5. Feature Tracking

Homography estimation (previous lecture), Optical Flow, Feature Tracking, Motion Segmentation

Estimating transformations from Image features

- If point correspondences (x,y)<-->(x',y') are known

We can find a 3X3 transformation, **A** such that
x' = **Ax**

A = $[a_1, a_2, a_3; a_4, a_5, a_6; a_7, a_8, a_9]$

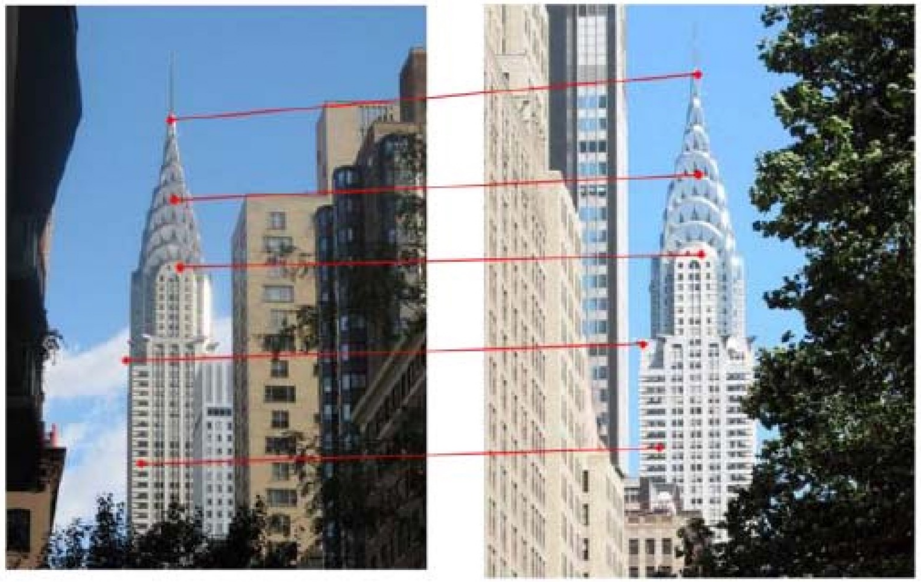
Remember from first lecture on transformations, a 2D projective transformation (most generic transformation) has 8 dof

Assume $a_9 = 1$, therefore

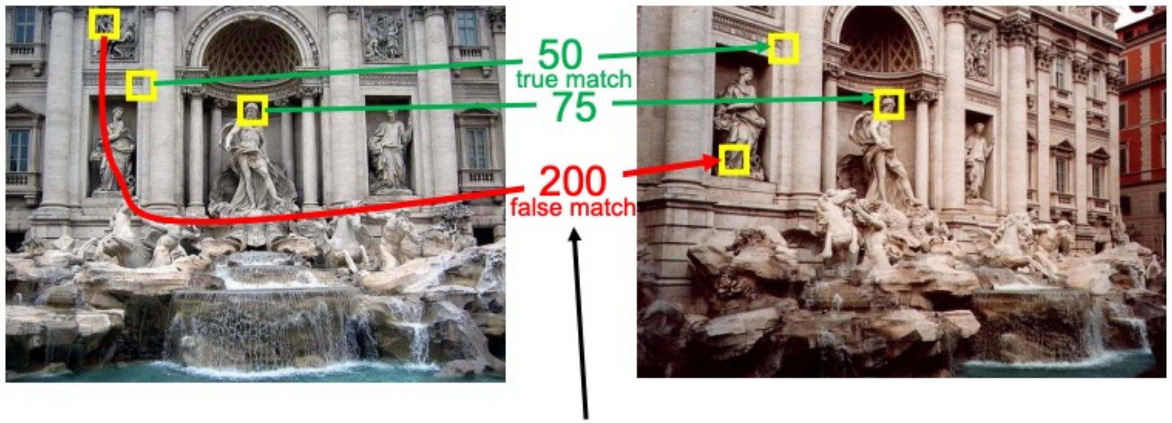
$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

