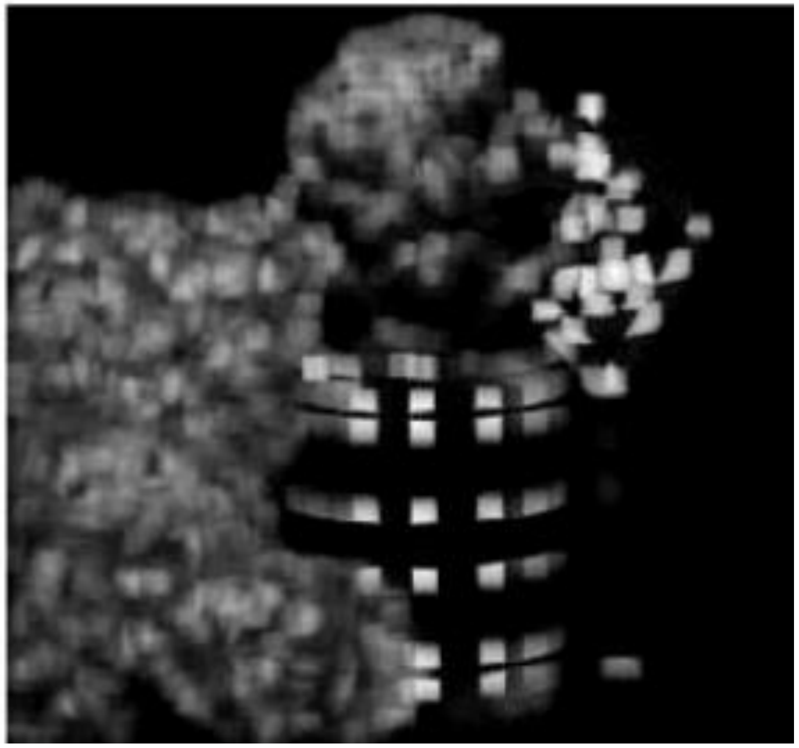
Good Features to Track

Code at:

http://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2015/eigvals_structure_tensor.m

Good Features to Track





Code at:

http://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2015/eigvals_structure_tensor.m

Motion Models:
1. Translation and
2. Affine

- The assumption of a single displacement/translation **p** for the whole patch can be unreasonable.
- Solution: Use an affine transformation model – each pixel in the patch now has its own displacement.

Motion Models:
1. Translation and
2. Affine

- Model 2 has more degrees of freedom than Model 1 (6 versus 2).
- Model 2 is more general, but less robust to noise.
- Model 2 will need larger patches to work well.
- This increases the fraction of patches that straddle boundaries of different objects.

Motion Models:
1. Translation and
2. Affine

- Motion between consecutive frames must be small (necessary for tracking to work) – hence most of the entries of W will be small.
- Hence model 1 is preferred for motion estimation between consecutive frames.
- Model 2 has another application in tracking! (we will short see – good feature to track)

Good Features to Track:

- Compute a dissimilarity measure between a patch containing the (tracked) feature point in the current frame $I(\boldsymbol{x}_{t+n})$ and the original patch $T(\boldsymbol{x})$ in the first frame.
- If the measure is small, then we have tracked the point well.
- If the measure is large, the tracking has failed, and this feature point should be abandoned.

Good Features to Track:

- Dissimilarity measure 1
  - Find the best translation between $I(\boldsymbol{x}_{t+n})$ and $T(\boldsymbol{x})$
  - Find squared difference between $T(\boldsymbol{x})$ and translated $W(I(\boldsymbol{x}_{t+n}); \boldsymbol{p})$
- Dissimilarity measure 2
  - Find the best affine transformation between $I(\boldsymbol{x}_{t+n})$ and $T(\boldsymbol{x})$
  - Find squared difference between $T(\boldsymbol{x})$ and affine transformed $W(I(\boldsymbol{x}_{t+n}); \boldsymbol{p})$

Synthetic Results: Motion Estimation

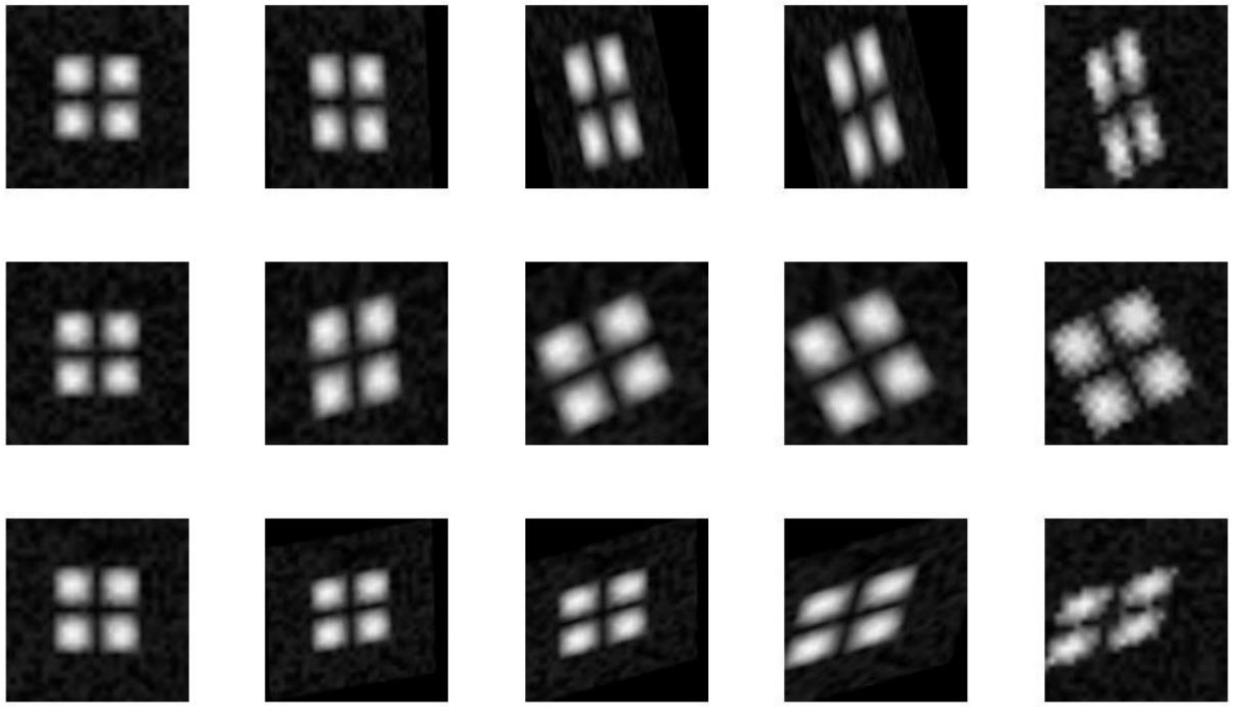


Figure 6.1: Original image (leftmost column) and image warped, translated and corrupted by noise (rightmost column) for three different motions. The intermediate columns are the images in the leftmost column deformed and translated by 4,8,and 19 iterations of the tracking algorithm.

