

Image and Video Processing

Global Motion Estimation and Object Tracking

Yao Wang
Tandon School of Engineering, New York University

Last Lecture

- 2-D motion vs. optical flow
- Optical flow equation and ambiguity in motion estimation
- General methodologies in motion estimation
 - Motion representation
 - Motion estimation criterion
 - Optimization methods
- Lucas-Kanade Flow Estimation Method and KLT tracker
- Block Matching Algorithm
 - EBMA algorithm
 - Half-pel EBMA
 - Hierarchical EBMA (HBMA)
- Deformable image registration (Skipped, optional)

Summary 1: General Methodology

- What causes 2D motion?
 - Object motion projected to 2D
 - Camera motion
 - Optical flow vs. true 2D motion
- Constraints for 2D motion
 - Optical flow equation
 - Derived from **constant intensity** and **small motion** assumption
 - Ambiguity in motion estimation
- Estimation criterion:
 - DFD (constant intensity)
 - OF (constant intensity+small motion)
 - Regularization (motion smoothness or other prior knowledge)
- Search method:
 - Exhaustive search, gradient-descent, multi-resolution
 - Least squares solution under optical flow equation and assuming motion is the same in a small neighborhood.

Summary 2: Motion Estimation Methods

- Pixel-based motion estimation (also known as optical flow estimation)
 - Most accurate representation, but also most costly to estimate
 - Need to put additional constraints on motion of neighboring pixels
 - Lucas-Kanade method
 - Assuming motion in the neighborhood is the same
 - Using Taylor expansion
 - How to handle large motion: iterative refinement, multiresolution
 - KLT tracker: apply LK method on feature points only
 - Automatically yield fractional accuracy
- Block-based motion estimation, assuming each block has a constant motion
 - Good trade-off between accuracy and speed
 - EBMA and its fast but suboptimal variants are widely used in video coding for motion-compensated temporal prediction.
 - HBMA can not only reduce computation but also yield physically more correct motion estimates

This Lecture

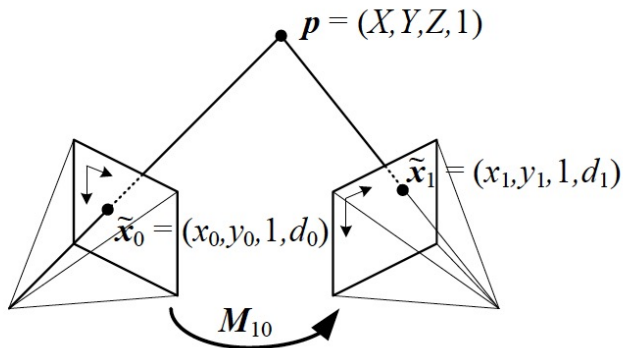
- • Global motion estimation
 - Direct estimation: global shift estimation
 - Indirect estimation: estimating global motion parameters from dense motion field
- Region-based motion estimation
- Video stabilization
- Moving object detection and background modeling
- Object tracking
 - Single object
 - Multiple object
 - Deep learning approaches
- Appendix (not required)
 - Gradient descent method for determining global motion
 - Background modeling and moving object detection: advanced methods
 - Video shot segmentation

Global Motion Estimation and Applications

- Camera movement induces a dense motion field that can be captured by a global motion function
- How to model the global motion due to common camera motion ?
- How to estimate the global motion?
- Why do we want to estimate the global motion?
 - Panoramic stitching
 - Medical image registration
 - Change detection
 - Video stabilization
 - Moving object detection
 - ...

Global Motion Model (Review)

- If two images F and G are taken of the same scene from different view points, they are related by a geometric mapping or transformation



\mathbf{x}_0 in F corresponds to \mathbf{x}_1 in G

Mapping function:

$$\mathbf{x}_1 = \mathbf{h}(\mathbf{x}_0)$$

or

$$x_1 = h_x(x_0, y_0), y_1 = h_y(x_0, y_0)$$

- What determines the mapping function?
 - Need to know camera 3D- \rightarrow 2D projection geometry
 - Need to know how to model camera motion

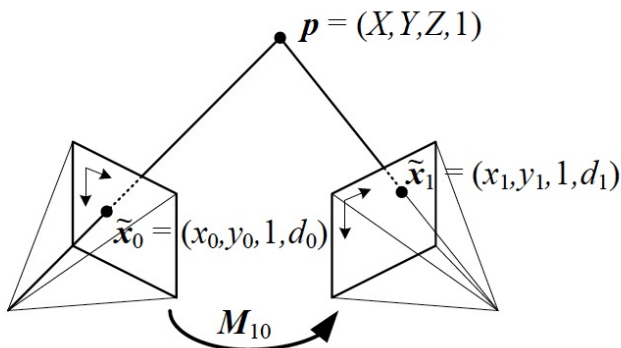
General Mapping Function

Assume left camera uses the world coordinate,

$$x_0 = FX / Z, y_0 = FY / Z,$$

Right camera is rotated and translated, so that $p=(X,Y,Z)^T$

appears as p' for camera 2



$$p' = \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = Rp + T$$

→
Perspective Projection

$$x_1 = FX' / Z', y_1 = FY' / Z'.$$

Can show:

$$x_1 = F \frac{(r_1 x_0 + r_2 y_0 + r_3 F)Z + T_x F}{(r_7 x_0 + r_8 y_0 + r_9 F)Z + T_z F}$$

$$y_1 = F \frac{(r_4 x_0 + r_5 y_0 + r_6 F)Z + T_y F}{(r_7 x_0 + r_8 y_0 + r_9 F)Z + T_z F}$$

The mapping function depends on Z and hence varies for every image point (corresponding to 3D points with different Z) in general.

Special Cases

- General relation

$$x' = F \frac{(r_1x + r_2y + r_3 F)Z + T_x F}{(r_7x + r_8y + r_9 F)Z + T_z F}$$

$$y' = F \frac{(r_4x + r_5y + r_6 F)Z + T_y F}{(r_7x + r_8y + r_9 F)Z + T_z F}$$
- Case 1: Arbitrary camera motion, but the scene is a flat surface ($Z = aX + bY + c$)

$$x' = \frac{h_1x + h_2y + h_3}{h_7x + h_8y + h_9}$$

$$y' = \frac{h_4x + h_5y + h_6}{h_7x + h_8y + h_9}$$

Homography
- Case 2: **No translation, only rotation** ($T_x = T_y = T_z = 0$)
 - Is also a homography!

$$x' = F \frac{r_1x + r_2y + r_3 F}{r_7x + r_8y + r_9 F}$$

$$y' = F \frac{r_4x + r_5y + r_6 F}{r_7x + r_8y + r_9 F}$$
- Case 3: **When rotation and translation is in the image plane and the scene has a constant depth ($Z = \text{const}$)**

$$[R] = \begin{bmatrix} \cos \theta_z & -\sin \theta_z & 0 \\ \sin \theta_z & \cos \theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}, T = \begin{bmatrix} T_x \\ T_y \\ 0 \end{bmatrix}$$
 - Can reduce to **affine mapping**:

$$x' = a_1x + a_2y + a_0$$

$$y' = b_1x + b_2y + b_0$$

Recap

- Under what conditions we have homography mapping?
 - When camera center remains the same, but the view orientation rotates (approximately true if the center movement is very small compared to the distance of the camera to the imaged scene)



- When the imaged scene is approximately flat (small depth change in the scene relative to the distance to the camera)

Approximation of Homography by Affine and Bilinear Model (Review)

- Affine (6 parameters):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_0 + a_1u + a_2v \\ b_0 + b_1u + b_2v \end{bmatrix}$$

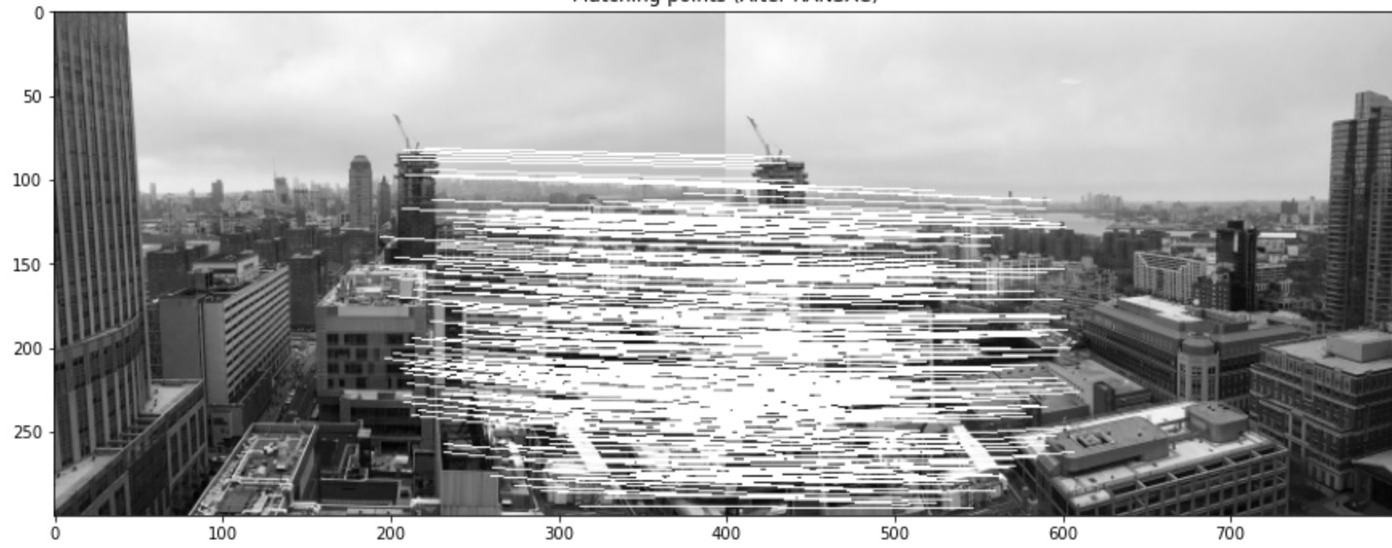
$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ w \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

- Affine model sufficiently capture mapping due to in-plane camera motion (scaling, roll and translation in x,y only)
- Also known as affine homography

- Bilinear (8 parameters):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_0 + a_1u + a_2v + a_3uv \\ b_0 + b_1u + b_2v + b_3uv \end{bmatrix}$$

Matching points (After RANSAC)



Stitched image



Global Motion Estimation

- Feature-based vs. intensity-based
 - Feature-based: First determine some corresponding feature points (using feature detector and descriptor) in both images, then try to fit the correspondences into a chosen mapping model (covered previously)
 - Least squares
 - Robust fitting: RANSAC
 - Intensity-based: Directly determine the motion field (or motion parameters) so that the intensities of corresponding pixels match (focus in this lecture)
- Direct vs. indirect estimation under intensity-based approach
 - Direct: Directly finding the motion parameters
 - Indirect: First find dense motion field, then fit the motion field to a chosen motion model

Direct Estimation based on intensity

- Parameterize the DFD error in terms of the motion parameters, and estimate these parameters by minimizing the DFD error over all pixels in the image:

$$E_{\text{DFD}} = \sum_{n \in \mathcal{N}} w_n |\psi_2(\mathbf{x}_n + \mathbf{d}(\mathbf{x}_n; \mathbf{a})) - \psi_1(\mathbf{x}_n)|^p$$

Weighting w_n coefficients depend on the importance of pixel \mathbf{x}_n .

Ex: Global translation: \mathbf{a} = global translation vector

Ex: Affine motion: $\begin{bmatrix} d_x(\mathbf{x}_n; \mathbf{a}) \\ d_y(\mathbf{x}_n; \mathbf{a}) \end{bmatrix} = \begin{bmatrix} a_0 + a_1 x_n + a_2 y_n \\ b_0 + b_1 x_n + b_2 y_n \end{bmatrix}$, $\mathbf{a} = [a_0, a_1, a_2, b_0, b_1, b_2]^T$

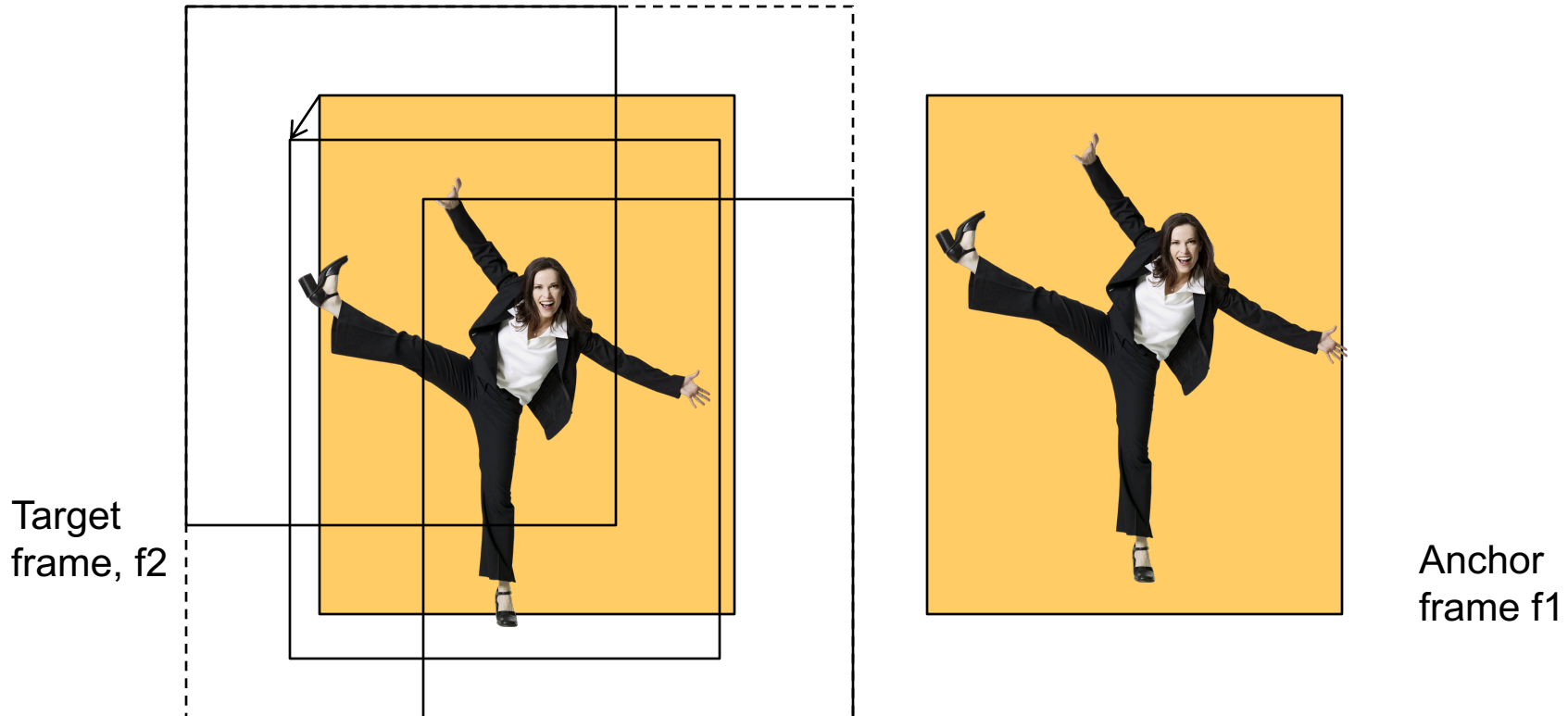
Exhaustive search or gradient descent method can be used to find \mathbf{a} that minimizes E_{DFD}

Global Translation Estimation

- Simple global motion: Every pixel is moved by the same amount due to camera in-plane shift (global translation)
- How to find the global translation?
- Exhaustive search: Applying EBMA to the entire frame
 - Find the shift between the anchor frame and the target frame so that the matching error is minimal
 - Integer or fractional pel search
 - Matching error should be calculated over overlapping pixels only, and the error should be normalized by the number of pixels in the overlapping area (average error / pixel)
- When the global translation is known to be small, the shift can be determined by solving an equation derived from optical flow constraint
 - LK method using a neighborhood containing the entire frame

Estimating Global Shift through Exhaustive Search

- Think of the whole anchor frame f_1 as a block
- Find the match of f_1 in the target frame f_2 by evaluating matching error with all possible shifts
- =EBMA using the whole anchor frame as the template!