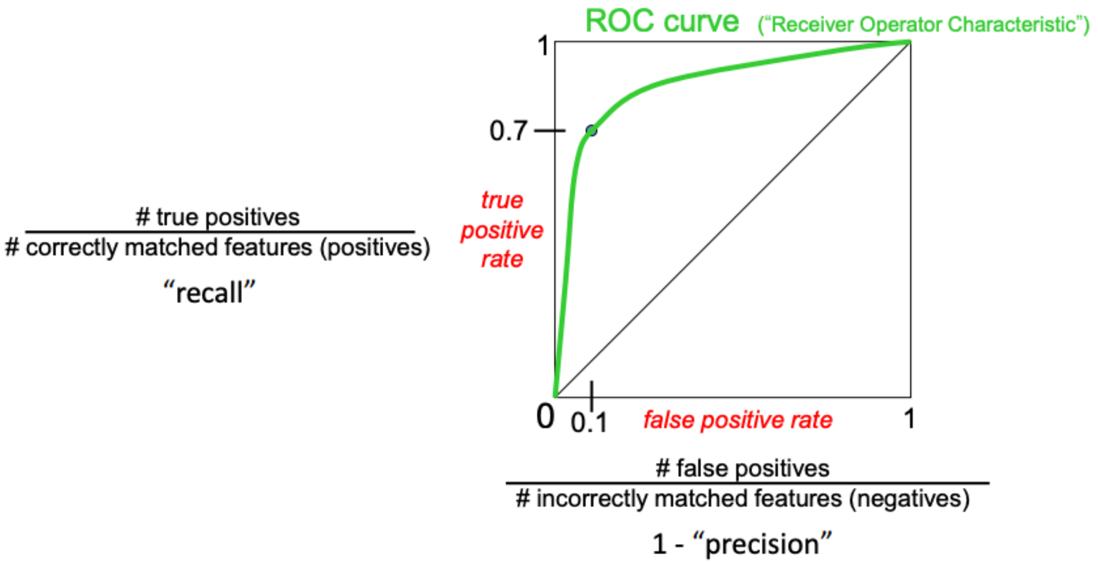
x = (x,y) **x'** = (x',y')



Feature Matching Performance



Feature distance



The distance threshold affects performance

- True positives = # of detected matches that are correct
- Suppose we want to maximize these—how to choose threshold?
- False positives = # of detected matches that are incorrect
- Suppose we want to minimize these—how to choose threshold?

Estimating transformations from Image features

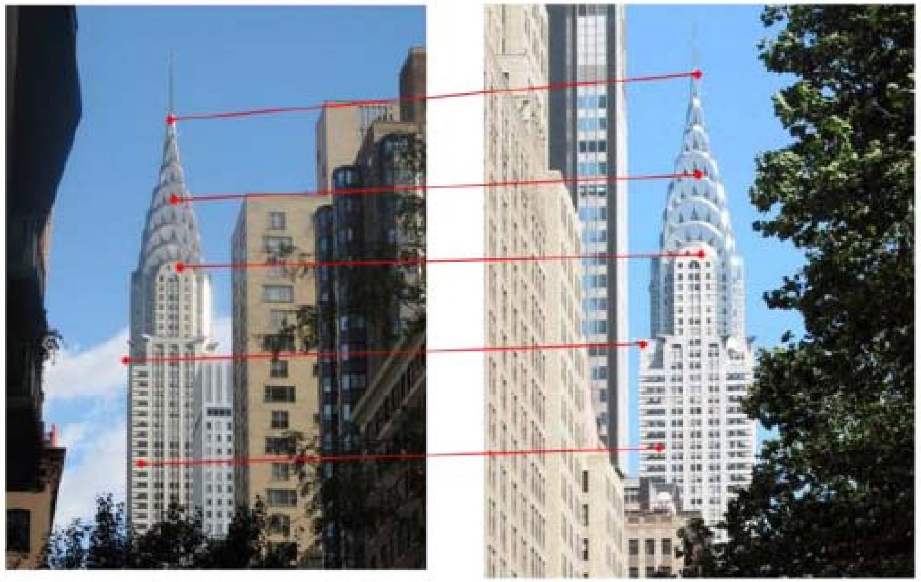
Assume $a_9 = 1$, therefore

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

$$a_7 x'x + a_8 x'y + x' = a_1 x + a_2 y + a_3$$
$$a_7 y'x + a_8 y'y + y' = a_7 x + a_8 y + a_6$$
$$x' = a_1 x + a_2 y + a_3 - a_7 x'x - a_8 x'y$$
$$y' = a_7 x + a_8 y + a_6 - a_7 y'x - a_8 y'y$$

x = (x,y) **x'** = (x',y')