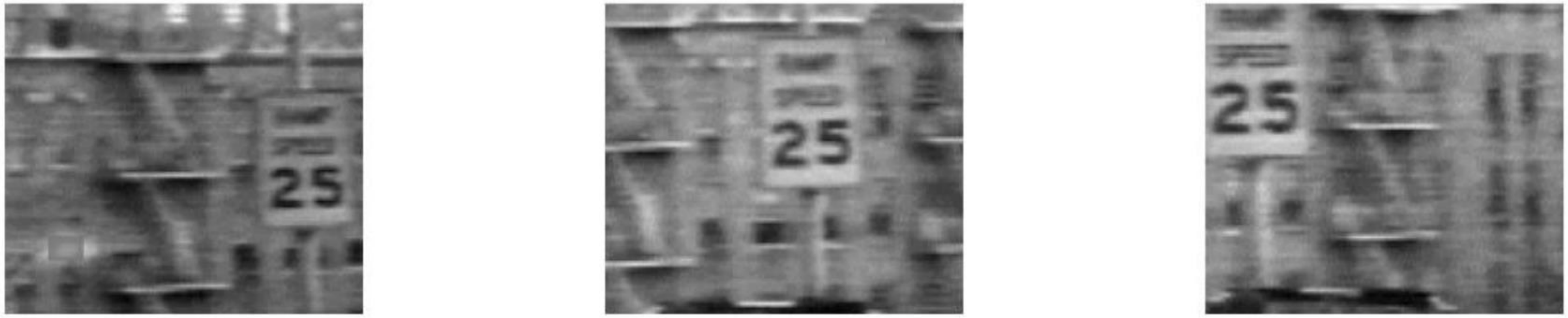


Figure 5.1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

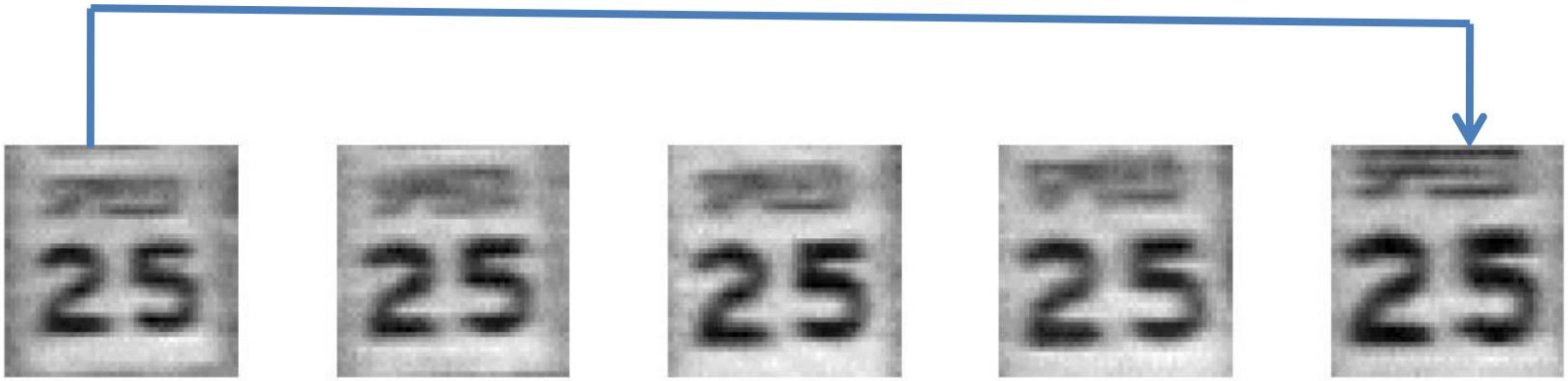


Figure 5.2: A window that tracks the traffic sign visible in the sequence of figure 5.1. Frames 1,6,11,16,21 are shown here.

Simple translation model will fail here!

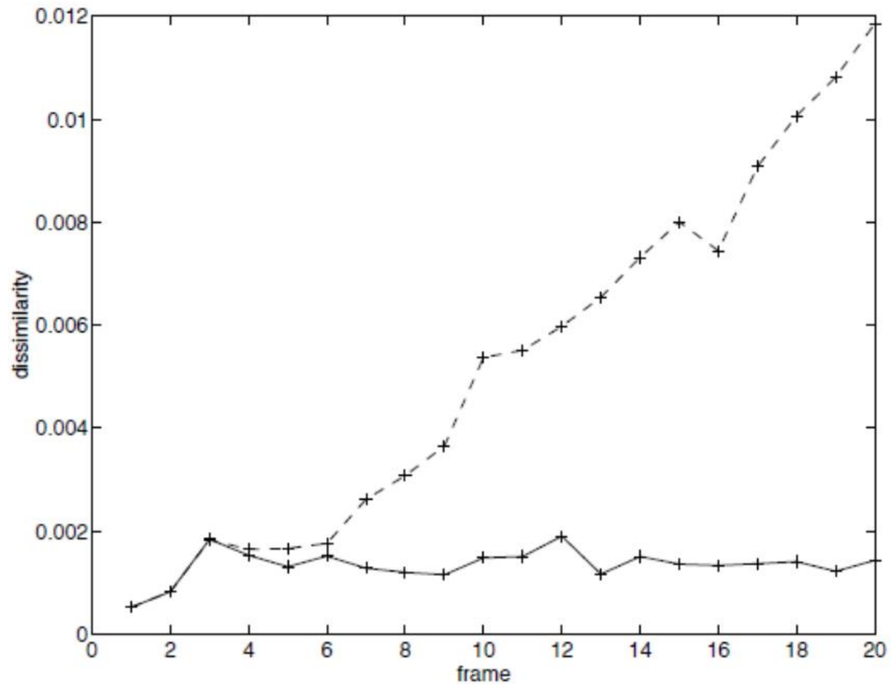


Figure 5.3: Pure translation (dashed) and affine motion (solid) dissimilarity measures for the window sequence of figure 5.2.