Sample MATLAB Code

```
function [dx,dy]=GlobalMVEstimation1(f2, f1,Rx, Ry)
%finding global MV of f2 with respect to f1
%Rx and Ry are search range (maximum possible absolute shift)

[H,W]=size(f);[Hr,Wr]=size(fr);Maxerror=255; dx=0;dy=0;
for (k=-Ry:Ry, l=-Rx:Rx) %try all possible shifts
    error=0;count=0;
    for (m1=1:Hr,n1=1:Wr)
        m2=m1+k;n2=n1+l;
        if ((m2>0) & (m2 <=H) &( n2>0) &( n2<=W))
            count+=1;
            error += abs(f1(m1,n1)-f2(m2,n2));
        end
    end
    error=error/count;
    if (error<maxerror)
        dy=k,dx=l,maxerror=error;
    end
end
end
```

%This script is not very efficient. How do you improve it by not looping through m1,n1 and checking “if ...”

Alternate Faster Implementation

```
function [dx,dy]=GlobalMVEstimation2(f2, f1,Rx, Ry)
%finding global MV of f2 with respect to f1
%Rx and Ry are search range (maximum possible absolute shift)

[H,W]=size(f2);[Hr,Wr]=size(f1);Maxerror=255; dx=0;dy=0;
for (k=-Ry:Ry, l=-Rx:Rx) %try all possible shifts
    error=0;count=0;
    mb=..., me=..., nb=..., ne=...
    count=(me-mb+1)*(nb-ne+1);
    %mb,me,nb,ne should be determined so that mb>=1 & mb+k>=1, similarly me<=Hr &
    me+k<=H. Similarly for nb,ne
    error=sum(sum(abs(f1(mb:me,nb:ne)-f2(mb+k:me+k,nb+l:ne+l))))/count;
    if (error<maxerror)
        dy=k,dx=l,maxerror=error;
    end
end
end

%Hint: To satisfy mb>=1 & mb+k>=1, we can set mb=max(1,1-k)
```

Global Affine Transformation

- Affine mapping is a good approximation of the global motion due to camera motion, especially for far-away view
- Global Affine Transformation (6 parameters)

$$\psi_1(x, y) = \psi_2(x + d_x(x, y), y + d_y(x, y)) = \psi_2(x + \mathbf{A}(x, y)\mathbf{a}, y + \mathbf{A}(x, y)\mathbf{b})$$

$$\begin{bmatrix} d_x(x, y) \\ d_y(x, y) \end{bmatrix} = \begin{bmatrix} a_0 + a_1x + a_2y \\ b_0 + b_1x + b_2y \end{bmatrix} = \begin{bmatrix} \mathbf{A}(x, y) & 0 \\ 0 & \mathbf{A}(x, y) \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$$

$$\mathbf{A}(x, y) = \begin{bmatrix} 1 & x & y \end{bmatrix}, \mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

- Special cases:
 - Translation only: $a_0 = x\text{-direction shift}, a_1 = 0, a_2 = 0$
 $b_0 = y\text{-direction shift}, a_1 = 0, a_2 = 0$

Direct Estimation of Affine Motion: Minimizing DFD Error

- Parameterize the DFD error in terms of the motion parameters, and estimate these parameters by minimizing the DFD error

$$E_{\text{DFD}} = \sum_{n \in \mathcal{N}} w_n |\psi_2(\mathbf{x}_n + \mathbf{d}(\mathbf{x}_n; \mathbf{a})) - \psi_1(\mathbf{x}_n)|^p$$

Weighting w_n coefficients depend on the importance of pixel \mathbf{x}_n .

Ex: Affine motion:

$$\begin{bmatrix} d_x(\mathbf{x}_n; \mathbf{a}) \\ d_y(\mathbf{x}_n; \mathbf{a}) \end{bmatrix} = \begin{bmatrix} a_0 + a_1 x_n + a_2 y_n \\ b_0 + b_1 x_n + b_2 y_n \end{bmatrix} = \mathbf{A}(\mathbf{x}_n) \mathbf{a}$$
$$\mathbf{A}(\mathbf{x}_n) = \begin{bmatrix} 1 & x_n & y_n & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_n & y_n \end{bmatrix}$$
$$\mathbf{a} = [a_0, a_1, a_2, b_0, b_1, b_2]^T$$

- Exhaustive search of all 6 parameters is computationally prohibitive!
- Using gradient descent method

Direct Estimation of Affine Motion Using Gradient Descent Method

$$E_{\text{DFD}}(\mathbf{a}, \mathbf{b}) = \frac{1}{2} \sum_{\mathbf{x} \in B(\mathbf{x}_n)} |\psi_2(\mathbf{x} + \mathbf{A}(\mathbf{x}, \mathbf{y})\mathbf{a}, \mathbf{y} + \mathbf{A}(\mathbf{x}, \mathbf{y})\mathbf{b}) - \psi_1(\mathbf{x}, \mathbf{y})|^2 \rightarrow \min$$

$$\mathbf{A}(x, y) = \begin{bmatrix} 1 & x & y \end{bmatrix}, \mathbf{a}^T = \begin{bmatrix} a_0 & a_1 & a_2 \end{bmatrix}, \mathbf{b}^T = \begin{bmatrix} b_0 & b_1 & b_2 \end{bmatrix}$$

B refers to whole frame. Should make frame center has coordinates $x=0, y=0$

$$\frac{\partial E}{\partial \mathbf{a}} = \begin{bmatrix} \frac{\partial E}{\partial a_0} \\ \frac{\partial E}{\partial a_1} \\ \frac{\partial E}{\partial a_2} \end{bmatrix} = \sum_{(x,y) \in B} e(x,y) \frac{\partial \psi_2}{\partial \mathbf{x}} \Big|_{(\mathbf{x} + \mathbf{A}(x,y)\mathbf{a}, \mathbf{y} + \mathbf{A}(x,y)\mathbf{b})} \quad \mathbf{A}(x,y)^T = \begin{bmatrix} \sum_{(x,y) \in B} e(x,y) G_x(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y) \\ \sum_{(x,y) \in B} e(x,y) G_x(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y)x \\ \sum_{(x,y) \in B} e(x,y) G_x(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y)y \end{bmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}} = \begin{bmatrix} \frac{\partial E}{\partial b_0} \\ \frac{\partial E}{\partial b_1} \\ \frac{\partial E}{\partial b_2} \end{bmatrix} = \sum_{(x,y) \in B} e(x,y) \frac{\partial \psi_2}{\partial \mathbf{y}} \Big|_{(\mathbf{x} + \mathbf{A}(x,y)\mathbf{a}, \mathbf{y} + \mathbf{A}(x,y)\mathbf{b})} \quad \mathbf{A}(x,y)^T = \begin{bmatrix} \sum_{(x,y) \in B} e(x,y) G_y(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y) \\ \sum_{(x,y) \in B} e(x,y) G_y(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y)x \\ \sum_{(x,y) \in B} e(x,y) G_y(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y)y \end{bmatrix}$$

$e(x, y) = \psi_2(\mathbf{x} + \mathbf{A}(x, \mathbf{y})\mathbf{a}, \mathbf{y} + \mathbf{A}(x, \mathbf{y})\mathbf{b}) - \psi_1(\mathbf{x}, \mathbf{y})$: Current prediction error image;

$G_x(x, y) = \frac{\partial \psi_2}{\partial x}(x, y)$: Gradient image in x-direction; $G_y(x, y) = \frac{\partial \psi_2}{\partial y}(x, y)$: Gradient image in y-direction

Implementation Details: Gradient Vector Calculation

First order gradient descent (starting from some initial condition):

At $(l+1)$ -th iteration:

$$\mathbf{a}^{(l+1)} = \mathbf{a}^{(l)} - \alpha \left. \frac{\partial E}{\partial \mathbf{a}} \right|_{\mathbf{a}^{(l)}, \mathbf{b}^{(l)}}, \quad \mathbf{b}^{(l+1)} = \mathbf{b}^{(l)} - \alpha \left. \frac{\partial E}{\partial \mathbf{b}} \right|_{\mathbf{a}^{(l)}, \mathbf{b}^{(l)}}$$

First calculate the gradient images of $\psi_2(x, y)$, to generate original gradient image $G_x(x, y), G_y(x, y)$

Find an initial solution, $\mathbf{a}^{(0)}, \mathbf{b}^{(0)}$.

At end of (l) -th iteration, you have $[a_0, a_1, a_2]^T = \mathbf{a}^{(l)}, [b_0, b_1, b_2]^T = \mathbf{b}^{(l)}$.

Determine the predicted image:

$$\psi_p^{(l)}(x, y) = \psi_2(x + a_0 + a_1x + a_2y, y + b_0 + b_1x + b_2y) = \text{warp}(\psi_2(x, y), a_0, a_1, a_2, b_0, b_1, b_2)$$

Determine prediction error image: $e^{(l)}(x, y) = \psi_p^{(l)}(x, y) - \psi_1(x, y)$

Determine the shifted gradient images:

$$G_x^{(l)}(x, y) = \text{warp}(G_x(x, y), a_0, a_1, a_2, b_0, b_1, b_2); \quad G_y^{(l)}(x, y) = \text{warp}(G_y(x, y), a_0, a_1, a_2, b_0, b_1, b_2)$$

Compute the gradient vectors, using

$$\frac{\partial E}{\partial \mathbf{a}} = \begin{bmatrix} \sum_{(x,y) \in B} e^{(l)}(x, y) G_x^{(l)}(x, y) \\ \sum_{(x,y) \in B} e^{(l)}(x, y) G_x^{(l)}(x, y) x \\ \sum_{(x,y) \in B} e^{(l)}(x, y) G_x^{(l)}(x, y) y \end{bmatrix}, \quad \frac{\partial E}{\partial \mathbf{b}} = \begin{bmatrix} \sum_{(x,y) \in B} e^{(l)}(x, y) G_y^{(l)}(x, y) \\ \sum_{(x,y) \in B} e^{(l)}(x, y) G_y^{(l)}(x, y) x \\ \sum_{(x,y) \in B} e^{(l)}(x, y) G_y^{(l)}(x, y) y \end{bmatrix}$$

Implementation Details: Gradient Image Calculation

- Simple implementation (using centered difference)

$$G_x(x, y) = (\psi_x(x+1, y) - \psi_x(x-1, y)) / 2$$

$$G_y(x, y) = (\psi_y(x, y+1) - \psi_y(x, y-1)) / 2$$

- Convolved with filters that approximate the gradient operation

$$G_x(x, y) = \psi(x, y) * H_x(x, y), \quad G_y(x, y) = \psi(x, y) * H_y(x, y),$$

– Sobel operator

$$H_x = \frac{1}{4} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_y = \frac{1}{4} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Derivative of Gaussian filters (see previous lecture notes)

How to determine the initial solution?

- If the anticipated rotation is small, may assume only translation is present. Estimate the translation parameters using the global translation estimation algorithm.

$$a_0 = \text{translation in } x, a_1 = 0, a_2 = 0;$$

$$b_0 = \text{translation in } y, b_1 = 0, b_2 = 0.$$

- If the anticipated translation is also small, can assume

$$a_0 = 0, a_1 = 0, a_2 = 0;$$

$$b_0 = 0, b_1 = 0, b_2 = 0.$$

When to stop the iteration?

- When the energy functional being minimized stop decreases
- Energy function at $(l+1)$ iteration

$$E^{(l+1)} = \sum (e^{(l+1)}(x, y))^2$$

- At end of $(l+1)$ iteration, check

$$\frac{(E^{(l+1)} - E^{(l)})}{E^{(l)}} < T?$$

Indirect Estimation of Global Motion

- First find the dense motion field using pixel-based or block-based approach (e.g. EBMA), or find motion vectors at selected feature points, resulting in a sequence of data pairs

$$(x_n, y_n), (d_{x,n}, d_{y,n}), n = 1, 2, \dots, N$$

- Then finding the motion model parameters to satisfy the equations:

$$d_x(x_n, y_n; \mathbf{a}) = d_{x,n}, d_y(x_n, y_n; \mathbf{b}) = d_{y,n}, n = 1, 2, \dots, N$$

- The parameters for dx and dy can be solved separately by fitting the models for dx and dy separately if dx and dy do not share parameters (e.g. affine or bilinear motion).
- Can use the same approaches described previously from determining the "mapping" parameters from feature correspondences. Here each pixel or block and its corresponding pixel or block forms a feature correspondence.

Least Squares Fitting for Affine Model

Fitting error for dx: $E_{fit,x} = \sum w_n (d_x(\mathbf{x}_n; \mathbf{a}) - d_{x,n})^2$.

Affine motion: $d_x(\mathbf{x}_n; \mathbf{a}) = [\mathbf{A}_n] \mathbf{a}$,

$$\begin{bmatrix} \frac{\partial E}{\partial a_0} \\ \frac{\partial E}{\partial a_1} \\ \frac{\partial E}{\partial a_2} \end{bmatrix} = \frac{\partial E_{fit}}{\partial \mathbf{a}} = \sum w_n [\mathbf{A}_n]^T ([\mathbf{A}_n] \mathbf{a} - d_{x,n}) = 0 \rightarrow \mathbf{Q} \mathbf{a} = \mathbf{r}_x \rightarrow \mathbf{a} = \mathbf{Q}^{-1} \mathbf{r}_x$$

$$[\mathbf{A}_n]^T [\mathbf{A}_n] = \begin{bmatrix} 1 & x_n & y_n \\ x_n & x_n^2 & x_n y_n \\ y_n & x_n y_n & y_n^2 \end{bmatrix}, [\mathbf{A}_n]^T d_{x,n} = \begin{bmatrix} d_{x,n} \\ x_n d_{x,n} \\ y_n d_{x,n} \end{bmatrix}$$

$$\mathbf{Q} = \sum w_n [\mathbf{A}_n]^T [\mathbf{A}_n] = \begin{bmatrix} \sum w_n & \sum w_n x_n & \sum w_n y_n \\ \sum w_n x_n & \sum w_n x_n^2 & \sum w_n x_n y_n \\ \sum w_n y_n & \sum w_n x_n y_n & \sum w_n y_n^2 \end{bmatrix}, \mathbf{r}_x = \sum w_n [\mathbf{A}_n]^T d_{x,n} = \begin{bmatrix} \sum w_n d_{x,n} \\ \sum w_n x_n d_{x,n} \\ \sum w_n y_n d_{x,n} \end{bmatrix}$$

Similarly,

$$\mathbf{b} = \mathbf{Q}^{-1} \mathbf{r}_y, \mathbf{r}_y = \sum w_n [\mathbf{A}_n]^T d_{y,n} = \begin{bmatrix} \sum w_n d_{y,n} \\ \sum w_n x_n d_{y,n} \\ \sum w_n y_n d_{y,n} \end{bmatrix}$$

- Weighting w_n coefficients depend on the accuracy of estimated motion at \mathbf{x}_n .
- This is similar to the “feature-based” approach, but now we use the dense motion vectors found for all pixels or small blocks.

Motion vs. Mapping Function

- Previously we tried to use the motion vectors to find the parameters of the global motion function.
- We could also using the pixel correspondences to directly find the global mapping function.

$$(x_n, y_n), (d_{x,n}, d_{y,n}), n = 1, 2, \dots, N \rightarrow$$

$$(x_n, y_n), (u_n, v_n), n = 1, 2, \dots, N, \text{ with } u_n = x_n + d_{x,n}, v_n = y_n + d_{y,n}$$

- For Affine mapping, the motion function is also affine:

$$x = a_0 + a_1 u + a_2 v \rightarrow$$

$$d_x(u, v) = x - u = a_0 + (a_1 - 1)u + a_2 v$$

- For homography, this is not true:

$$x = \frac{a_0 + a_1 u + a_2 v}{c_0 + c_1 u + c_2 v} \rightarrow$$

$x - u$ cannot be characterized by a homography function

- But least squares method can be used to directly find the parameters for the mapping function (see lecture note for feature matching for determining the homograph parameters)

Problem with Least Squares Fitting

- Estimated motions at some pixels may be grossly wrong (outliers)
- Outliers can significantly impact the global motion estimation results when using square error, leading to errors in object detection as well
 - Outliers can be due to moving objects, or tree leave movements, etc.
- Alternatives:
 - Instead of minimizing the square error, minimize the L0 norm (very hard)
 - Convex relaxation: Minimize L1 norm instead.
 - Computationally more demanding than minimizing L2 error, but solvable.
 - RANSAC method if based on feature correspondence (May not be computationally feasible based on dense motion vectors)
 - Can also use iterative weighted least square: the weights are smaller for pixels with larger fitting errors in the previous iteration

Pop Quiz

- What is the difference between direct and indirect methods?
- What are the two ways to set up your intensity-based objective function?
- What are the pros and cons using the optical flow equation to set up your optimization criterion?
- How would you estimate the global translation based on the optical flow equation?
- How would you estimate the homography parameters between two frames using the indirect method?