Segregate the equations into matrices/vectors of knowns and unknowns

Estimating transformations from Image features

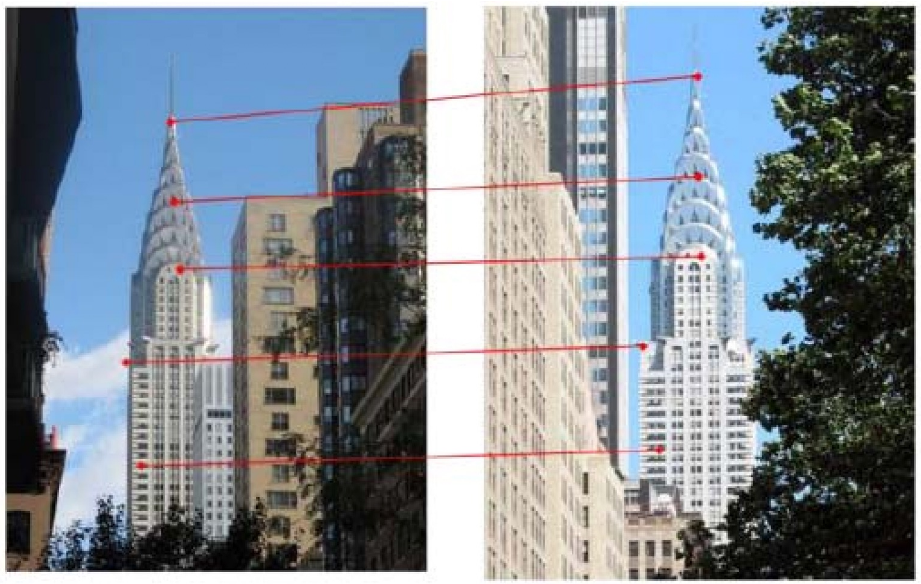
$$a_7 x'x + a_8 x'y + x' = a_1 x + a_2 y + a_3$$

$$a_7 y'x + a_8 y'y + y' = a_7 x + a_8 y + a_6$$

$$x' = a_1 x + a_2 y + a_3 - a_7 x'x - a_8 x'y$$

$$y' = a_7 x + a_8 y + a_6 - a_7 y'x - a_8 y'y$$

x = (x,y) **x'** = (x',y')



Two rows for each point *i*

$$\begin{bmatrix} & & & & \vdots & & & \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i' & -y_i x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y_i' & -y_i y_i' \\ & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} = \begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix}$$

$$\mathbf{T}_{8X8}\mathbf{a}_{8X1} = \mathbf{B}_{8X1}$$

This looks like Linear Least squares formulation. Once the solution is obtained re-arrange the solution to form **A**

Homography using Direct Linear Transformation (DLT)

Don't assume $a_9 = 1$, add it to the system

In order to obtain a non-trivial solution,

we obtain **a :**

min || **T'a** || such that ||**a**|| =1

Two rows for each point i

$$\begin{bmatrix} & & & & \vdots & & & \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i' & -y_i x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y_i' & -y_i y_i' \\ & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} = \begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix}$$

a_9

$\mathbf{T'}_{8X9}\mathbf{a}_{9X1} = \mathbf{0}_{9X1}$

$\mathbf{T'} = [\mathbf{T},\mathbf{-B}]$

Homography using Direct Linear Transformation (DLT)

Don't assume $a_9 = 1$, add it to the system

In order to obtain a non-trivial solution,

we obtain **a :**

min || **T'a** || such that ||**a**|| =1

Solution: SVD of **T'**. The eigenvector corresponding to smallest eigenvalue is the solution.