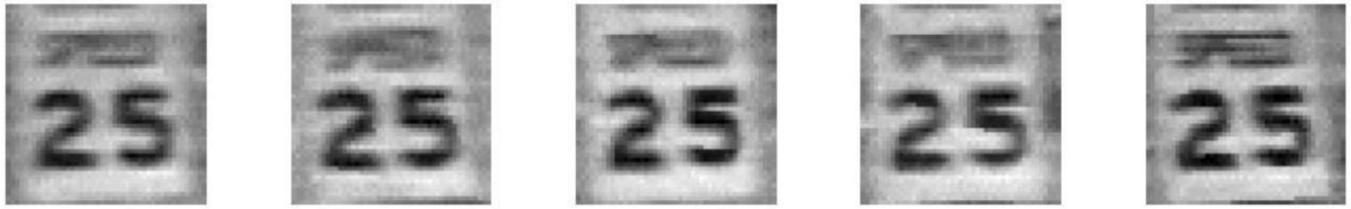


Figure 5.4: The same windows as in figure 5.2, warped by the computed deformation matrices.

Effect of Occlusion

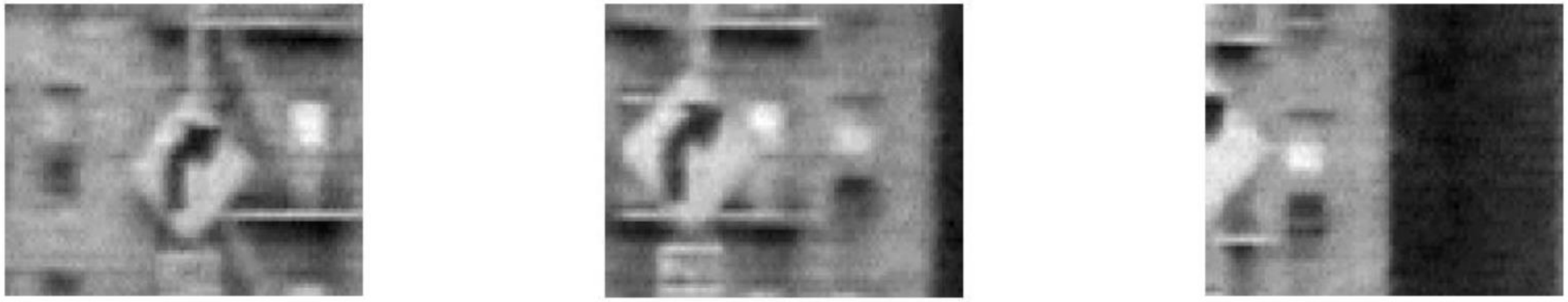


Figure 5.5: Three frame details from Woody Allen's *Manhattan*. The feature tracked is the bright window on the background, just behind the fire escape on the right of the traffic sign.

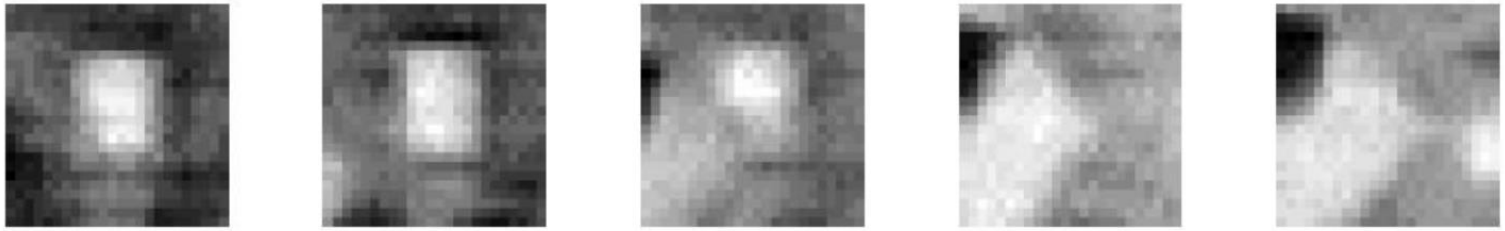


Figure 5.6: The bright window from figure 5.5, visible as the bright rectangular spot in the first frame (a), is occluded by the traffic sign (c).

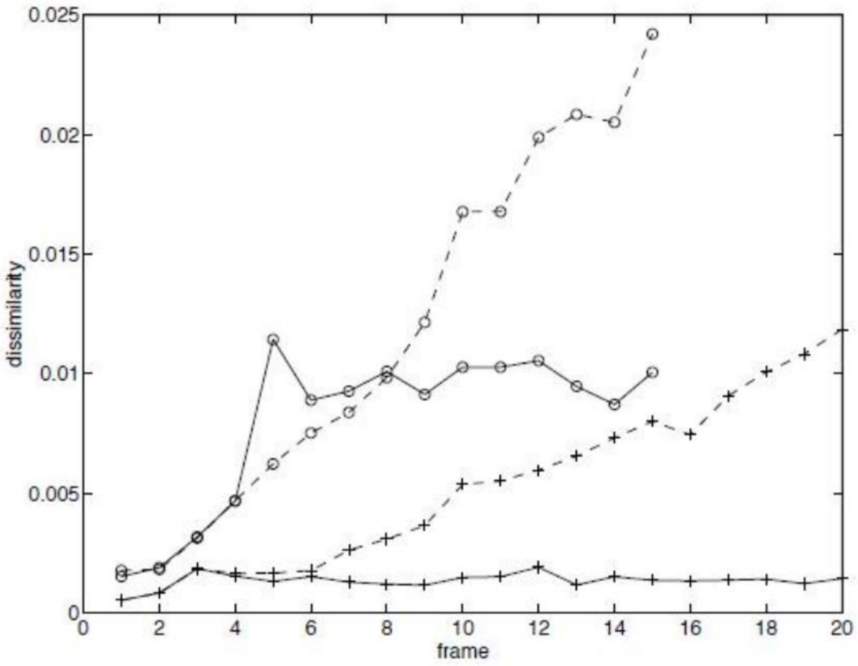


Figure 5.7: Pure translation (dashed) and affine motion (solid) dissimilarity measures for the window sequence of figure 5.1 (plusses) and 5.5 (circles).