Pop Quiz (w/ Answers)

- What is the difference between direct and indirect methods?
 - Direct: directly determine the motion parameters by minimizing the sum of intensity difference between corresponding pixels
 - Indirect: first determine the dense motion field, then fit the motion field into a global function
- What are the two ways to set up your intensity-based objective function for the direct method? What are the pros and cons of each?
 - Directly minimizing the sum of intensity difference, require exhaustive search or gradient descent method.
 - Minimizing the error in satisfying the optical flow equation, which can be solved with closed form solution. But it is valid only if the motion vector at every pixel is small under the global motion. Typically not true unless the global translation or rotation is very small.
- How would you estimate the global translation based on the optical flow equation?
 - Set up a set of equation for all pixels assuming that same motion vector -> LK method using a neighborhood = full frame
- How would you estimate the homography parameters between two frames using the indirect method?
 - First determine the dense motion field
 - Set up two equations for every pixel and its corresponding pixel
 - Solve using least squares method

This Lecture

- Global motion estimation
 - Direct estimation: global shift estimation
 - Indirect estimation: estimating global motion parameters from dense motion field
- • Region-based motion estimation
- Video stabilization
- Object tracking
 - Single object
 - Multiple object
 - Deep learning approaches
- Appendix (not required for the class)
 - Gradient descent method for determining global motion
 - Background modeling and moving object detection
 - Video shot segmentation

What if not all pixels experience the same camera motion?

- The background experiences the camera motion
- Moving objects have different motion than the background
- Because moving objects are typically small in a frame, the pixels corresponding to these objects can be considered as “outliers”
- First determine camera motion by minimizing the error over all pixels (or feature points), then detect pixels with large error (in intensity or position)
- Moving objects are the outliers!
- Once moving objects are detected, can also estimate the motion of the object
 - > Region based motion estimation

Region-Based Motion Estimation (Optional)

- Assumption: the scene consists of multiple objects, with the region corresponding to each object (or sub-object) having a coherent motion
 - Background can be one of the region
 - Physically more correct than block-based, mesh-based, global motion model
- Method:
 - Region First: Segment the frame into multiple regions based on texture/edges, then estimate motion in each region using the global motion estimation method
 - Motion First: Estimate a dense motion field, then segment the motion field so that motion in each region can be accurately modeled by a single set of parameters
 - This can be done by clustering: partition all pixels into different groups based on their motion similarity, using e.g. K-means algorithm
 - Joint region-segmentation and motion estimation: iterate the two processes
 - Layered motion estimation
 - Fit all pixels into a global (background or layer 1) motion, determine outliers
 - Fit all outliers into another global motion (the motion of the most dominant object or layer 2), determine remaining outliers
 - ...

Layered Motion Estimation (Optional)

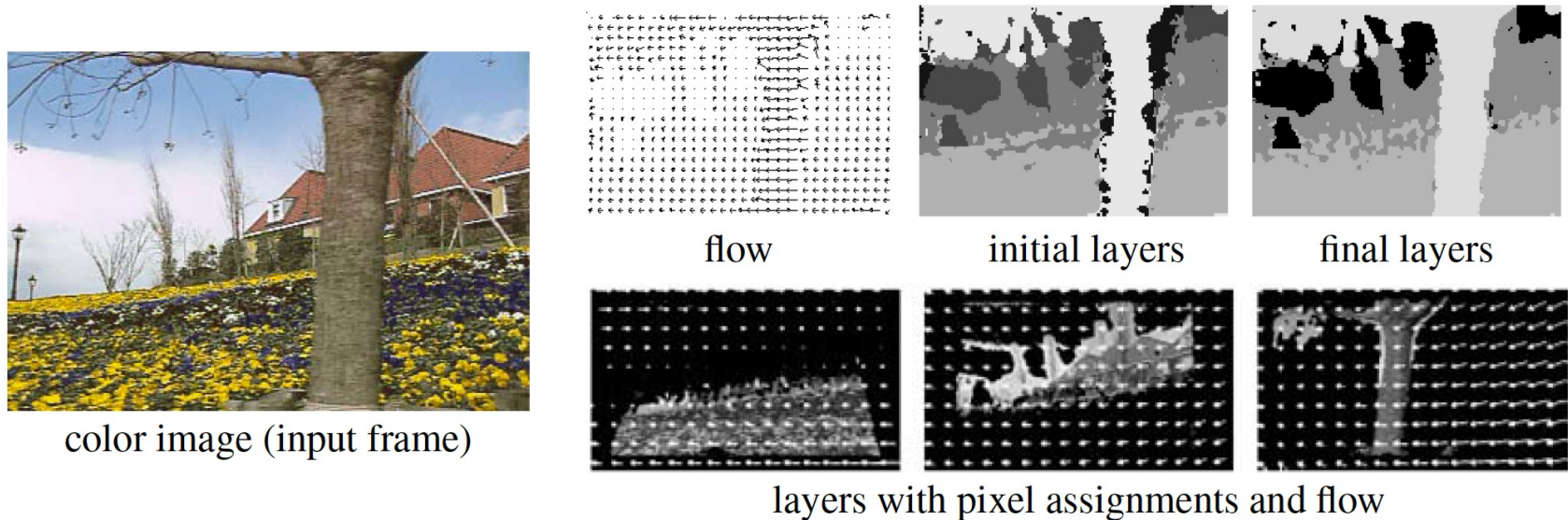


Figure 8.15 Layered motion estimation results (Wang and Adelson 1994) © 1994 IEEE.

Wang, J. Y. A. and Adelson, E. H. (1994). Representing moving images with layers. IEEE Transactions on Image Processing , 3(5):625–638.

See [Szeliski2010] Sec. 8.5.

This Lecture

- Camera motion estimation
 - Direct estimation: global shift estimation
 - Indirect estimation: estimating global motion parameters from dense motion field
- Region-based motion estimation
- • Video stabilization
- Object tracking
 - Single object
 - Multiple object
 - Deep learning approaches
- Appendix (not required for the class)
 - Gradient descent method for determining global motion
 - Background modeling and moving object detection
 - Video shot segmentation

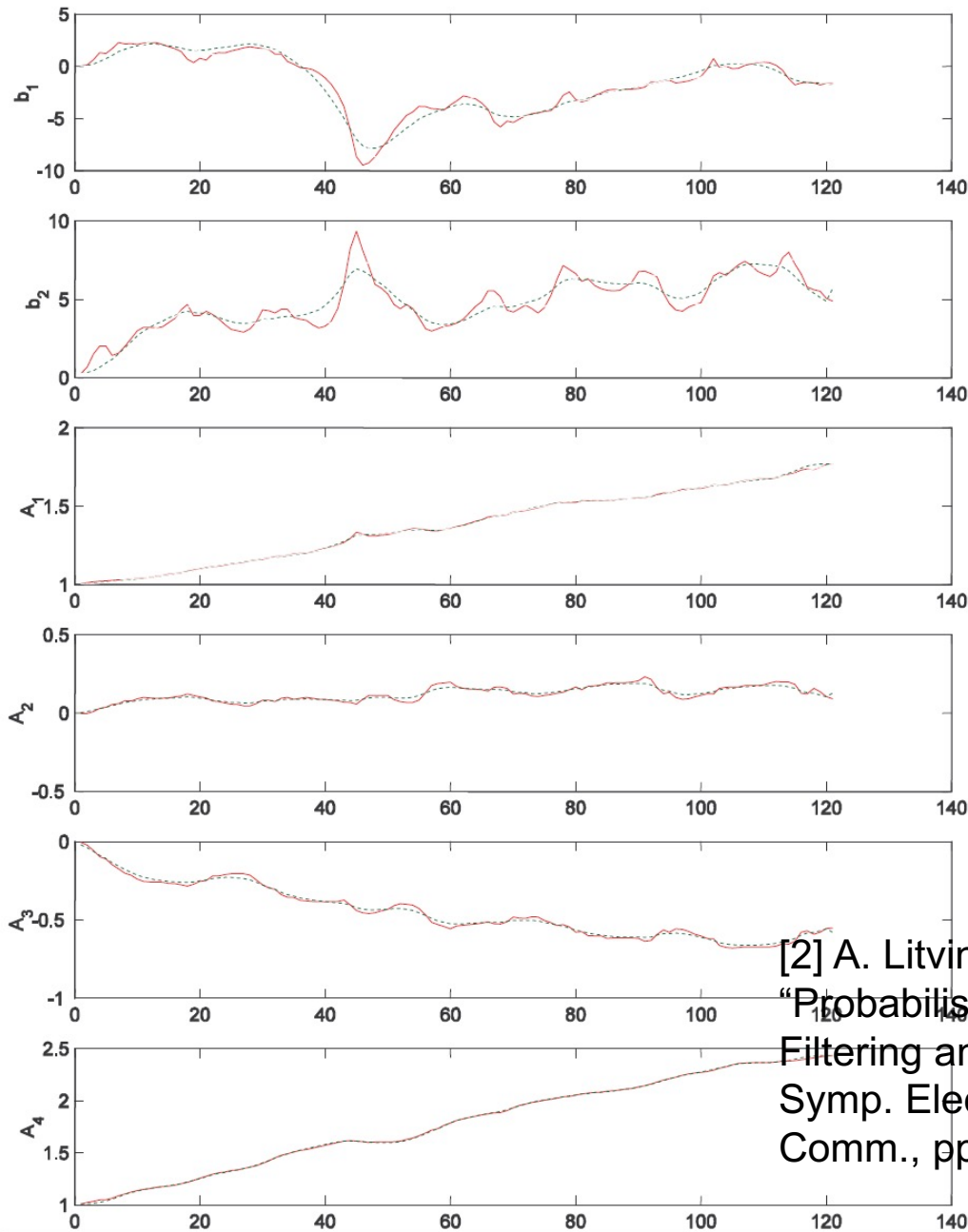
Video Stabilization

- A video may be unstable due to unwanted camera motion
- Especially prevalent in home video captured by hand-held cameras with a “shaky hand”
- Also prevalent in aerial surveillance video
- Goal: remove the motion due to unwanted camera motion, so that the video plays smoothly

General Approach

- Estimate global motion between every two adjacent frames
 - Assuming a global translation is often sufficient
 - More generally using affine to account for in-plane tilt or homography to account for out of plane rotation
 - Can use either feature-based or intensity-based approaches
- Smooth motion parameters in time (to remove shaking, but keep the smooth camera motion)
- Warping each frame so that it undergoes the smoothed motion between frames
 - Remove undesired global motion due to hand shaking
- Filling missing pixels (on the border) in each frame (image completion)

Global Motion Smoothing



- Separating observed motion (red) parameters into intentional (smooth, dotted) and unwanted motion
- Simple low pass filtering
 - Kalman filtering approach: [Ref 2]

[2] A. Litvin, J. Konrad, and W. Karl, "Probabilistic Video Stabilization Using Kalman Filtering and Mosaicking," Proc. IS&T/SPIE Symp. Electronic Imaging, Image, and Video Comm., pp. 663-674, 2003.

Figure 5: Red straight lines – Inter-frame motion parameters obtained for sequence A; Green dashed lines - intentional cumulative transform parameters estimated using smoothing Kalman filtering

Motion Correction Needed

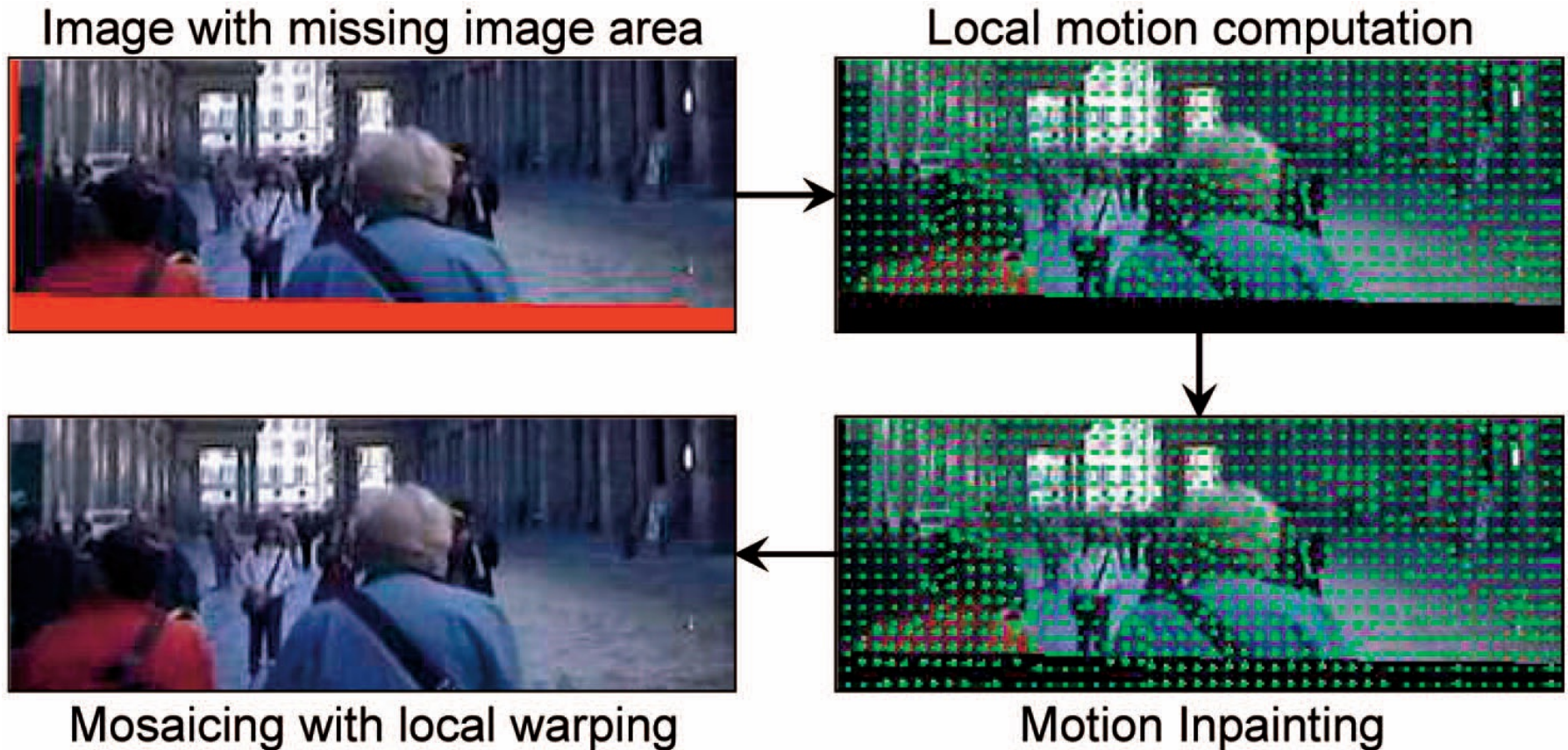
- Original global motion parameters between successive frames I_{n-1} and I_n : a_n
 - $x_n = x_{n-1} + h(x_{n-1}, a_n)$
- Smoothed parameters: b_n
- Correction needed for frame n
 - Observed pixel location $x_n = x_{n-1} + h(x_{n-1}, a_n)$
 - Desired pixel location $x'_n = x_{n-1} + h(x_{n-1}, b_n)$
 - Correction needed: $x'_n - x_n = h(x_{n-1}, b_n) - h(x_{n-1}, a_n) = d(x_{n-1}, b_n, a_n)$
 - Should warp x_n in I_n to x'_n in stabilized frame I'_n
 - Using inverse mapping $I'(x'_n) = I(x_n = x'_n - d(x_{n-1}, b_n, a_n))$
- Special case: Consider only a global translation
 - $d(x_{n-1}, b_n, a_n) = b_n - a_n$: difference in smoothed translation and original translation

Sample Results: Considering Translation only



From: Y. Matsushita, E. Ofek, W. Ge, X. Tang, H.-Y. Shum, "Full-Frame Video Stabilization with Motion Inpainting," IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 28, NO. 7, JULY 2006

Video Completion by Motion Inpainting [Ref 1]



[1] Y. Matsushita, E. Ofek, W. Ge, X. Tang, H.-Y. Shum, "Full-Frame Video Stabilization with Motion Inpainting," IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 28, NO. 7, JULY 2006

Pop Quiz

- What are the major steps in stabilization?