In order to get a stable solution, Normalise the data before DLT.

a) Translate for zero mean
b) b) Scale so that average distance to origin is ~sqrt(2)

https://www.cs.cmu.edu/~16385/s17/Slides/10.2_2D_Alignment__DLT.pdf

Homography using Direct Linear Transformation (DLT)

Don't assume $a_9 = 1$, add it to the system

In order to obtain a non-trivial solution,

we obtain **a** :

min || **T'a** || such that ||**a**|| =1

Solution: SVD of **T'**. The eigenvector corresponding to smallest eigenvalue is the solution.

In order to get a stable solution, Normalise the data before DLT.

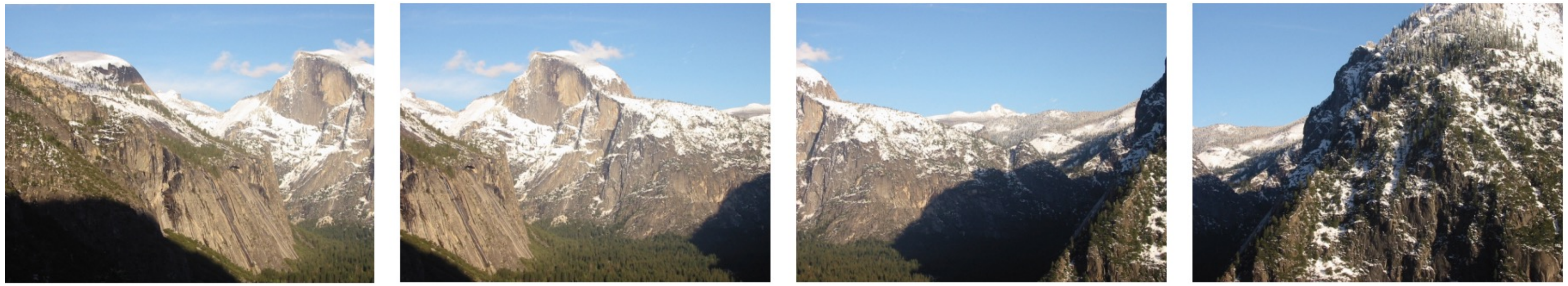
a) Translate for zero mean

b) b) Scale so that average distance to origin is ~sqrt(2)

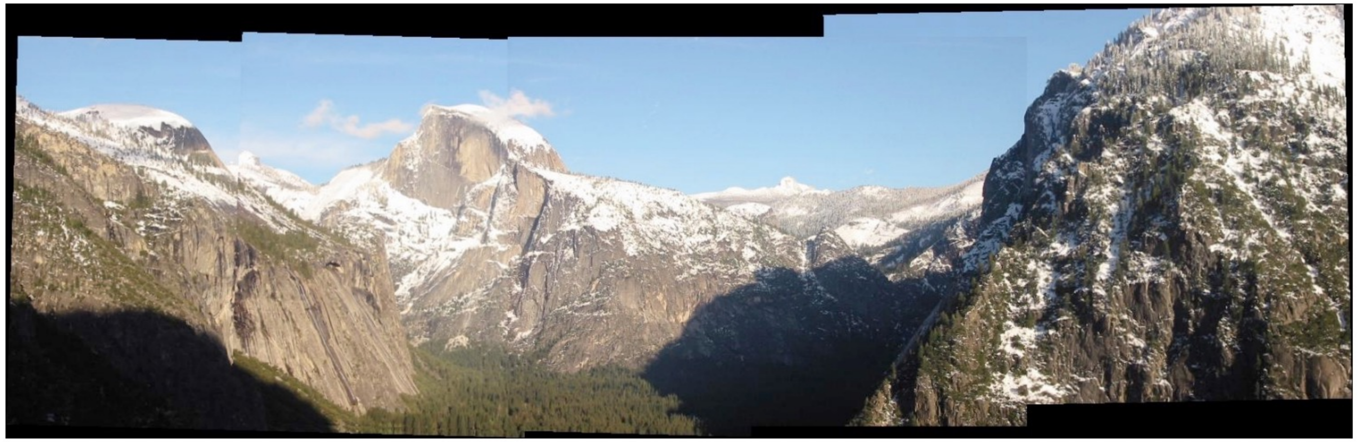
https://www.cs.cmu.edu/~16385/s17/Slides/10.2_2D_Alignment__DLT.pdf