Figure 5.8: The same windows as in figure 5.6, warped by the computed deformation matrices.

Experiments with a Real Sequence

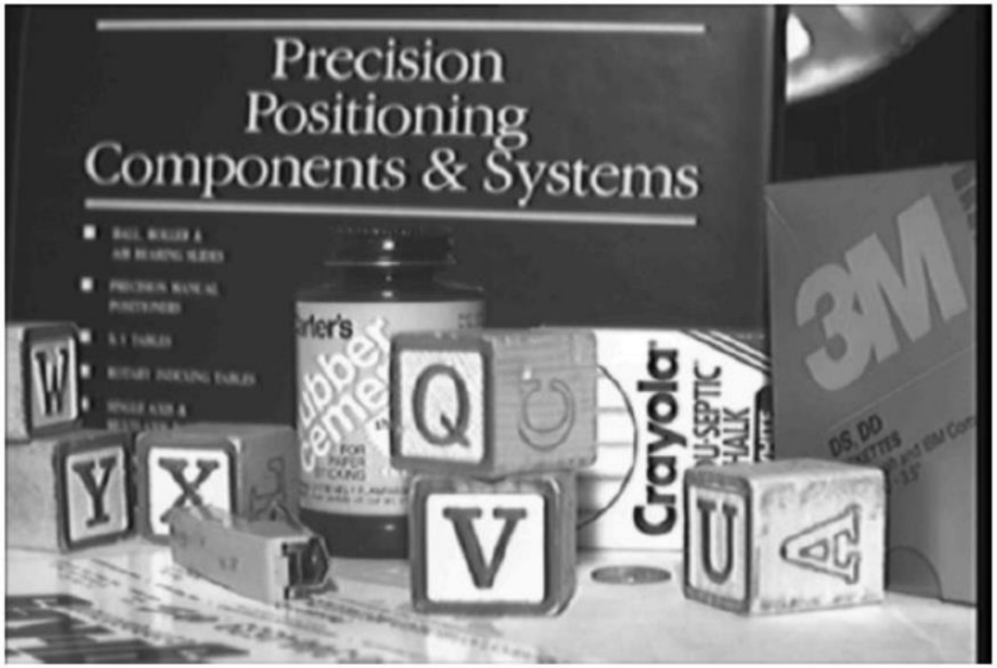


Figure 7.1: The first frame of a 26 frame sequence taken with a forward moving camera.

Camera moving forward at the rate of 2 mm per frame. Focal length of lens is 16 mm.

Experiments with a Real Sequence

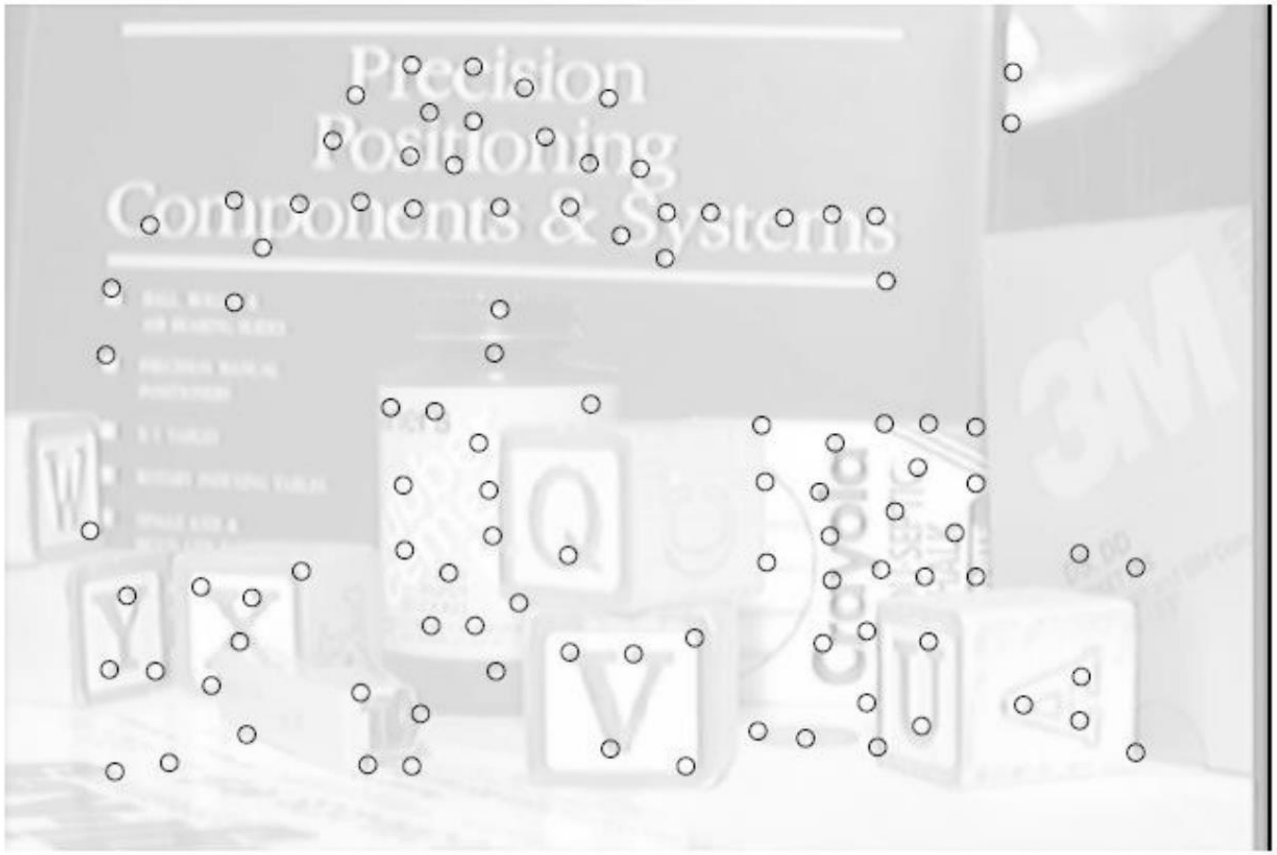


Figure 7.2: The features selected according to the texturedness criterion of chapter 4.