Pop Quiz (w/ Answer)

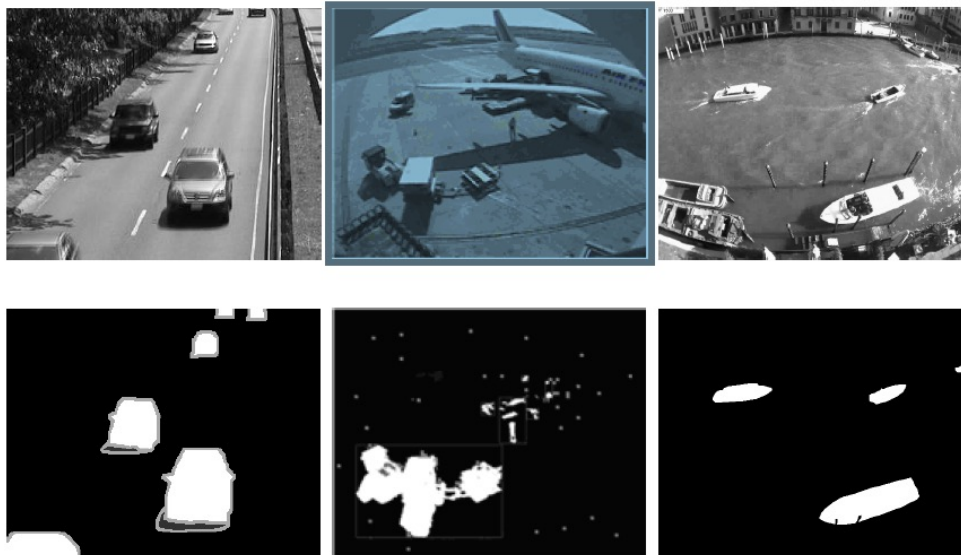
- What are the major steps in stabilization?
 - Global motion (mapping) estimation -> motion parameter smoothing -> warping based on the difference of the smoothed motion and the measured motion -> image completion

This Lecture

- Global motion estimation
 - Direct estimation: global shift estimation
 - Indirect estimation: estimating global motion parameters from dense motion field
- Region-based motion estimation
- Video stabilization
- • Moving object detection and background modeling
- Object tracking
 - Single object
 - Multiple object
 - Deep learning approaches
- Appendix (not required)
 - Gradient descent method for determining global motion
 - Background modeling and moving object detection: advanced methods
 - Video shot segmentation

Object Detection and Background Modeling

- Main applications:
 - Visual surveillance (Road, Airport, Parking lot, In home security,...)
 - Activity pattern discovery: e.g. # of people, # of cars
- In most applications, we want to detect the moving objects
- In some applications, we want to form a complete background from video frames.
 - Once we have the background, moving objects can be detected by taking the difference from the background image and thresholding



Moving Object Detection

- Simple idea
 - Assuming background is stationary, color changes only at moving regions
 - Take difference between two frames, detect pixels with large difference.
 - Post processing is needed to form smooth, connected foreground regions

Moving object detection by examining frame difference

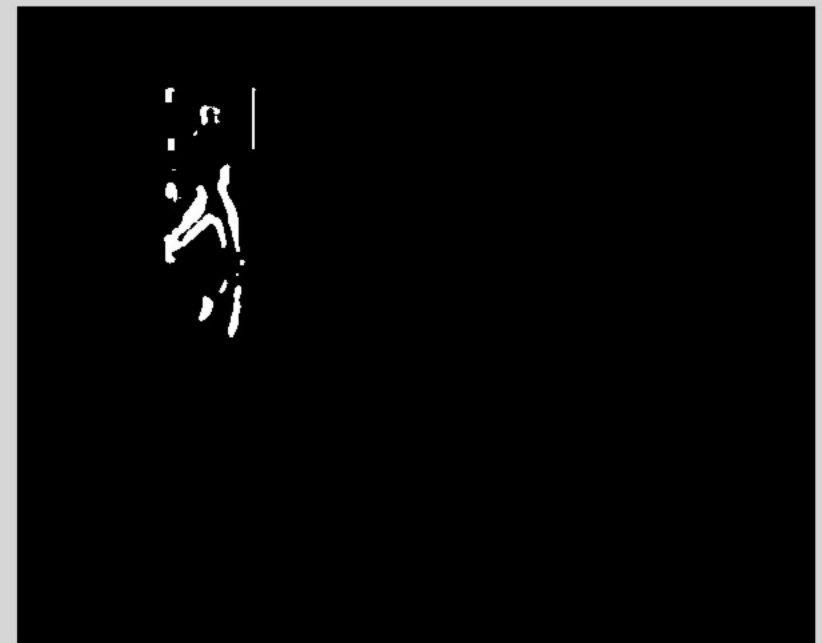
- Frame at t: $f(x,y,t)$
- Frame Difference at t: $e(x,y,t)=|f(x,y,t)-f(x,y,t-1)|$
- Thresholding the difference to highlight pixels with large change
- Postprocessing
 - Remove isolated foreground pixels due to false detection
 - Find a connected blob covering the foreground pixels (blob detection, connectivity analysis, and other tools in openCV/Matlab)
 - Alternative: Put a bounding box covering all detected foreground pixels after removing isolated pixels



```
>> img1=imread('frame31.jpg');  
>> img2=imread('frame32.jpg');  
>> img1=rgb2gray(img1);  
>> img2=rgb2gray(img2);  
>> img1=int16(img1);  
>> img2=int16(img2);  
>> diff=abs(img1-img2);  
  
>> figure(1),imshow(img1,[])  
>> figure(2),imshow(img2,[])  
>> figure(3),imshow(diff,[])
```



```
>> figure(3),imshow(diff,[])  
>> max(max(diff))  
>> diffT=(diff>20);  
>> figure(4),imshow(diffT,[])  
>> diffTM=medfilt2(diffT,[5 5]);  
>> figure(5),imshow(diffTM,[])
```



Problem with frame difference

- The background may not be stationary
 - Tree leaf motion
 - Lighting change
 - Camera motion
- More advanced methods are needed to compensate for such changes
 - Background modeling to account for such changes
 - Only pixels that do not fit with the background model are considered foreground object

Challenge due to Illumination Changes



a) Light-on

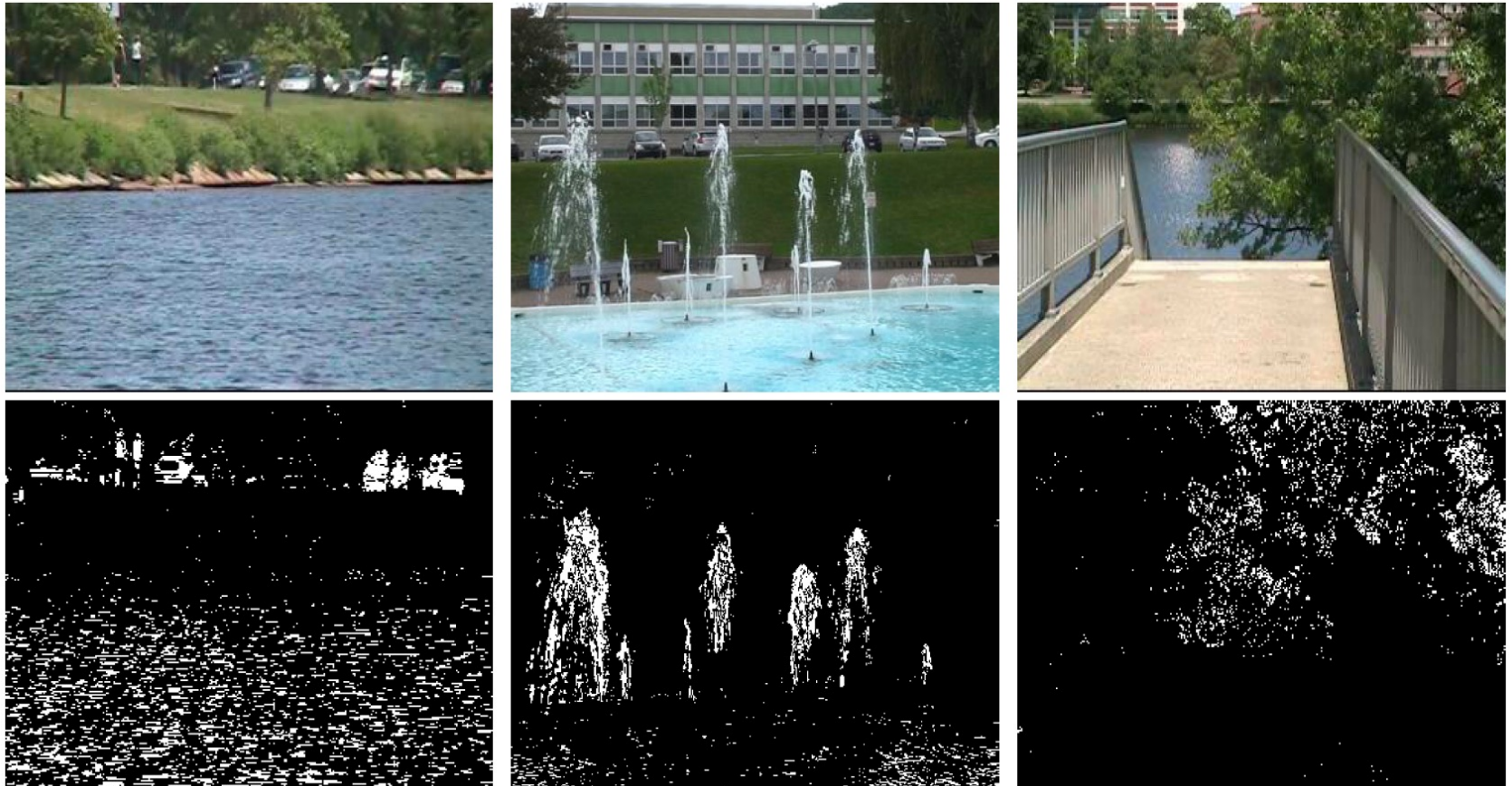
b) Light-off

c) Foreground mask

Foreground detected using the MoG algorithm

From: T. Bouwmans, F. El-Baf, and B. Vachon. Background modeling using mixture of Gaussians for foreground detection: A survey. Recent Patents on Computer Science , 1(3):219–237, November 2008.

Challenge due to Dynamic Background



Foreground detected using the MoG algorithm

From: T. Bouwmans, F. El-Baf, and B. Vachon. Background modeling using mixture of Gaussians for foreground detection: A survey. Recent Patents on Computer Science , 1(3):219–237, November 2008.

Challenge due to Shadows



Foreground detected using the MoG algorithm

From: T. Bouwmans, F. El-Baf, and B. Vachon. Background modeling using mixture of Gaussians for foreground detection: A survey. Recent Patents on Computer Science , 1(3):219–237, November 2008.

Background Modeling (Threshold-Based)

- Simple method
 - Averaging all frames (hoping moving objects will be averaged out over a long period of time)
 - Work well when the camera is stationary and illumination is nearly constant, and you can average many many frames
- Recursive update
 - The background up to the previous frame B_{t-1}
 - Given a new frame F_t , form difference $D(x,y)=F_t(x,y)-B_{t-1}(x,y)$
 - If $|D(x,y)| < T$, assign (x,y) to Background, use $F(x,y)$ to update $B_{t-1}(x,y)$.
 - $B_t(x,y)=(1-a) B_{t-1}(x,y)+ aF_t(x,y)$
 - Need to determine parameters T and a properly through a “validation” process.
- Does not work well in challenging situations (illumination change, dynamic background, shadows)

Advanced Methods (Optional)

- Modeling the colors at each pixel using a Gaussian mixture model (GMM) (aka mixture of Gaussian or MoG)
 - Recursively update the GMM parameter at each pixel
- Decompose the video into a low-rank component (corresponding to stationary background) and a sparse (corresponding to moving objects) component (L+S)
- What if the background is moving due to camera motion?
 - Have to determine the camera motion between two frames first and compensate for the camera motion
- Not required. See Appendix

This Lecture

- Camera motion estimation
 - Direct estimation: global shift estimation
 - Indirect estimation: estimating global motion parameters from dense motion field
- Region-based motion estimation
- Video stabilization
- Moving object detection and background modeling
- • Object tracking
 - Single object
 - Multiple object
 - Deep learning approaches
- Appendix (not required)
 - Gradient descent method for determining global motion
 - Background modeling and moving object detection: advanced methods
 - Video shot segmentation

Single Object Tracking

- Suppose you identified a person or an object in one frame, and you want to find how does it move in the subsequent frames.
- How do you do that?
- Simple approach:
 - If you put a bounding box over the person, then the color pattern within the bounding box (template block) should not change much even if the box is moving over time
 - We can find how does the box move by searching for a same sized box with similar color pattern in successive frames –known as template matching
 - Can be accomplished by using the EBMA method repeatedly on two adjacent frames, where the block is the object bounding box in the current frame (anchor frame), and you find the matching block in the next frame (target frame)

Object Tracking by Template Matching



```
>> figure(1),imshow(img1,[])  
>> figure(2),imshow(img2,[])  
>> x0=112,y0=59,x1=175,y1=202  
>> Rx=24,Ry=10  
>> template=img1(y0:y1,x0:x1);  
>> [xm,ym,matchblock]=EBMA(template,img2,x0,y0,Rx,Ry);  
>> xm, ym
```

Problems with Template Matching

- The shape and appearance of the object may change if the motion is not just a shift
 - Allow “affine” or “homography” mapping between the current block and the tracked block
 - Find the best mapping parameters that minimizes the color differences between corresponding pixels in the current box and the mapped box.
- Different parts of the object may move differently
- Some parts may disappear, new parts may appear (occlusion issues)
- More sophisticated algorithms are needed to solve these challenges (outside the scope of this lecture)
- However, when two frames are very close in time (e.g under high frame rate), the movement of most objects are small and simple block matching can work quite well

KTL Tracker (feature based)

- Basic steps:
 - Detect feature points in a bounding box surrounding the object in the current frame (e.g. Harris corner detector)
 - Find correspond features in the next frame using the Lucas-Kanade (LK) method
 - Repeat for the next frame.
 - Find the affine mapping between corresponding points in non-adjacent frames using robust estimator, drop outliers. Put bounding box on remaining features
 - The number of identified corresponding points will reduce over time. Need to redetect features periodically or when the number of feature points is below a threshold.

- Three papers:

Bruce D. Lucas and Takeo Kanade. [An Iterative Image Registration Technique with an Application to Stereo Vision](#). *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. *Carnegie Mellon University Technical Report CMU-CS-91-132*, April 1991.

Jianbo Shi and Carlo Tomasi. Good Features to Track. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

How to track multiple objects (Optional)

- Can detect multiple objects (each in a detected box) in a starting frame and track each one independently.
 - Challenging if these objects cross each other at some frame
- Can also use feature points
 - Detect all feature points in the current frame
 - Link correspond features in following frames using KLT tracker to form “tracklets”
 - Merge tracklets belonging to the same object
 - Motion consistency, spatial adjacency
 - Using a clustering algorithm to identify clusters, so that within each cluster the feature points are nearby and have similar global motion

Robust Vehicle Tracking for Urban Traffic Videos at Intersections

Li C., Chiang A., Dobler G., Wang Y., Xie K., Ozbay K., Ghandehari M., Zhou J., Wang D.
Center for Urban Science + Progress (CUSP), New York University
Department of Electrical and Computer Engineering, New York University
Paper ID 96

Introduction

- A robust system to **automatically extract vehicle trajectory** data from video data obtained by existing traffic cameras from the New York City Department of Transportation (NYCDOT).
- Automatic trajectory information will be used to develop **realistic surrogate safety measures** to both identify high risk locations and assess implemented safety improvements.

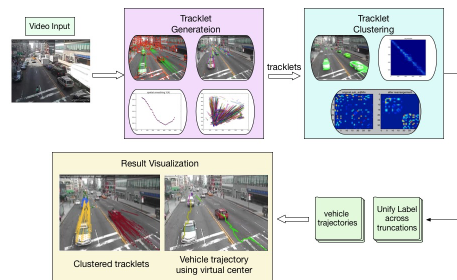


Figure 1: A representation of the steps in our vehicle tracking system.