Panoramic Image Stitching

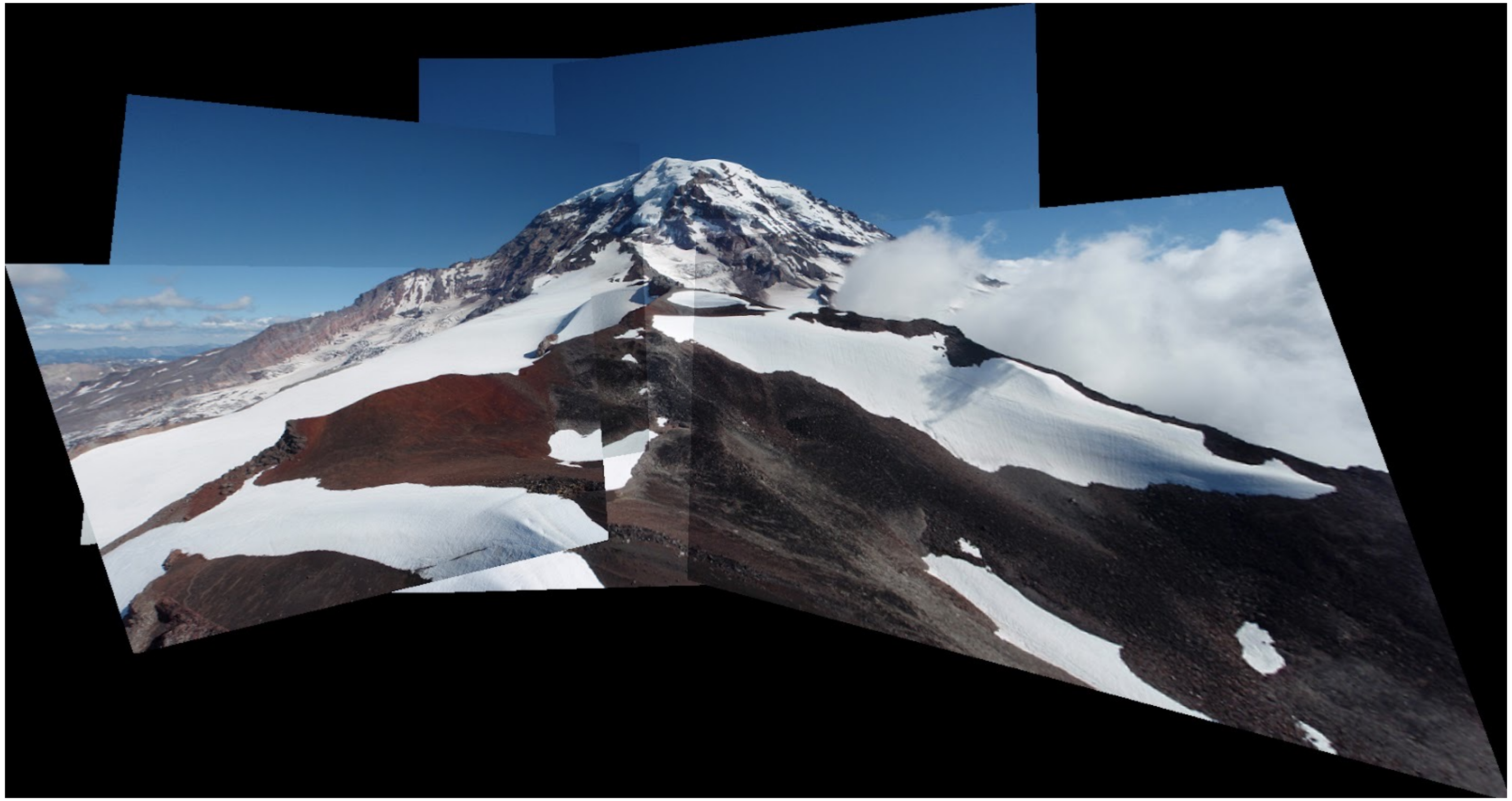
Given a set of images

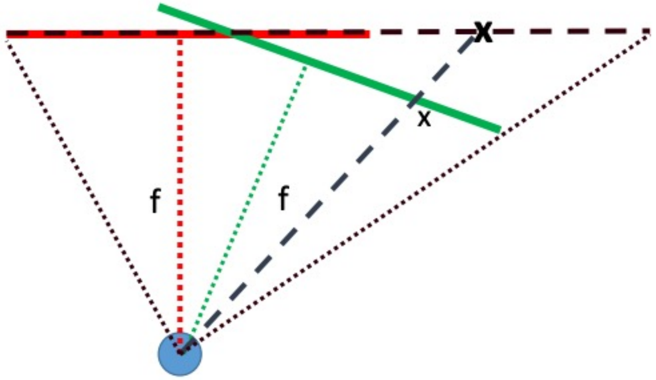


Obtain:



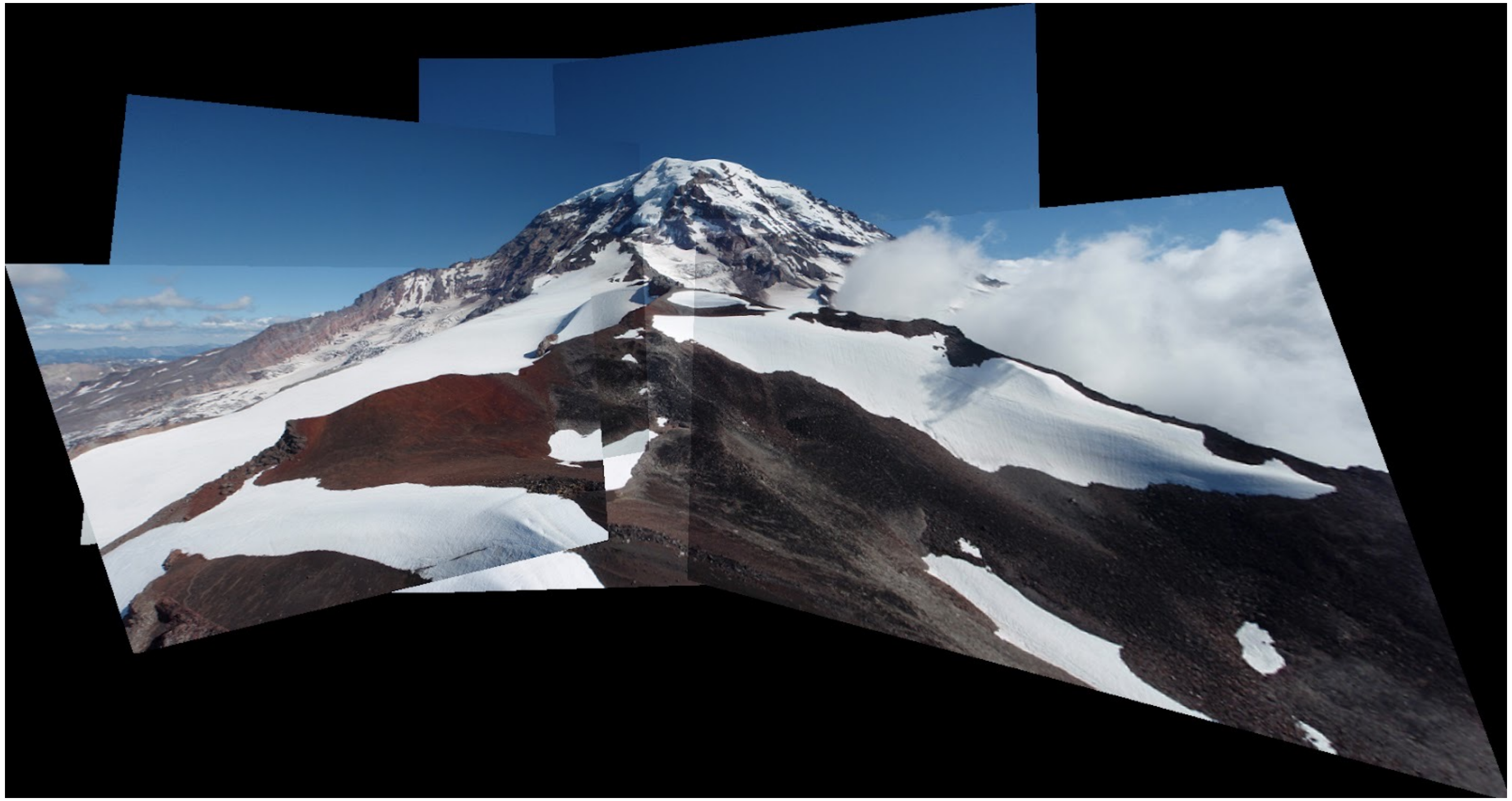
In Practice, we map planes

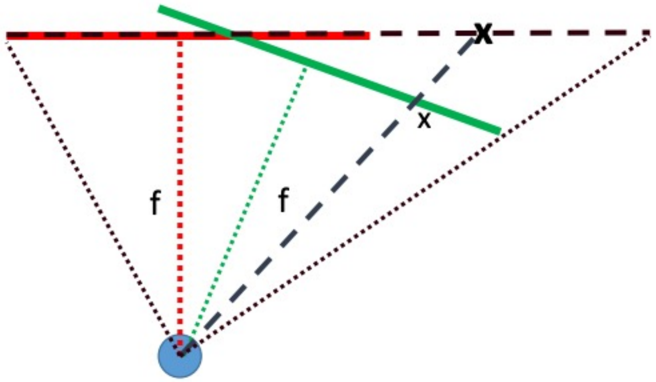




For red image: pixels are already on the plane