Experiments with a Real Sequence

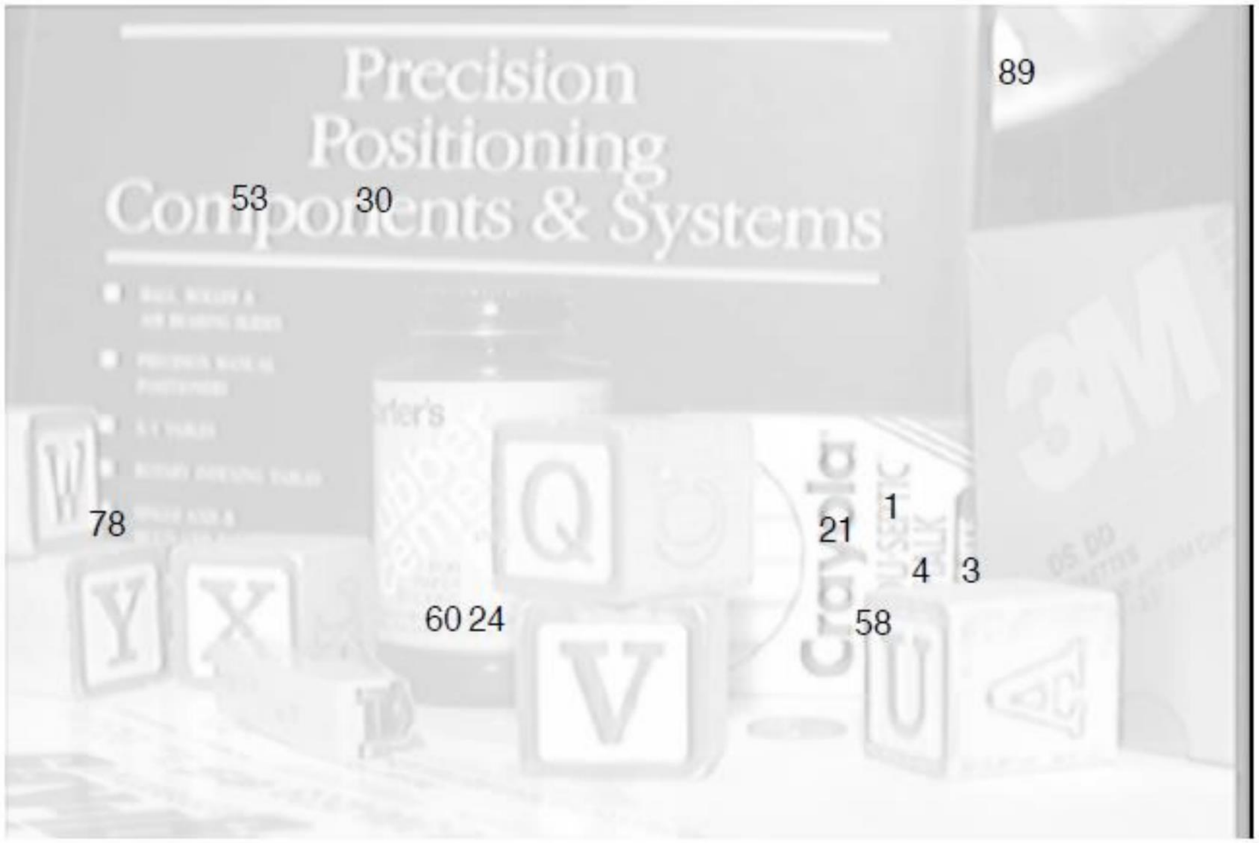


Figure 7.4: Labels of some of the features in figure 7.2.

Experiments with a Real Sequence

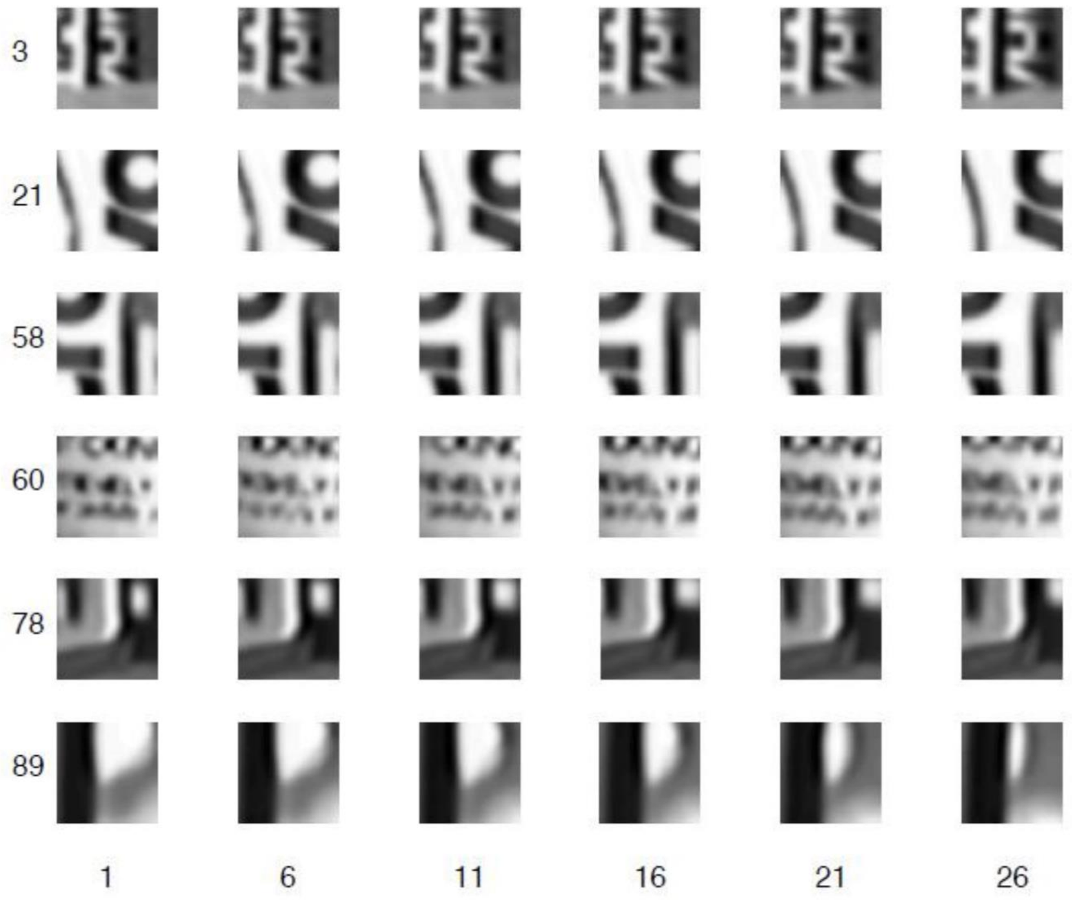


Figure 7.5: Six sample features through six sample frames.