KLT Tracklet Generation

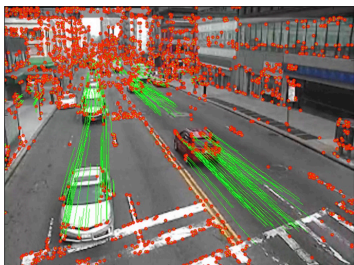


Figure 2: The KLT tracking result shows the effects of perspective as well as the stationary points which are to be filtered out as non-vehicle tracklets in subsequent steps. Feature points are shown in red and tracklets in green.

Motivation and Objectives

We develop a robust, **unsupervised** vehicle tracking system for videos of very congested road intersections in urban environments. Raw tracklets from the standard **Kanade-Lucas-Tomasi (KLT)** tracking algorithm are treated as sample points and grouped to form different vehicle candidates. Each tracklet is described by multiple features including position, velocity, and a foreground score derived from **robust PCA** background subtraction. By considering each tracklet as a node in a graph, we build the adjacency matrix for the graph based on the feature similarity between the tracklets and group these tracklets using spectral embedding and **Dirichlet Process Gaussian Mixture Models**. The proposed system yields excellent performance for traffic videos captured in urban environments and highways.

Challenges from Urban Street Level Videos

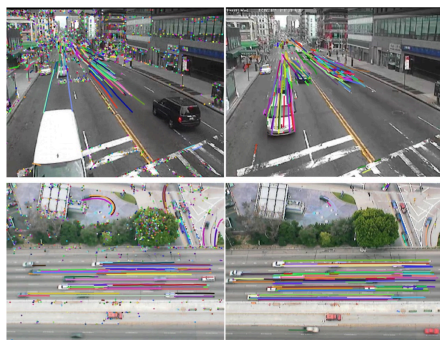


Figure 3: Tracklets are shown before (left) and after (right) filtering and smoothing. Note that the filtering process removes the majority of stationary points on building corners and street paint.

- High degree of **partial occlusions** in dense traffic
- Vehicles have **deformable appearances** due to viewing angles as opposed to the high bird's eye view
- Traffic lights at the intersection lead to vehicle **stop-and-go** conditions
- NYC DOT surveillance videos have low resolution (480 x 640 pixels), frequent illumination changes among frames

Tracklet Clustering

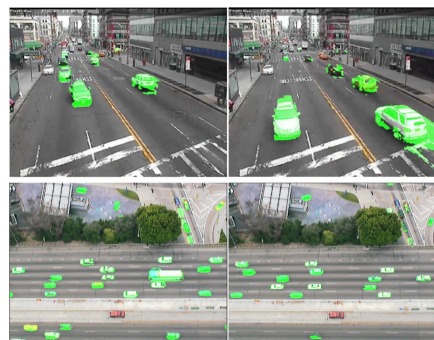


Figure 4: The results of the rPCA foreground/background separation are shown for both street-level NYCDOT (top) and NGSIM (bottom) video.

- The **adjacency matrix** between two tracklets A_{ij} is defined as,

$$\ln A_{ij} = -\tilde{w} \cdot \tilde{f}_{ij}, \quad (1)$$

- \tilde{w} is a weight vector for each feature in $\tilde{f}_{ij} \equiv (x_{ij,max}, v_{x,ij,max}, v_{y,ij,max}, b_{cen,ij,max})$.
- $x_{ij,max} \equiv \max_k |\tilde{x}_i(t_k) - \tilde{x}_j(t_k)|$ is the maximum **positional separation** along the tracklet
- $v_{x,ij,max} \equiv \max_k |v_{x,i}(t_k) - v_{x,j}(t_k)|$ and $v_{y,ij,max} \equiv \max_k |v_{y,i}(t_k) - v_{y,j}(t_k)|$ are the maximum **velocity separations** in two dimensions along the tracklet
- $b_{cen,ij,max} \equiv \max_k |b_{cen,i}(t_k) - b_{cen,j}(t_k)|$ is the maximum **separation of the center of mass of the blob labels**

Result

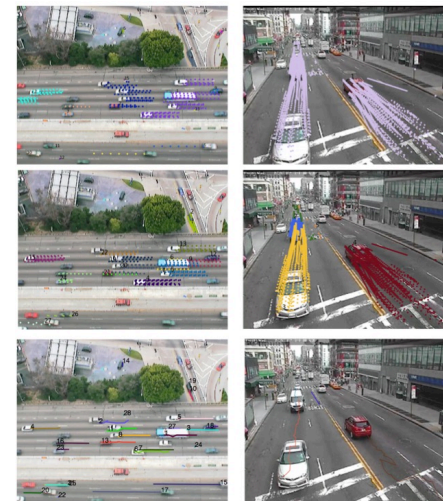


Figure 5: The initial groups from thresholding adjacency matrix (top) and the clustered results for all KLT tracklets (middle) are shown for NGSIM (left) and NYCDOT (right) videos. The final extracted trajectories after spectral clustering and DPGMM are shown in the (bottom) panels.

- Perspective Transformation:** Warping the non-parallel trajectories into parallel
- Hard thresholding** A_{ij} is thresholded according to $x_{ij,max} \leq d$ to form connected components which are never separated by more than a distance d .
- Spectral embedding and Dirichlet Process Gaussian Mixture Model (DPGMM)** to identify the number of clusters automatically and label each tracklet.

Contact: Chenge Li, chengeli@nyu.edu

Challenges with Object Tracking

- The shape and appearance of the object may change if the motion is not just a shift
 - Search for the parameters of possible object motion to minimize the intensity difference after motion compensation in the object region. (See following for global motion estimation)
- Different parts of the object may move differently
- Some parts may disappear, new parts may appear (occlusion issues)
- Many advanced algorithms have been developed to solve these challenges (outside the scope of this lecture)
- Good references:
 - Yilmaz, Alper, Omar Javed, and Mubarak Shah. "Object tracking: A survey." *Acm computing surveys (CSUR)* 38.4 (2006): 13.
<http://7xq232.com1.z0.glb.clouddn.com/talk/2013.12.20-Student.Workshop.pdf>
 - Wu, Yi, Jongwoo Lim, and Ming-Hsuan Yang. "Online object tracking: A benchmark." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013. http://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Wu_Online_Object_Tracking_2013_CVPR_paper.pdf

Deep Learning for Object Tracking (Optional)

- Single object tracking:
 - Held, David, Sebastian Thrun, and Silvio Savarese. "Learning to track at 100 fps with deep regression networks." *European Conference on Computer Vision*. Springer, Cham, 2016. <https://arxiv.org/pdf/1604.01802>, <http://davheld.github.io/GOTURN/GOTURN.html>
 - Bertinetto, Luca, Jack Valmadre, Joao F. Henriques, Andrea Vedaldi, and Philip HS Torr. "Fully-convolutional siamese networks for object tracking." In *European conference on computer vision*, pp. 850-865. Springer, Cham, 2016. <https://arxiv.org/pdf/1606.09549.pdf>
- Multiple object tracking:
 - Ciaparrone, Gioele, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. "Deep learning in video multi-object tracking: A survey." *Neurocomputing* 381 (2020): 61-88
 - Work at NYU Video Lab (TrackNet)

GOTURN (Single Object Tracking)

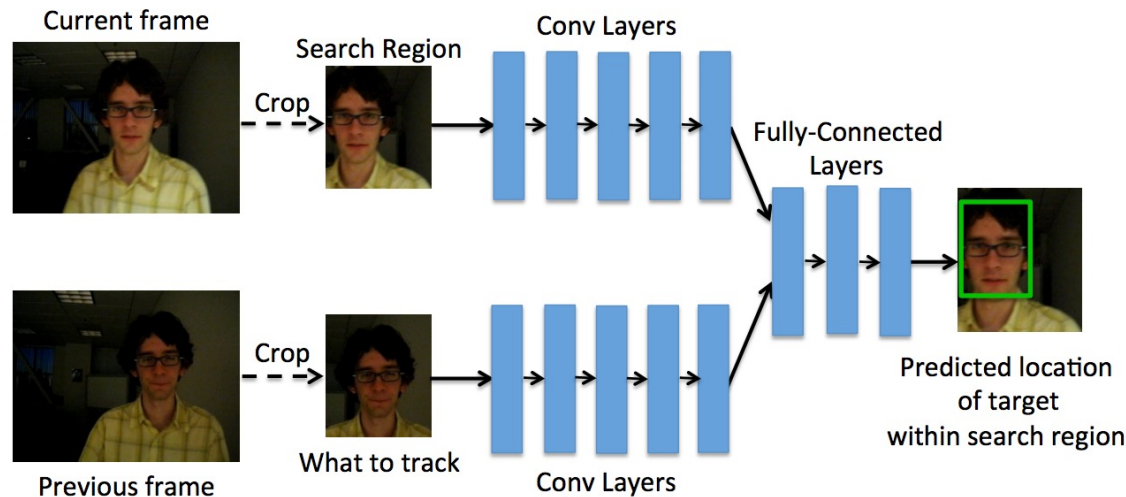


Fig. 2. Our network architecture for tracking. We input to the network a search region from the current frame and a target from the previous frame. The network learns to compare these crops to find the target object in the current image

Held, David, Sebastian Thrun, and Silvio Savarese. "Learning to track at 100 fps with deep regression networks." *European Conference on Computer Vision*. Springer, Cham, 2016. <https://arxiv.org/pdf/1604.01802>,
<https://learnopencv.com/goturn-deep-learning-based-object-tracking/>. Watch the video!

Previous
video frame
centered on
object



Current video frame,
shifted, with
ground-truth
bounding box

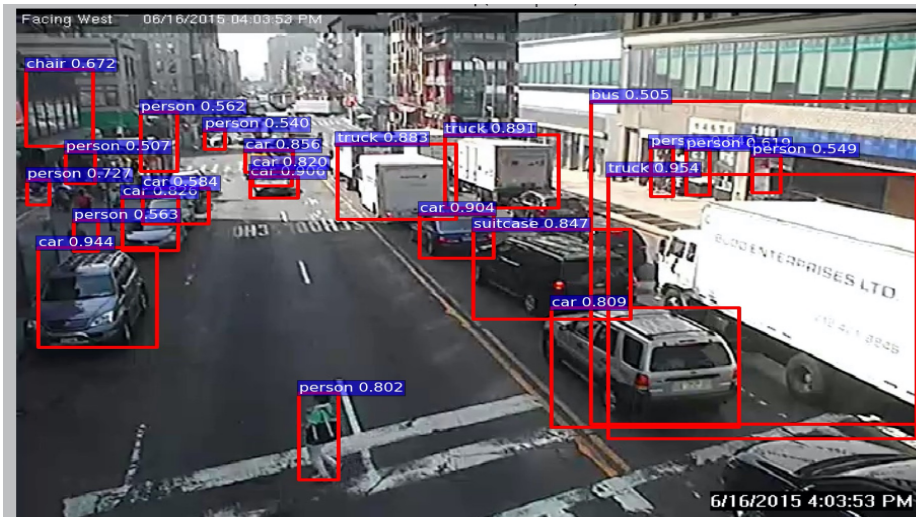
Held, David, Sebastian Thrun, and Silvio Savarese. "Learning to track at 100 fps with deep regression networks." *European Conference on Computer Vision*. Springer, Cham, 2016. <https://arxiv.org/pdf/1604.01802>,

Multiple Object Tracking Using Deep Learning

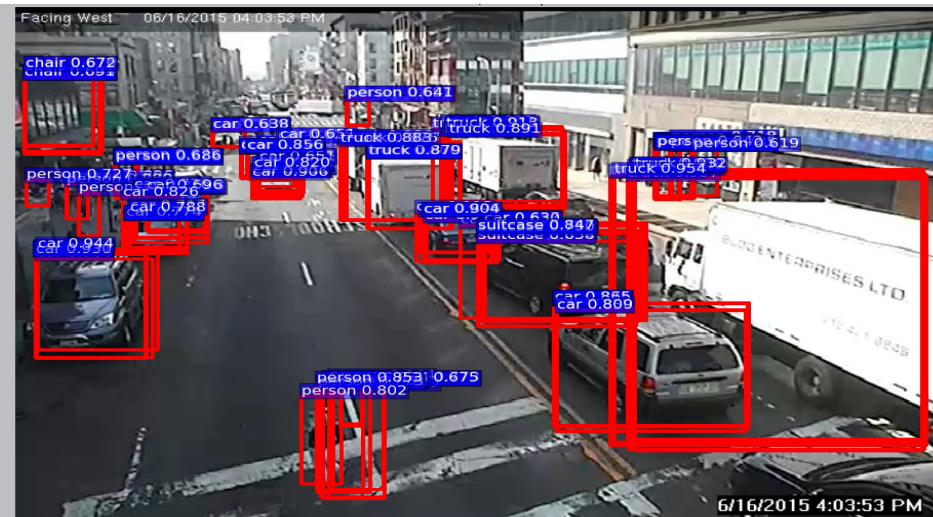
- Detect and track multiple objects
- Conventional approach
 - Detect individual objects in each frame, then associate corresponding objects (Detect and then track)
- Work at NYU video lab
 - Detect a “tube” in a video segment that contains the object in successive frames
 - Chenge Li, Gregory Dobler, Xin Feng, Yao Wang “[TrackNet: Simultaneous Object Detection and Tracking and Its Application in Traffic Video Analysis](https://arxiv.org/abs/1902.01466)”.

<https://arxiv.org/abs/1902.01466>

Tracking by Detection in Individual Frames (Conventional Approach)

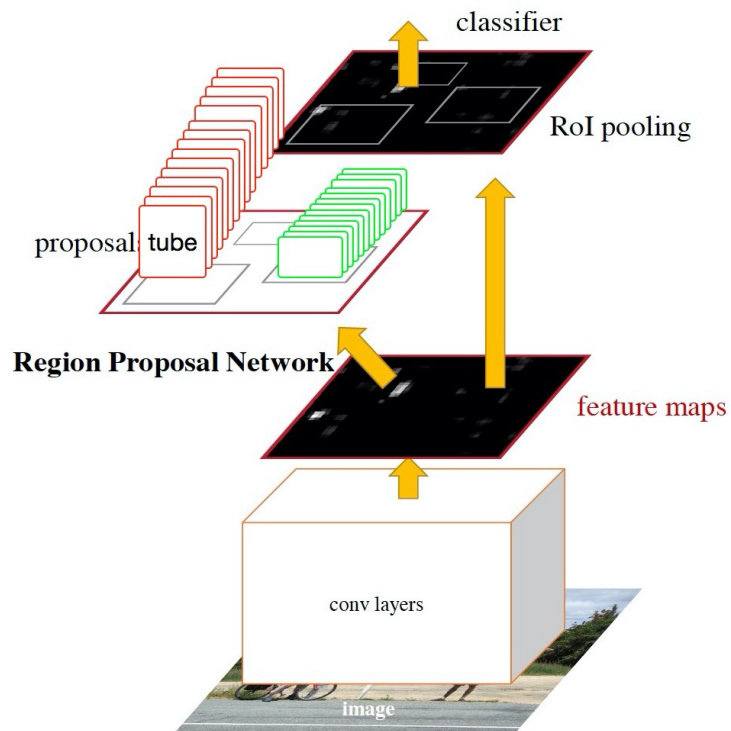


Single frame object detection
(results of faster-region-CNN on one frame)