For green image: map to first image plane

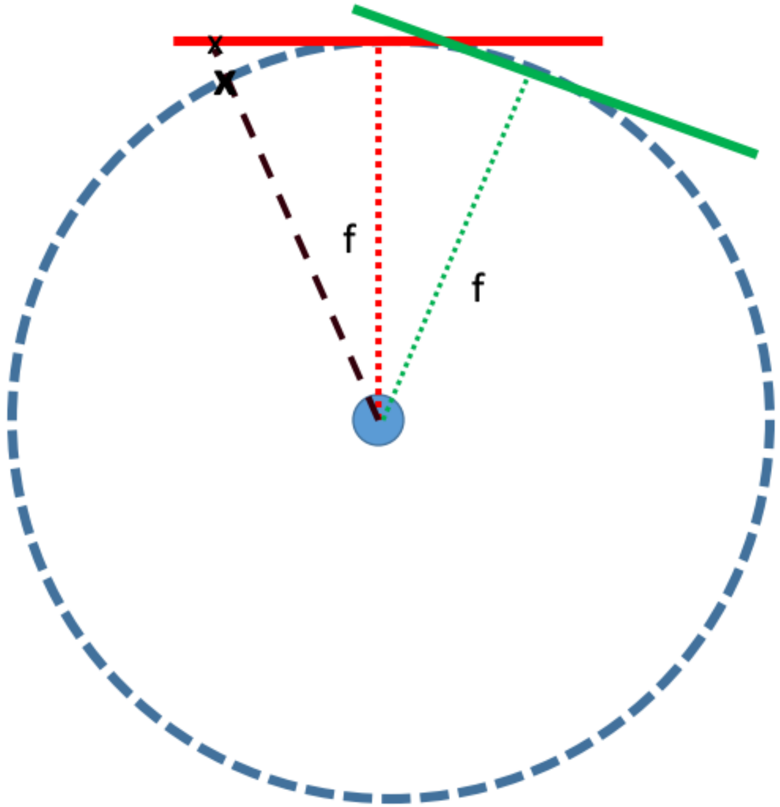
Planar mapping





For red image: pixels are already on the plane
For green image: map to first image plane