Experiments with a Real Sequence

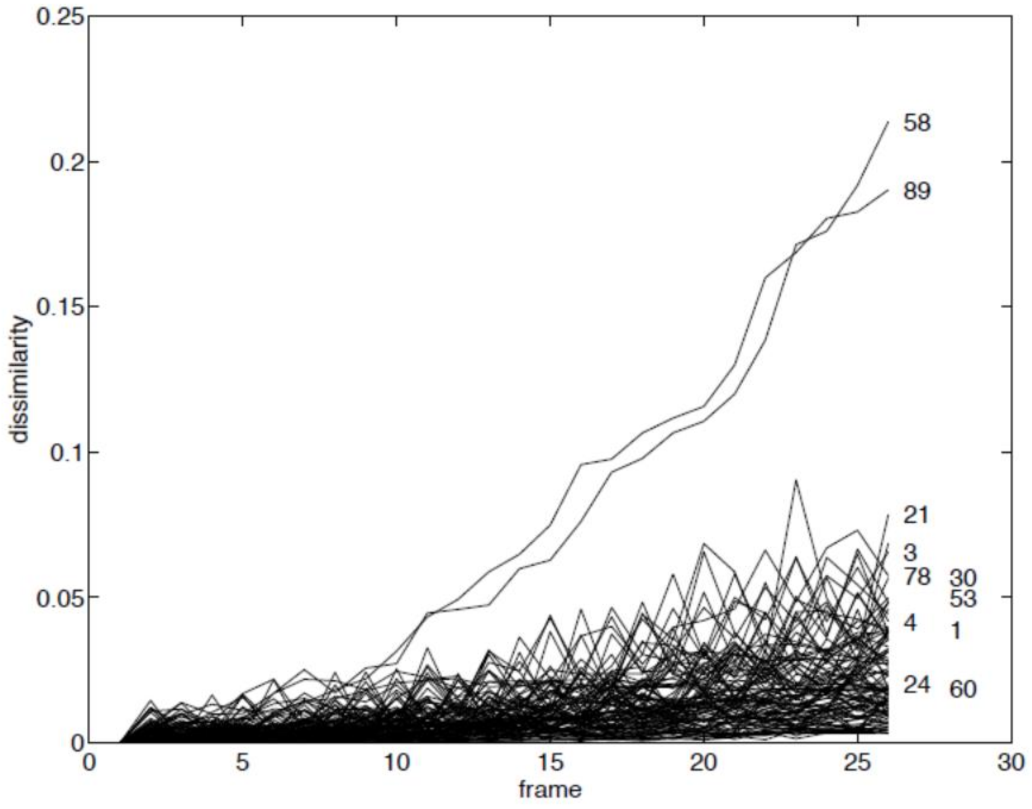


Figure 7.3: Pure translation dissimilarity for the features in figure 7.2. This dissimilarity is nearly useless for feature discrimination.

Experiments with a Real Sequence

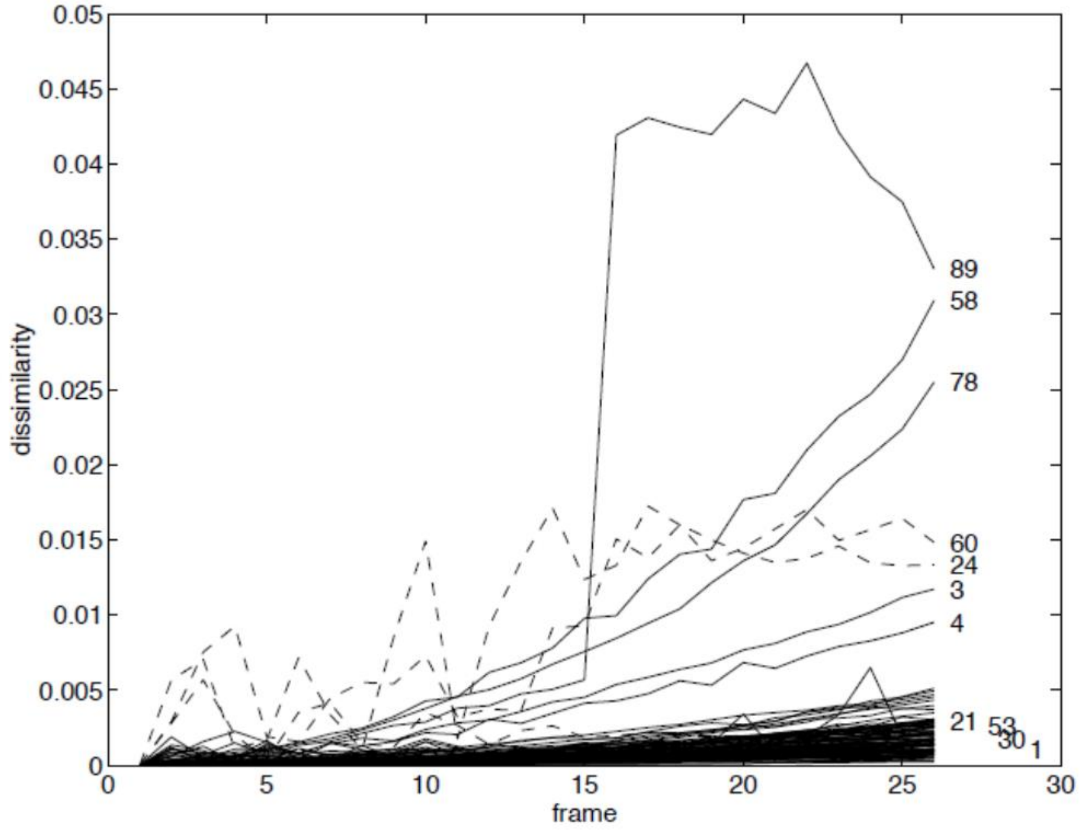


Figure 7.6: Affine motion dissimilarity for the features in figure 7.2. Notice the good discrimination between good and bad features. Dashed plots indicate aliasing (see text).