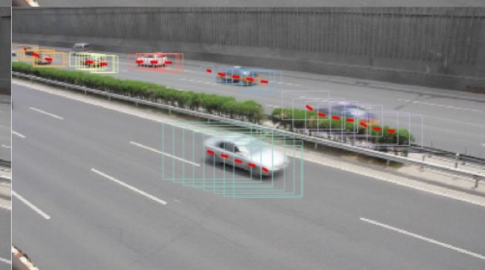
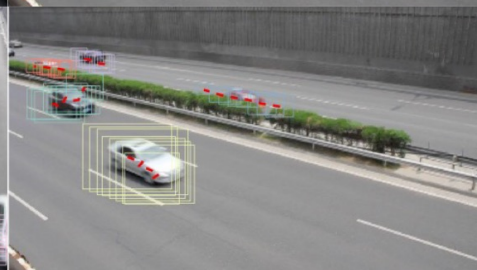
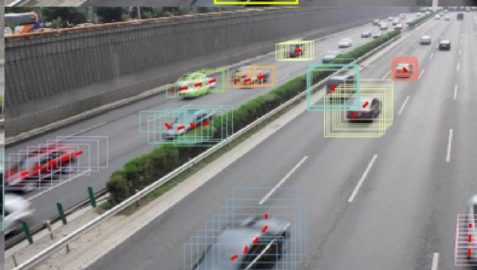
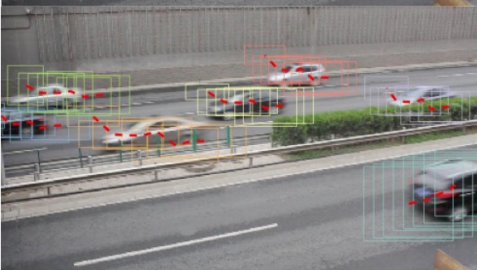
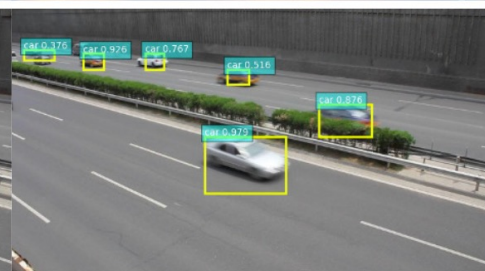
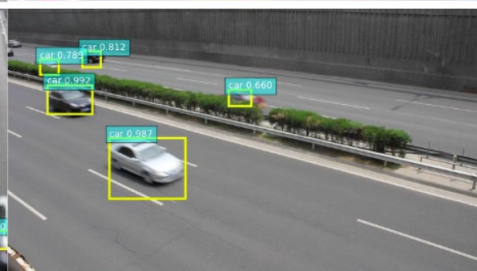
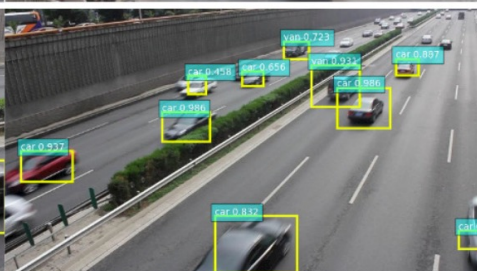
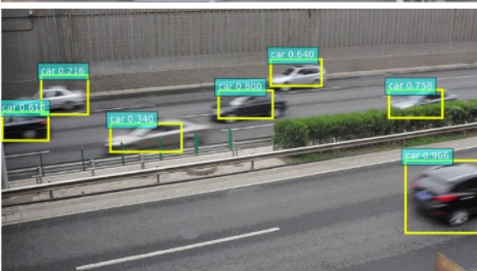
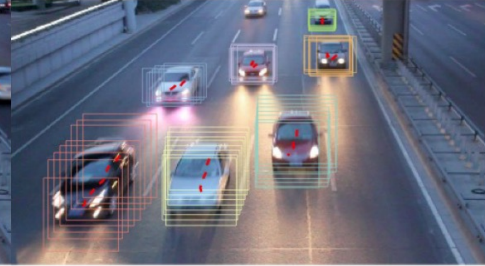
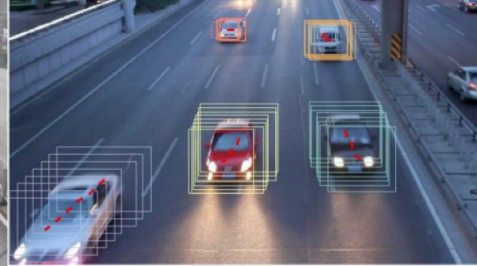
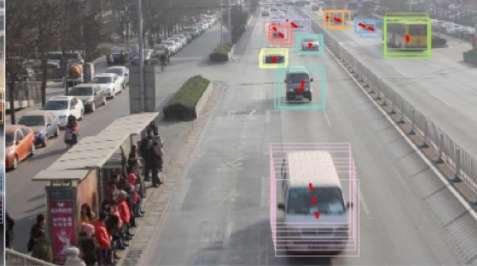
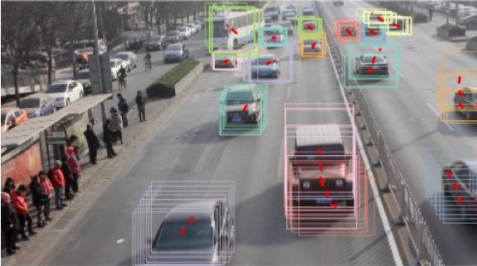
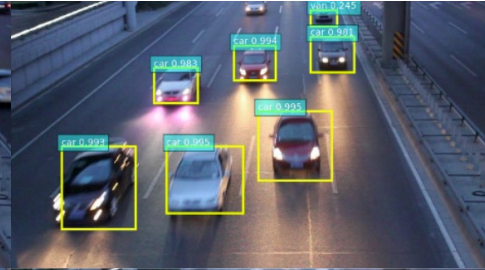
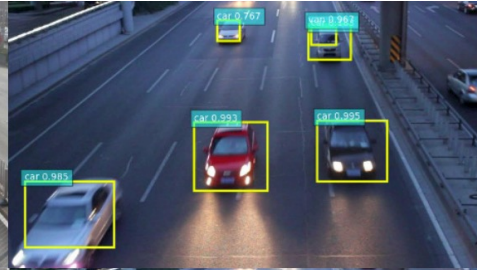
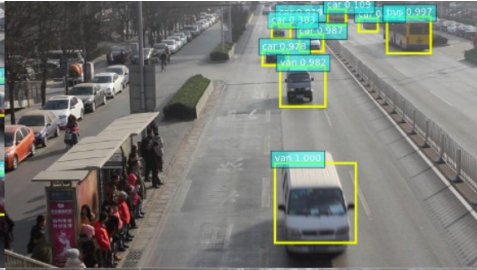
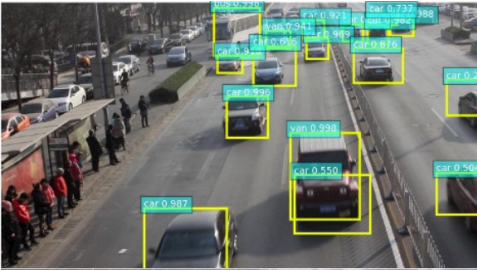


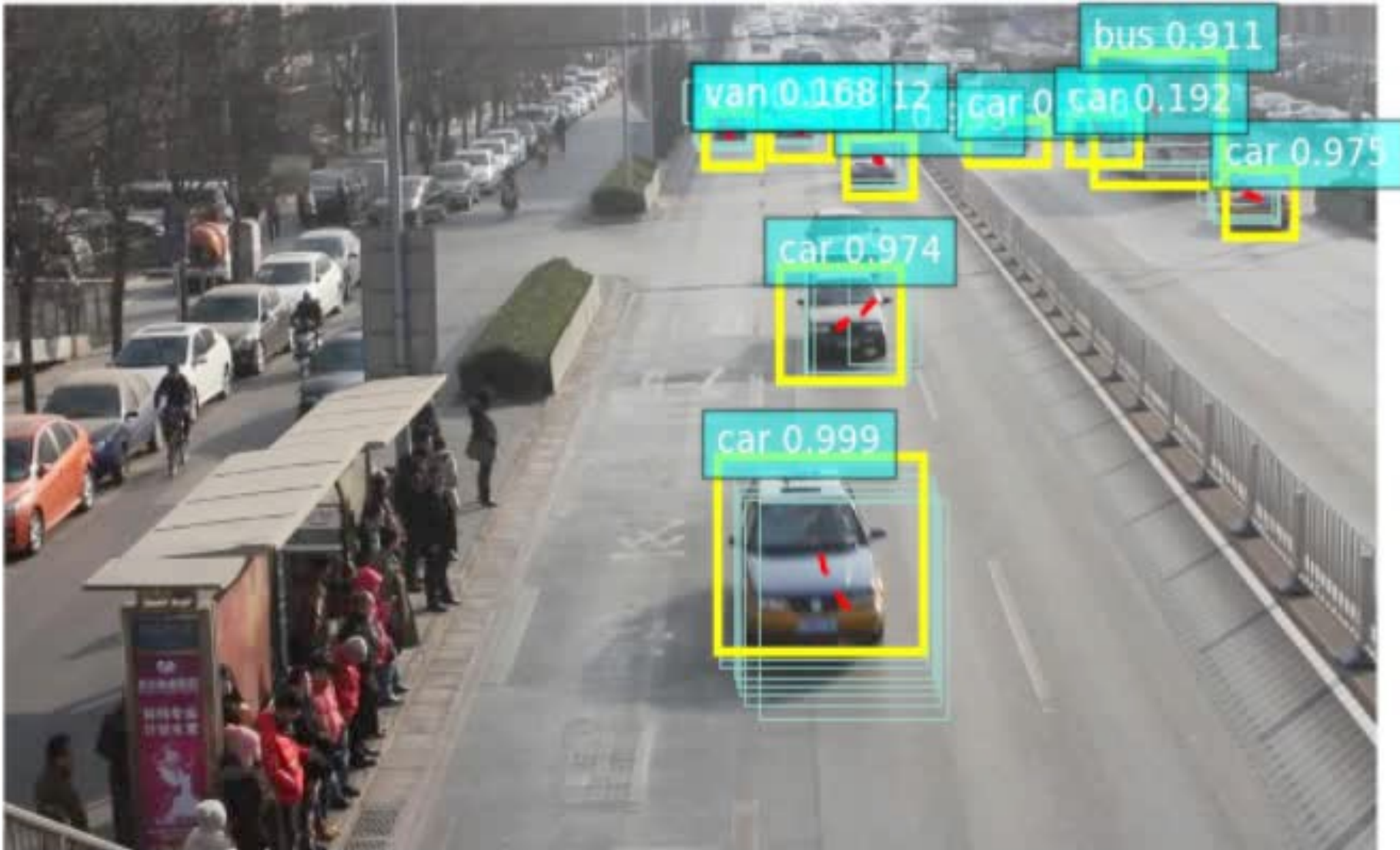
Multiple frame object detection
(results of applying faster-region-CNN on each
frame. Need to determine which boxes in different
frames correspond to the same object)

Extending faster Region-CNN For Moving Object Detection in Video

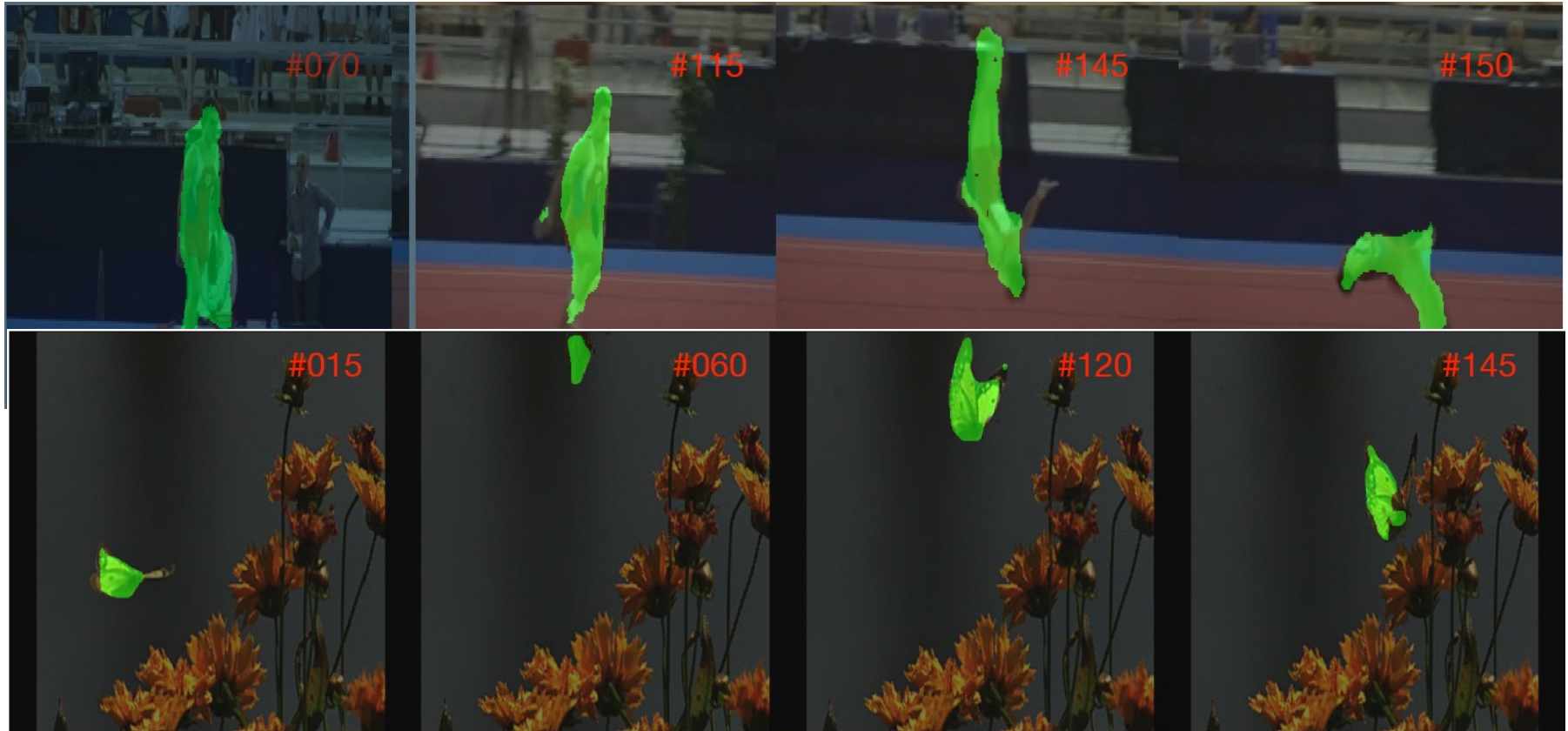


- Consider a video segment consisting of multiple frames
- Detecting a tube bounding each moving object
- Use 3D and 2D convolution for feature extraction (C3D and VGG)
- Generate tube proposals (bounding tubes of various sizes and orientations)
- Refine proposals and classify each detected tube (car, van, bus, pedestrian, ...)



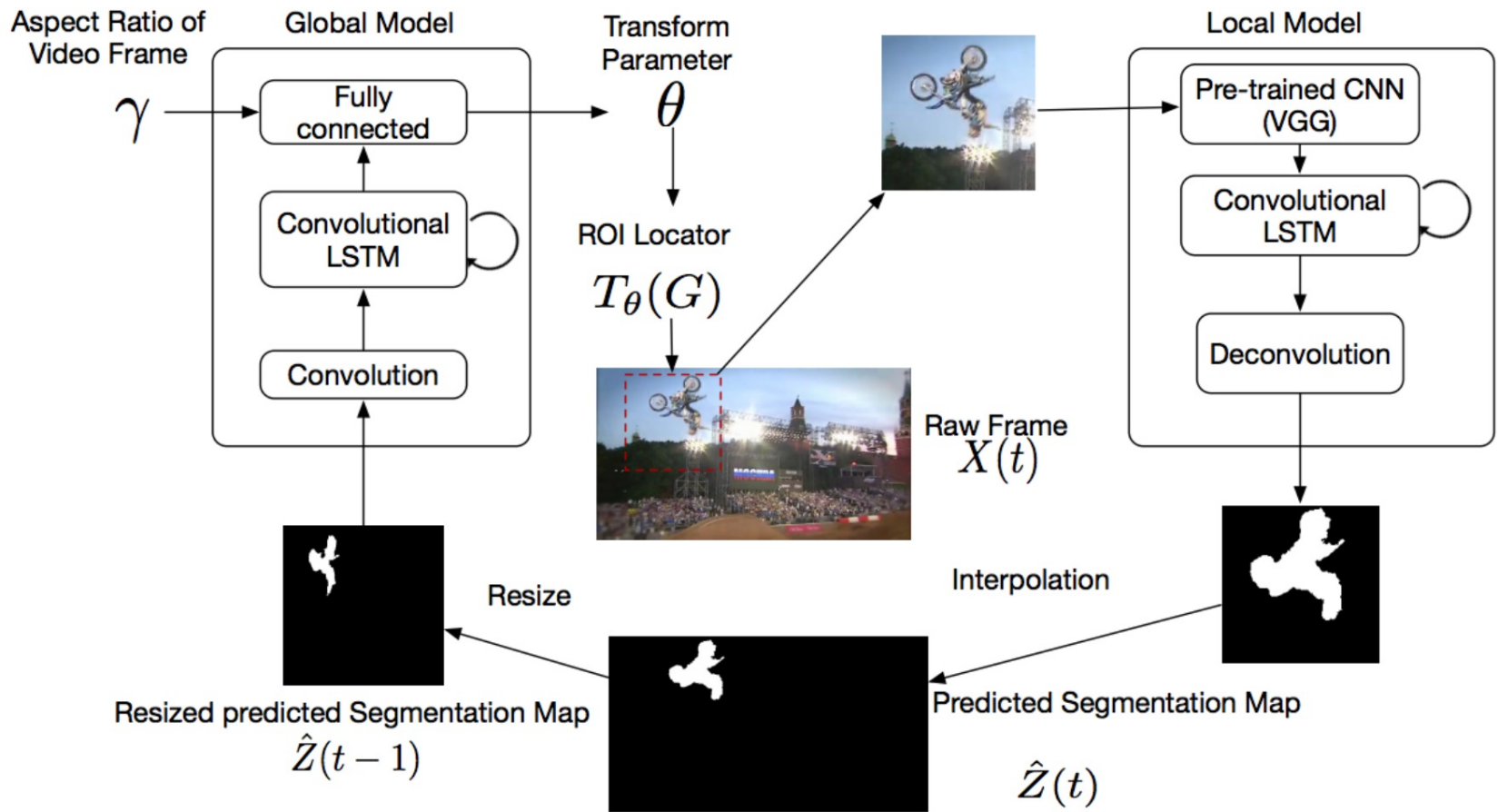


Pixel-Wise Object Tracking



Proposed Framework

- **Two stage tracking:**
 - The global model predicts a region of interests (RoI) in the new frame based on the segmentation masks of past frames.
 - Employs a convolutional LSTM structure to generate the latent feature characterizing the object motion.
 - The local model segments the RoI to identify pixels belonging to the object.
 - Also uses a convolutional LSTM structure whose memory state evolves with object appearance.
- The two-stage framework is robust to significant appearance shift, occlusion, and large motion and varying object sizes.
- Yilin Song, Chenge Li, Yao Wang “[Pixel-wise object tracking](https://arxiv.org/abs/1711.07377)“, Initial version: Nov. 2017, Last updated: July 2018. <https://arxiv.org/abs/1711.07377>