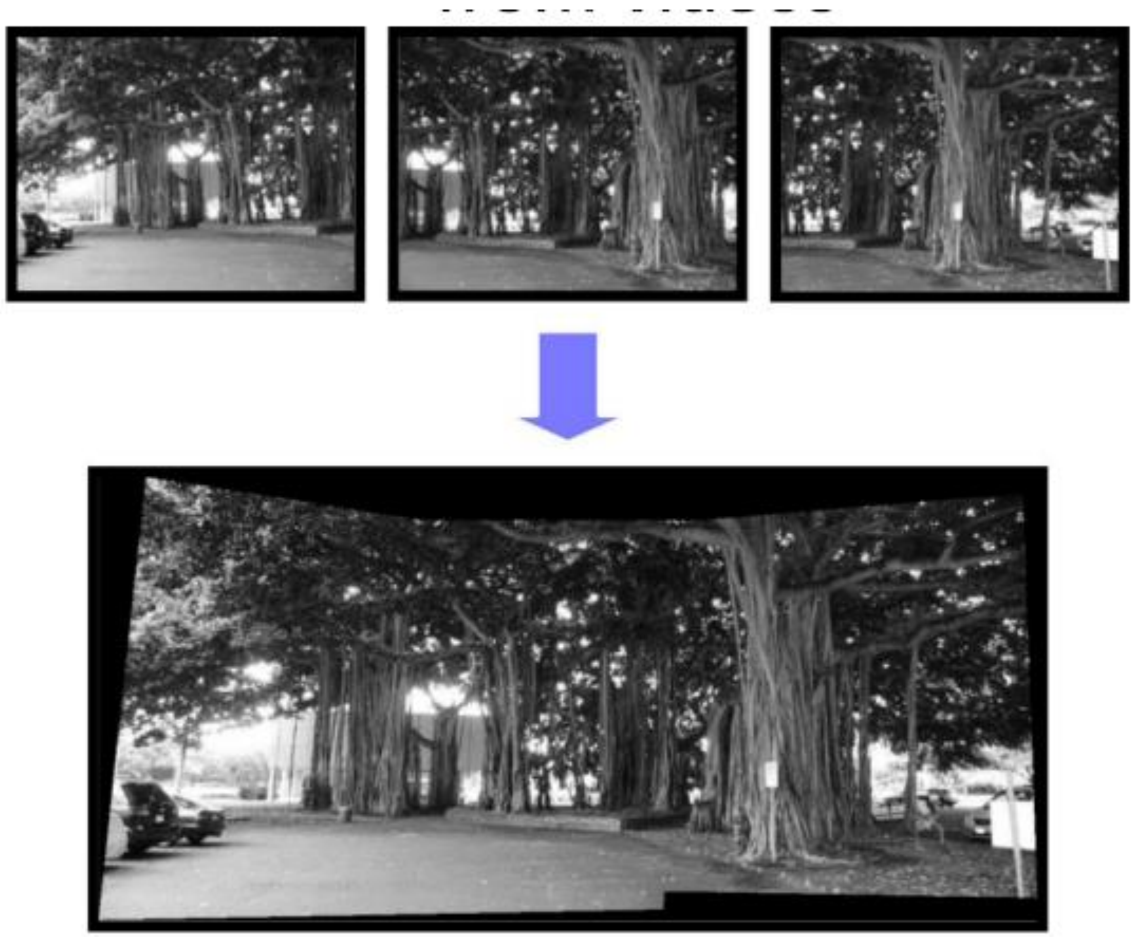
KLT Tracker: Appearance of new Objects

- When a new object appears in the scene, it yields new feature points which will be ignored by the present version of the algorithm.
- Can be alleviated by running a feature detector (our minor eigenvalue method) on every k-th frame of the video and initializing new feature points (take care to ensure they aren’t too close to any existing tracked points).

KLT Tracker: Algorithm

1. Use the minor eigen-value method to detect salient feature points in frame 1.
2. For each feature at location **x**, create templates $T(x)$ around feature points in frame t and determine the motion **p**
3. Check for validity of tracking using the affine model (Model 2) based dissimilarity measure. If the measure falls above a certain threshold, discard the feature point, i.e. don’t track it any further.
4. Every 10-15 frames introduce new Harris corners to account for occlusion and "lost" features.
5. Repeat for t=1 to T.