Summary (1): Camera Motion Estimation

- Camera motion induced motion field models:
 - Homography is accurate when the camera only rotates (or translation is small relative to the scene distance) or if the imaged scene is approximately flat
 - Affine is accurate when the camera motion is in plane (rotation, zooming, shifting)
- Direct method: Determine the motion parameters directly by minimizing the intensity matching errors
 - Minimizing the DFD error over all pixels in a frame
 - No closed form solution, can be solved using exhaustive search or gradient descent
 - Minimizing the optical flow equation error
 - Can lead to a linear equation with closed-form solutions
 - Should know how to set up the energy function, obtain the linear equation by setting the gradient to zero
 - Optical flow equation is only accurate if the motion at every pixel is small, which is usually not true for camera motion.
 - Can get around by iterative warping.
- Indirect method: Estimate the pixel-wise or block-wise motions first and then determine the motion parameters that best fit these motion vectors
 - Least squares fitting
 - Robust estimation
 - Similar to methods applied for feature-based correspondence

Summary (2): Video Stabilization

- Estimate the camera motion
- Smooth motion parameters in time
- Warp images to follow the smoothed motion (removing jittering)
- Fill in missing regions after warping

Summary (3): Background/Foreground Separation

- Simple approach for moving object detection using frame difference
- Simple approach for background modeling: Using average or median of frames to form the background image. Recursive update with outlier removal.
- Not robust to changes due to illumination, shadow, background dynamics, and noise.
- GMM and Low Rank plus sparse component decomposition (Robust PCA): Not required
- When the camera is moving, one has to estimate the camera motion and finding regions with different motion as objects.

Summary (4): Object Tracking

- Single object tracking
 - Simple method: Template matching
 - KLT tracker on feature points of a single object
- Multiple object tracking (optional)
 - KLT tracker + clustering of points based on motion model (same object should follow consistent motion)
- Deep learning methods for object tracking (optional)
 - Single objects
 - Multiple objects

Tools for motion estimation, object detection and tracking

- Python tools for motion estimation and object detection
 - http://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html
 - [cv2.calcOpticalFlowPyrLK\(\)](#)
 - This function computes the flow at a set of feature points, using pyramid representation
 - [cv2.calcOpticalFlowFarneback\(\)](#)
 - This function computes dense flow (at every pixel)
 - [cv2.BackgroundSubtractorMOG2\(\)](#)
 - [cv2.BackgroundSubtractorMOG\(\)](#)
- KLT tracker
 - <https://github.com/TimSC/PyFeatureTrack> (3rd party package)
- Tutorial on object detection and tracking using OpenCV
 - <https://www.intorobotics.com/how-to-detect-and-track-object-with-opencv/>

Useful Resources for Background Subtraction

- Background subtraction:
 - <http://bmc.iut-auvergne.com/> (some datasets)
 - <https://sites.google.com/site/backgroundsubtraction/> (algorithm, datasets, codes)
 - changedetection.net (large dataset)

Reading Assignments

- [Szeliski2010] Richard Szeliski, Computer Vision: Algorithms and Applications. 2021. Section 9.2.1, 9.4.4, 7.1.5.
- [Wang2002] Wang, et al, Digital video processing and communications. Sec. 6.7,6.8, Apx. A, B.
- **Other optional references:**
- Smeulders, A. W. M., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., and Shah, M. (2014). Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468.
- Yilmaz, Alper, Omar Javed, and Mubarak Shah. "Object tracking: A survey." *Acm computing surveys (CSUR)* 38.4 (2006): 13. <http://7xq232.com1.z0.glb.clouddn.com/talk/2013.12.20-Student.Workshop.pdf>
- Wu, Yi, Jongwoo Lim, and Ming-Hsuan Yang. "Online object tracking: A benchmark." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013. http://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Wu_Online_Object_Tracking_2013_CVPR_paper.pdf
- [Andrews Sobral and Antoine Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122:4-21, 2014](#)
- Stauffer, Chris, and W. Eric L. Grimson. "Adaptive background mixture models for real-time tracking." *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on..* Vol. 2. IEEE, 1999.
- Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. *Journal of the ACM (JACM)*, 58(3), 11.
- T. Bouwmans, "Traditional and Recent Approaches in Background Modeling for Foreground Detection: An Overview", *Computer Science Review*, 2014. [[pdf](#)]
- T. Bouwmans, E. Zahzah, "Robust PCA via Principal Component Pursuit: A Review for a Comparative Evaluation in Video Surveillance", *Special Issue on Background Models Challenge, Computer Vision and Image Understanding, CVIU 2014, Volume 122, pages 22–34, May 2014.* [[pdf](#)]

Written Assignment (1)

1. Consider two video frames taken when the camera underwent a rotation and translation between the first frame and the second frame. (a) Under what conditions can the mapping function between the two frames be approximated well by a homography? Would the motion field between the two frames also be approximated well by a homograph function? (b) Under what conditions can the mapping function be approximated well by an affine function? Would the motion field between the two frames also be approximated well by an affine function?
2. Suppose you know that the mapping function between two video frames can be characterized by a homography function, and you have already determined a dense motion field between the two frames. Let $(d_{n,x}, d_{n,y})$ denote the motion vector at n -th pixel (x_n, y_n) , for $n = 1, 2, \dots, N$, where N is the total number of pixels. Show how would you derive the homography parameters.
3. Consider a video taken while the camera undergoes irregular motion due to involuntary hand motion. Assume the global motion between every two frames can be approximated as a global shift. How would you create a stabilized video as if the camera is not moving at all? Describe all the major steps, and the algorithm for each step.

Written Assignment (2)

4. In the class, we discussed a simple method for moving object detection when the background is stationary: Taking the difference between two frames, and mark all pixels where the difference magnitude is above a threshold as belonging to moving objects. Would this method be appropriate if there are overall illumination changes between the frames due to the lighting condition change? If not, propose a simple pre-processing step to mitigate this problem.
5. We discussed the simple template matching method for object tracking in the class. Suppose you marked an object in frame $t-1$ using a box of size $W \times H$. You want to find the location of the object in frame t using template matching using a search range of $-S$ to S in both horizontal and vertical direction. Describe roughly how the template matching algorithm work, and what would be the complexity in terms of W, H, S . For simplicity, assume that you only consider integer motion of the object box.

Appendix (Optional Material)

- Intensity based global motion estimation: gradient descent methods
 - DFD error minimization
 - Optical flow equation based

Global Affine Transformation

- Affine mapping is a good approximation of the global motion due to camera motion, especially for far-away view
- Global Affine Transformation (6 parameters)

$$\psi_1(x,y) = \psi_2(x + d_x(x,y), y + d_y(x,y)) = \psi_2(x + \mathbf{A}(x,y)\mathbf{a}, y + \mathbf{A}(x,y)\mathbf{b})$$

$$\begin{bmatrix} d_x(x,y) \\ d_y(x,y) \end{bmatrix} = \begin{bmatrix} a_0 + a_1x + a_2y \\ b_0 + b_1x + b_2y \end{bmatrix} = \begin{bmatrix} \mathbf{A}(x,y) & 0 \\ 0 & \mathbf{A}(x,y) \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$$

$$\mathbf{A}(x,y) = \begin{bmatrix} 1 & x & y \end{bmatrix}, \mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

- Special cases:
 - Translation only: $a_0 = x\text{-direction shift}, a_1 = 0, a_2 = 0$
 $b_0 = y\text{-direction shift}, a_1 = 0, a_2 = 0$

Direct Estimation of Affine Motion: Minimizing DFD Error

- Parameterize the DFD error in terms of the motion parameters, and estimate these parameters by minimizing the DFD error

$$E_{\text{DFD}} = \sum_{n \in \mathcal{N}} w_n |\psi_2(\mathbf{x}_n + \mathbf{d}(\mathbf{x}_n; \mathbf{a})) - \psi_1(\mathbf{x}_n)|^p$$

Weighting w_n coefficients depend on the importance of pixel \mathbf{x}_n .

Ex: Affine motion:
$$\begin{bmatrix} d_x(\mathbf{x}_n; \mathbf{a}) \\ d_y(\mathbf{x}_n; \mathbf{a}) \end{bmatrix} = \begin{bmatrix} a_0 + a_1 x_n + a_2 y_n \\ b_0 + b_1 x_n + b_2 y_n \end{bmatrix} = \mathbf{A}(\mathbf{x}_n) \mathbf{a}$$

$$\mathbf{A}(\mathbf{x}_n) = \begin{bmatrix} 1 & x_n & y_n & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_n & y_n \end{bmatrix}$$
$$\mathbf{a} = [a_0, a_1, a_2, b_0, b_1, b_2]^T$$

Exhaustive search of all 6 parameters is computationally prohibitive!
Using gradient descent method.

Direct Estimation of Affine Motion Using Gradient Descent Method

$$E_{\text{DFD}}(\mathbf{a}, \mathbf{b}) = \frac{1}{2} \sum_{\mathbf{x} \in B(\mathbf{x}_n)} |\psi_2(\mathbf{x} + \mathbf{A}(\mathbf{x}, \mathbf{y})\mathbf{a}, \mathbf{y} + \mathbf{A}(\mathbf{x}, \mathbf{y})\mathbf{b}) - \psi_1(\mathbf{x}, \mathbf{y})|^2 \rightarrow \min$$

$$\mathbf{A}(x, y) = \begin{bmatrix} 1 & x & y \end{bmatrix}, \mathbf{a}^T = \begin{bmatrix} a_0 & a_1 & a_2 \end{bmatrix}, \mathbf{b}^T = \begin{bmatrix} b_0 & b_1 & b_2 \end{bmatrix}$$

B refers to whole frame. Should make frame center has coordinates $x=0, y=0$

$$\frac{\partial E}{\partial \mathbf{a}} = \begin{bmatrix} \frac{\partial E}{\partial a_0} \\ \frac{\partial E}{\partial a_1} \\ \frac{\partial E}{\partial a_2} \end{bmatrix} = \sum_{(x,y) \in B} e(x,y) \frac{\partial \psi_2}{\partial \mathbf{x}} \Big|_{(\mathbf{x} + \mathbf{A}(x,y)\mathbf{a}, \mathbf{y} + \mathbf{A}(x,y)\mathbf{b})} \quad \mathbf{A}(x,y)^T = \begin{bmatrix} \sum_{(x,y) \in B} e(x,y) G_x(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y) \\ \sum_{(x,y) \in B} e(x,y) G_x(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y)x \\ \sum_{(x,y) \in B} e(x,y) G_x(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y)y \end{bmatrix}$$

$$\frac{\partial E}{\partial \mathbf{b}} = \begin{bmatrix} \frac{\partial E}{\partial b_0} \\ \frac{\partial E}{\partial b_1} \\ \frac{\partial E}{\partial b_2} \end{bmatrix} = \sum_{(x,y) \in B} e(x,y) \frac{\partial \psi_2}{\partial \mathbf{y}} \Big|_{(\mathbf{x} + \mathbf{A}(x,y)\mathbf{a}, \mathbf{y} + \mathbf{A}(x,y)\mathbf{b})} \quad \mathbf{A}(x,y)^T = \begin{bmatrix} \sum_{(x,y) \in B} e(x,y) G_y(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y) \\ \sum_{(x,y) \in B} e(x,y) G_y(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y)x \\ \sum_{(x,y) \in B} e(x,y) G_y(x+a_0+a_1x+a_2y, y+b_0+b_1x+b_2y)y \end{bmatrix}$$

$e(x, y) = \psi_2(\mathbf{x} + \mathbf{A}(x, \mathbf{y})\mathbf{a}, \mathbf{y} + \mathbf{A}(x, \mathbf{y})\mathbf{b}) - \psi_1(\mathbf{x}, \mathbf{y})$: Current prediction error image;

$G_x(x, y) = \frac{\partial \psi_2}{\partial x}(x, y)$: Gradient image in x-direction; $G_y(x, y) = \frac{\partial \psi_2}{\partial y}(x, y)$: Gradient image in y-direction

Solving Affine Mapping Using Optical Flow Constraint

Optical Flow Equation: $\frac{\partial \psi_2}{\partial x} d_x + \frac{\partial \psi_2}{\partial y} d_y + \psi_2(x, y) - \psi_1(x, y) \Rightarrow$

$$\frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y) \mathbf{a} + \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y) \mathbf{b} + \psi_2(x, y) - \psi_1(x, y) = 0$$

We can find \mathbf{a}, \mathbf{b} by minimizing the following energy cost:

$$E_{OF}(\mathbf{a}, \mathbf{b}) = \frac{1}{2} \sum_{\mathbf{x} \in B} \left| \frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y) \mathbf{a} + \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y) \mathbf{b} + \psi_2(x, y) - \psi_1(x, y) \right|^2 \rightarrow \min \quad \text{B includes all pixels in a frame}$$

Set:

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{a}} &= \sum_{\mathbf{x} \in B} \left(\frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y) \mathbf{a} + \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y) \mathbf{b} + \psi_2(x, y) - \psi_1(x, y) \right) \frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y)^T \\ &= \sum_{\mathbf{x} \in B} \left(\frac{\partial \psi_2}{\partial x} \frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \mathbf{a} + \frac{\partial \psi_2}{\partial x} \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \mathbf{b} + (\psi_2(x, y) - \psi_1(x, y)) \frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y)^T \right) = 0 \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{b}} &= \sum_{\mathbf{x} \in B} \left(\frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y) \mathbf{a} + \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y) \mathbf{b} + \psi_2(x, y) - \psi_1(x, y) \right) \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \\ &= \sum_{\mathbf{x} \in B} \left(\frac{\partial \psi_2}{\partial x} \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \mathbf{a} + \frac{\partial \psi_2}{\partial y} \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \mathbf{b} + (\psi_2(x, y) - \psi_1(x, y)) \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \right) = 0 \end{aligned}$$

This leads to a linear equation

$$\begin{bmatrix} \sum_{\mathbf{x} \in B} \left(\frac{\partial \psi_2}{\partial x} \frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \right) & \sum_{\mathbf{x} \in B} \left(\frac{\partial \psi_2}{\partial x} \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \right) \\ \sum_{\mathbf{x} \in B} \left(\frac{\partial \psi_2}{\partial x} \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \right) & \sum_{\mathbf{x} \in B} \left(\frac{\partial \psi_2}{\partial y} \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \right) \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \sum_{\mathbf{x} \in B} (\psi_1(x, y) - \psi_2(x, y)) \frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y)^T \\ \sum_{\mathbf{x} \in B} (\psi_1(x, y) - \psi_2(x, y)) \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \end{bmatrix}$$