Applications – Panorama Generation from Videos

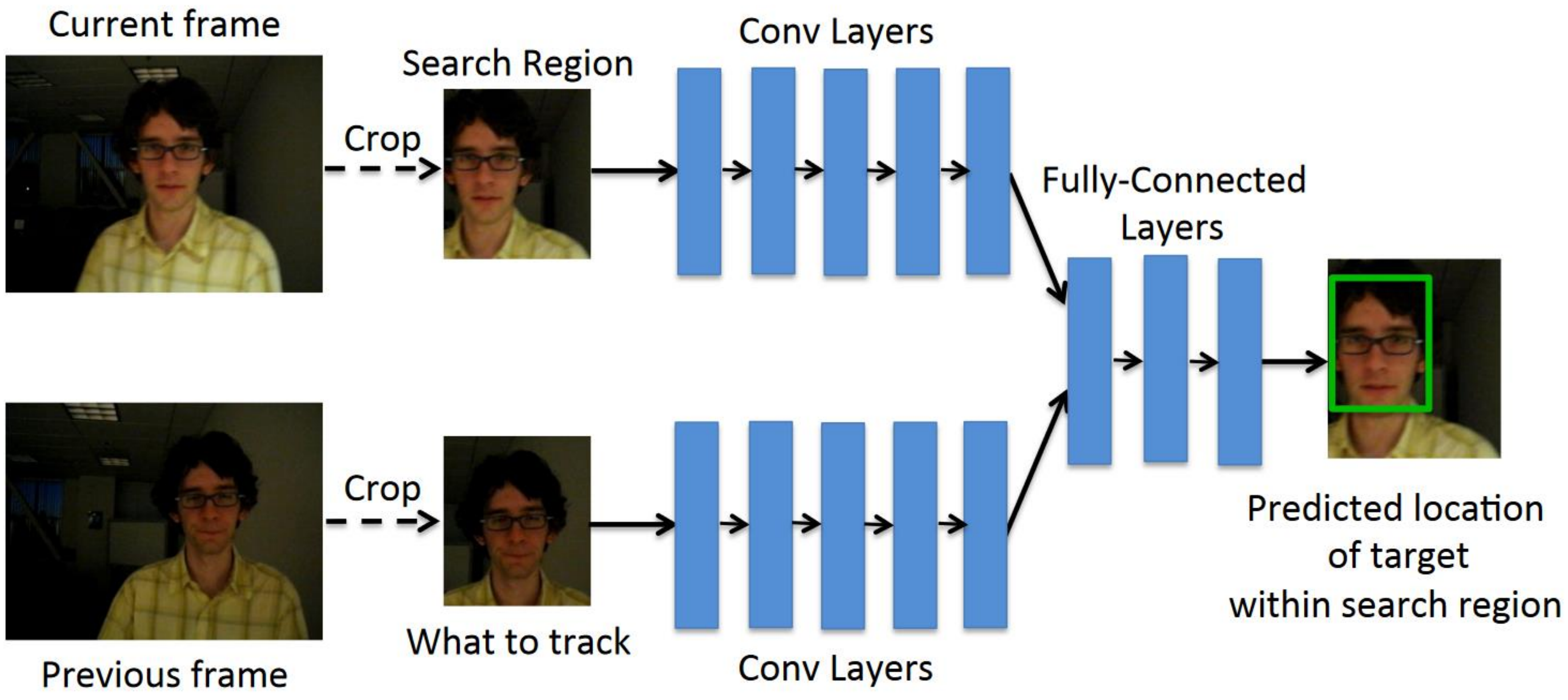


http://www.infotech.oulu.fi/Annual/2005/pics/mvg_01.jpg

ConvNet based Tracking

- GOTURN (Generic Object Tracking Using Regression Networks) - Learning to Track at 100 FPS with Deep Regression Networks (ECCV 2016, Held et. al.)

ConvNet based Tracking: Architechture



ConvNet based Tracking: at test time

- Suppose at frame $t-1$ the position of object is $\mathrm{c} = (c_x, c_y)$, with size (h, w)
- At time t take a crop centred at (c_x, c_y), with size $(k_1 h, k_1 w)$ from frame $t-1$
  - This crop tells the network which object is being tracked. The value of k_1 determines how much context the network will receive about the target object from the previous frame.

ConvNet based Tracking: at test time

- Crop the current frame t centred at $c' = (c_x', c_y')$, with size $(k_2 h, k_2 w)$
  - For author's model $k_1 = k_2 = 2$ and $c' = c$ that is constant position motion model, more sophisticated motion models can be used.
  - For fast-moving objects, the size of the search region could be increased, at a cost of increasing the complexity of the network.
- Feed above two crops to the network.

ConvNet based Tracking: at training time

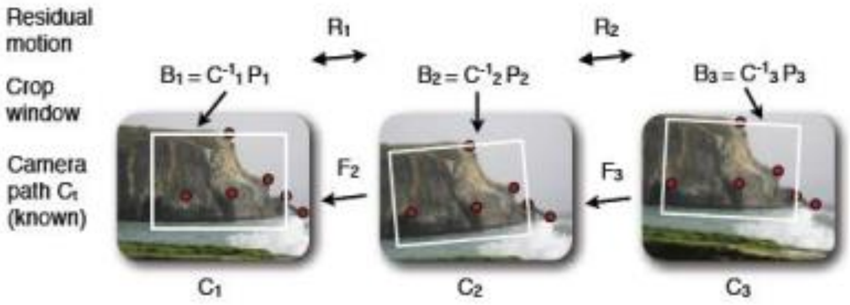
- Only need still images!
- Random crops of the images
- The convolutional layers of network are pre-trained on ImageNet
- L1 loss is used for loss function.

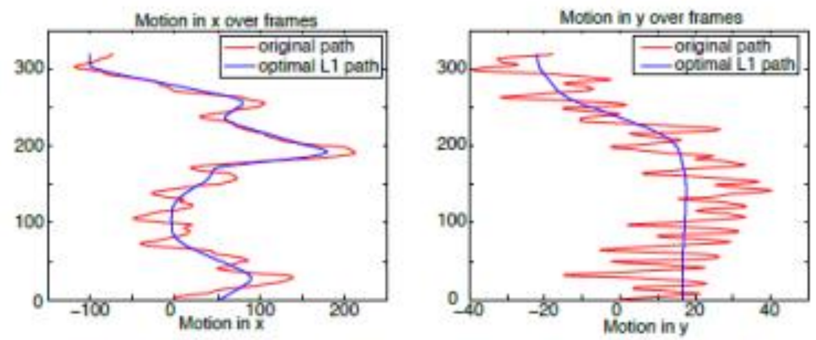
Applications – Video Stabilization

Applications – Video Stabilization

- Obtain salient points in first frame. Track them in the next k frames.
- Obtain affine/homography transformations between consecutive frames of the video.
- This will produce a sequence of transformation coefficients.
- Smooth this sequence.
- Warp the images as per new (smoothed) motion estimates – giving non-shaky video.

Applications – Video Stabilization





http://www.cc.gatech.edu/cpl/projects/videostabilization/

Grundmann, Kwatra, Essa, "Auto-directed video stabilization using robust L1 optimal paths", ICCV 2011.

Applications – Structure from Motion

- Input: video sequence (F frames) of an object undergoing rotational motion (assume orthographic projection)

- Aim: Given a sequence of P corresponding feature points (i.e. 2D coordinates) in all images, determine (1) the 3D coordinates of all the points, and (2) the rotational motion at each time instant.

Applications

- Moving object detection
- Mosaic generation
- Tracking

Slide Information

- The slide template has been created by Cyrill Stachniss (cyrill.stachniss@igg.uni-bonn.de) as part of the photogrammetry and robotics courses.
- A lot of material from Ajit Rajwade’s CS763 course
- **I tried to acknowledge all people from whom I used images or videos. In case I made a mistake or missed someone, please let me know.**

Arjun Jain, ajain@cse.iitb.ac.in