A Closer Look at the Equation ...

$$\begin{bmatrix} \sum_{x \in B} \left(\frac{\partial \psi_2}{\partial x} \frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \right) & \sum_{x \in B} \left(\frac{\partial \psi_2}{\partial x} \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \right) \\ \sum_{x \in B} \left(\frac{\partial \psi_2}{\partial x} \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \right) & \sum_{x \in B} \left(\frac{\partial \psi_2}{\partial y} \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \mathbf{A}(x, y) \right) \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \sum_{x \in B} (\psi_1(x, y) - \psi_2(x, y)) \frac{\partial \psi_2}{\partial x} \mathbf{A}(x, y)^T \\ \sum_{x \in B} (\psi_1(x, y) - \psi_2(x, y)) \frac{\partial \psi_2}{\partial y} \mathbf{A}(x, y)^T \end{bmatrix} \Rightarrow \mathbf{S} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \mathbf{t}$$

$$\mathbf{B}(x, y) = \mathbf{A}(x, y)^T \mathbf{A}(x, y) = \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \begin{bmatrix} 1 & x & y \end{bmatrix} = \begin{bmatrix} 1 & x & y \\ x & x^2 & xy \\ y & xy & y^2 \end{bmatrix}$$





























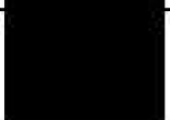













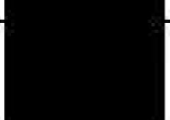














$$\mathbf{S} = \begin{bmatrix} \sum (G_x(x, y))^2 \mathbf{B}(x, y) & \sum G_x(x, y) G_y(x, y) \mathbf{B}(x, y) \\ \sum G_x(x, y) G_y(x, y) \mathbf{B}(x, y) & \sum (G_y(x, y))^2 \mathbf{B}(x, y) \end{bmatrix}$$

$$\mathbf{t} = \begin{bmatrix} \sum e(x, y) G_x(x, y) \\ \sum e(x, y) G_x(x, y) x \\ \sum e(x, y) G_x(x, y) y \\ \sum e(x, y) G_y(x, y) \\ \sum e(x, y) G_y(x, y) x \\ \sum e(x, y) G_y(x, y) y \end{bmatrix}; \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \end{bmatrix} = \mathbf{S}^{-1} \mathbf{t}$$

Background Modeling using GMM

- Modeling the colors at each pixel using a Gaussian mixture model (GMM) (aka mixture of Gaussian or MoG)
 - Recursively update the GMM parameter at each pixel
- Initial paper:
 - Stauffer, Chris, and W. Eric L. Grimson. "Adaptive background mixture models for real-time tracking." *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on..* Vol. 2. IEEE, 1999.
 - http://www.ai.mit.edu/projects/vsam/Publications/stauffer_cvpr98_track.pdf
- A good review:
 - T. Bouwmans, F. El-Baf, and B. Vachon. Background modeling using mixture of Gaussians for foreground detection: A survey. *Recent Patents on Computer Science* , 1(3):219–237, November 2008.
- Not required for this class

Foreground Detection using GMM-based Approaches

Sequence	MO	TD	LS	WT	C	B	FA
Test image							
Ground Truth							
Stauffer <i>et al.</i> [1] MOG							
White <i>et al.</i> [138] MOG with PSO							
Setiawan <i>et al.</i> [118] MOG using IHLS							
Wang <i>et al.</i> [70] Improved MOG							
Schindler <i>et al.</i> [153] MOG with MRF							
Cristani <i>et al.</i> [43] S-TAPMOG	-	-	-				
Cristani <i>et al.</i> [92] ASTNA	-	-	-				

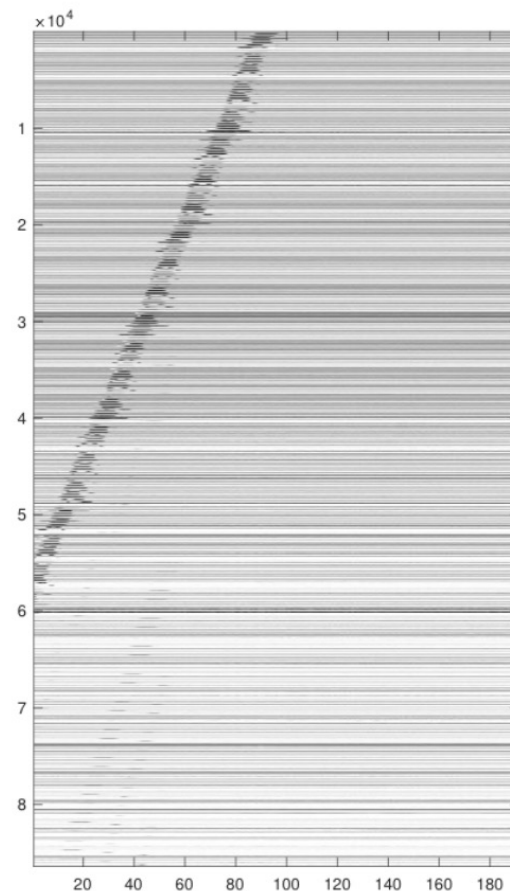
From: T. Bouwmans, F. El-Baf, and B. Vachon. Background modeling using mixture of Gaussians for foreground detection: A survey. Recent Patents on Computer Science , 1(3):219–237, November 2008.

Background Separation

Using Low Rank + Sparse Decomposition

- Main idea
 - Reorder pixels in each frame into a vector
 - Put successive frame vectors as columns of a matrix \mathbf{M}
 - If all the frames are the same (stationary), then all columns will be the same, the matrix has rank 1
 - If all the frames only differ by a scale factor (e.g. due to illumination change), the matrix still has rank 1
 - If all the frames vary from each other slightly, generally the matrix has a low rank (each frame is a linear combination of a few other frames)
 - If there is a moving object in the scene, the matrix may not be low rank any more
 - Generally, \mathbf{M} may be decomposed into a low rank matrix \mathbf{L} (corresponding to slowly changing background) and a sparse matrix \mathbf{S} (corresponding to moving foreground, occupying only a sparse set of pixels)

Stacking Video frames in a Matrix



How to find L and S from M?

$$M = L + S$$

L: Background, low rank

S: Moving foreground, sparse

Rank of a Matrix and Singular Vector Decomposition (SVD)

- Rank = # of independent columns (or rows) in a matrix
- Rank = # non-zero singular values of the matrix
- SVD: any matrix can be decomposed as

$$X = U\Lambda V^T = \sum_{r=1}^R \lambda_r \mathbf{u}_r \mathbf{v}_r^T$$

Principle Component Analysis (PCA)

- Original formulation of PCA
 - Given observation vectors x_i , form matrix $M=[x_1, x_2, \dots, x_N]$
 - Covariance matrix $C=M M^T$
 - Principle components = Eigenvectors of C : $Cu_i = \lambda_i u_i$,
 - SVD of M : $M = USV^T$
 - $C = M M^T = USV^T VSU^T = US^2U^T$, $CU = US^2$
 - Principle components can be found using SVD on M : u_i is eigenvector with eigenvalue $\lambda_i = s_i^2$
- Another interpretation of PCA
 - Finding a low rank approximation of M with minimal L2 error
 - $\min \|M - L\|_2$, subj to $\text{rank}(L) = K$
 - L =SVD of M with K largest singular values:
 - $M = USV^T \rightarrow L = US_KV^T$

Robust PCA (RPCA)

- PCA: $\min \|M - L\|_2$, subj to $\text{rank}(L) = K$
 - the principle components are greatly affected by outliers (noise with large values)
- Robust PCA: $\min \text{Rank}(L) + \lambda \|S\|_0$, subj to $L + S = M$
 - S represent “outliers”, which occur rarely but can be large
 - RPCA=Low Rank+Sparse Decomposition!
- Under mild conditions, RPCA is equivalent to solve
$$\min \|L\|_* + \lambda \|S\|_1, \text{ subj to } L + S = M.$$
 - Also known as Principle Component Pursuit or PCP
 - Can be solved using ADMM

Low-rank+Sparse Decomposition

- Given M , determine L and S , so that
 - $M=L+S$, L is low rank, S is sparse (Small L_0 norm)
- Mathematical formulation
 - $\min \text{Rank}(L) + \lambda \|S\|_0$, subj to $L + S = M$. (Original problem)
 - Hard to solve!
- Candès et al and Wright et al proved, under some conditions and for a suitably chosen λ , the above problem is equivalent to
 - $\min \|L\|_* + \lambda \|S\|_1$, subj to $L + S = M$. (Convex relaxed problem)
 - $\|L\|_*$ is the **Nuclear Norm** of L (Sum of singular values of L)
 - Convex problem, and can be solved through ADMM (iterative SVD and soft thresholding)

- Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. *Journal of the ACM (JACM)*, 58(3), 11.
- Wright, J., Ganesh, A., Rao, S., Peng, Y., & Ma, Y. (2009). Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems* (pp. 2080-2088).
- <https://sites.google.com/site/backgroundsubtraction/available-implementation/recent-background-modeling/background-modeling-via-rpca>

Alternating direction method of multipliers (ADMM, Review)

- ▶ ADMM problem form (with f, g convex)

$$\begin{aligned} & \text{minimize} && f(x) + g(z) \\ & \text{subject to} && Ax + Bz = c \end{aligned}$$

Typical use case:
 $f(x)$ is quadratic in x
 $g(z)$ contains L1 norm
 $B = \text{diagonal}$

- two sets of variables, with separable objective

Can be grouped as $\rho/2 \|y/\rho + (Ax+Bz-c)\|^2$ by completing square

- ▶ $L_\rho(x, z, y) = f(x) + g(z) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2$

- ▶ ADMM: Minimizing a quadratic problem, with closed-form solution

$$x^{k+1} := \underset{x}{\operatorname{argmin}} L_\rho(x, z^k, y^k) \quad // \textit{x-minimization}$$

$$z^{k+1} := \underset{z}{\operatorname{argmin}} L_\rho(x^{k+1}, z, y^k) \quad // \textit{z-minimization}$$

Soft thresholding if $B = \text{diagonal}$

$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \quad // \textit{dual update}$$

Cropped from: https://web.stanford.edu/~boyd/papers/pdf/admm_slides.pdf

ADMM Solution of L+S

- Original relaxed problem:
$$\begin{array}{ll} \text{minimize} & \|L\|_* + \lambda \|S\|_1 \\ \text{subject to} & L + S = M \end{array}$$

- Augmented Lagrangian:

$$l(L, S, Y) = \|L\|_* + \lambda \|S\|_1 + \langle Y, M - L - S \rangle + \frac{\mu}{2} \|M - L - S\|_F^2.$$

- ADMM: Solve L and S alternately, each with closed form solution

L-minimization:
$$\arg \min_L l(L, S, Y) = \mathcal{D}_{1/\mu}(M - S + \mu^{-1}Y)$$

$$\mathcal{D}_\tau(X) = U \mathcal{S}_\tau(\Sigma) V^*, \text{ where } X = U \Sigma V^*$$

Find the SVD of $X = M - S + \mu^{-1}Y$,
soft hresholding singular values with threshold μ^{-1}

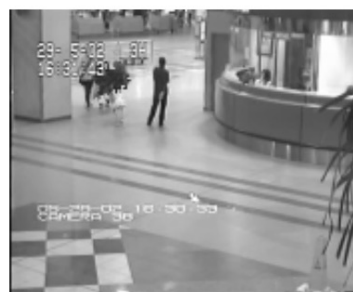
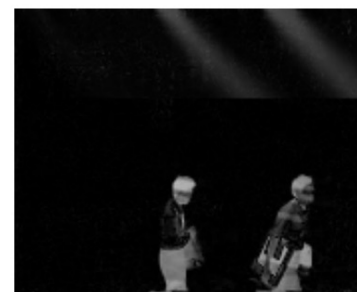
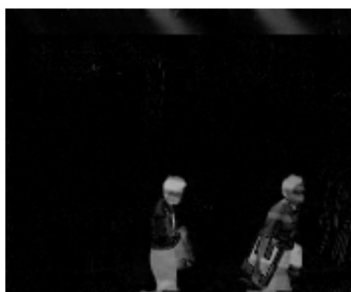
S-minimization:
$$\arg \min_S l(L, S, Y) = \mathcal{S}_{\lambda/\mu}(M - L + \mu^{-1}Y)$$
$$\mathcal{S}_\tau[x] = \text{sgn}(x) \max(|x| - \tau, 0)$$

L+S Using ADMM

ALGORITHM 1: (Principal Component Pursuit by Alternating Directions Yuan and Yang 2009)]

- 1: **initialize:** $S_0 = Y_0 = 0, \mu > 0.$
 - 2: **while** not converged **do**
 - 3: compute $L_{k+1} = \mathcal{D}_{1/\mu}(M - S_k + \mu^{-1}Y_k);$
 - 4: compute $S_{k+1} = \mathcal{S}_{\lambda/\mu}(M - L_{k+1} + \mu^{-1}Y_k);$
 - 5: compute $Y_{k+1} = Y_k + \mu(M - L_{k+1} - S_{k+1});$
 - 6: **end while**
 - 7: **output:** $L, S.$
-

From: Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. *Journal of the ACM (JACM)*, 58(3), 11.



(a) Original frames

(b) Low-rank \hat{L}
Convex optimization (this work)

(c) Sparse \hat{S}

(d) Low-rank \hat{L}
Alternating minimization [47]

(e) Sparse \hat{S}

From: Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. *Journal of the ACM (JACM)*, 58(3), 11.

Variations of RPCA (not required)

- Complexity issue:
 - ADMM requires solving SVD in each iteration
 - Many fast algorithms have been developed
- Robustness to noise:
 - Beyond the moving object, variations in the measurement matrix M may be due to camera noise, dynamic background (moving tree leaves, ocean waves, etc)
 - Requiring $M=L+S$ exactly may not be appropriate
 - Stable PCP:
 - $M= L+S+E$, where E represents small random variations
 - $\min \|L\|_* + \lambda \|S\|_1$, subj to $\|M - L - S\|_F \leq \delta$
 - Alternate formulation
 - $\min \|L\|_* + \lambda_1 \|S\|_1 + \lambda_2 \|M - L - S\|_F^2$
- T. Bouwmans, E. Zahzah, “Robust PCA via Principal Component Pursuit: A Review for a Comparative Evaluation in Video Surveillance”, Special Issue on Background Models Challenge, Computer Vision and Image Understanding, CVIU 2014, Volume 122, pages 22–34, May 2014. [[pdf](#)]

Pop Quiz

- Is thresholding the frame difference a good method to detect moving objects?
- Where does it fail?

- What is the assumption of RPCA method?
- Can RPCA handle lighting changes of background?
- Can RPCA handle dynamic background (tree leaves, water movement)?
- What is the objective function that RPCA minimizes?
- How to find the optimal solution?

Video Shot Boundary Detection (or Scene Change Detection)

- A video often contains different shots, each has a coherent scene
- How to divide a video into separate shots or detect scene change?
 - An important first step for video analysis
- Simple approach:
 - Frame difference: if sum of DFD is large, there is a scene change
 - Sensitive to changes due to camera motion, object motion, illumination variation
 - Does not work well in gradual transitions
- More advanced approaches
 - Based on difference in color histogram, entropy of color distribution
 - Machine learning based approach

TRECVID Competition

- TRECVID (TREC Video Retrieval Evaluation) is sponsored by NIST to encourage research in digital video indexing and retrieval. It has focused on different video analysis tasks. Shot boundary detection was one
 - <http://trecvid.nist.gov/>
- Smeaton, Alan F., Paul Over, and Aiden R. Doherty. "Video shot boundary detection: Seven years of TRECVID activity." *Computer Vision and Image Understanding* 114.4 (2010): 411-418.
<http://doras.dcu.ie/4080/1/sbretro.pdf>. Contain results up to 2005
- Liu, Z., Gibbon, D., Zavesky, E., Shahraray, B., & Haffner, P. (2007, November). AT&T research at trecvid 2006. In *Proc. TRECVID Workshop* (pp. 19-26). (best in TRECVID2006 SBD Competition)
https://www.researchgate.net/profile/Behzad_Shahraray/publication/224718827_A_Fast_Comprehensive_Shot_Boundary_Determination_System/links/02e7e51a8b6cea0546000000.pdf