Solution: Use cylindrical mapping instead



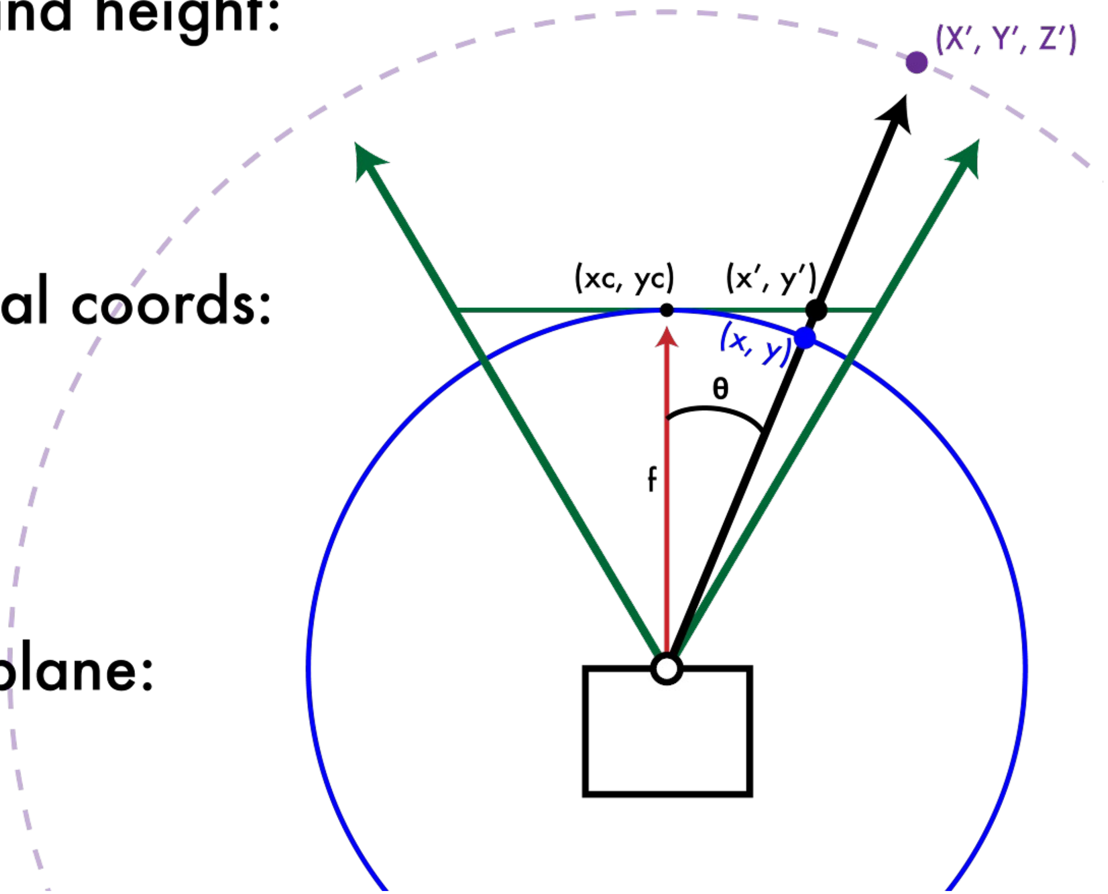
For red image: compute h, theta on cylindrical surface from (u, v)
For green image: map to first image plane, than map to cylindrical surface

Solution: Use cylindrical mapping instead

Calculate angle and height:
$\theta = (x - xc) / f$
$h = (y - yc) / f$

Find unit cylindrical coords:
$X' = \sin(\theta)$
$Y' = h$
$Z' = \cos(\theta)$

Project to image plane:
$x' = f\, X'/Z' + xc$
$y' = f\, Y'/Z' + yc$



RANSAC

Look for ‘only’ inliers.

1. Randomly select a seed group to estimate transformation
2. Compute transformation
3. Compute inliers
4. If inliers are large, recompute transformation

Keep transformation with most inliers

For octave, you can use https://github.com/RANSAC/RANSAC-Toolbox

P1: Image warping

1. Warp simpsons.jpeg to bus.jpeg such that simpsons appear on the bus advertisement

You can manually select the image features for matching

P2: Image stitching

1. Build a panaroma using keble images.
2. Use at least 3 methods to find features.
3. Estimate transformations with and without RANSAC
4. Manually match images and estimate transformation
5. Is any result from 3 is close to 4 ?
6. Repeat the experiment on various images of “La Place des Jacobins”.

Bonus: Can you automatically align images?

Feature Tracking & Optical Flow

Feature tracking: Extract visual features (corners, textured areas) and “track” them over multiple frames

Optical Flow: Recover image motion at each pixel from spatio-temporal image brightness variations

Optical flow = apparent motion of brightness patterns

Optical Flow

Why bother with motion?

G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis", *Perception and Psychophysics 14, 201-211, 1973.*

Feature Tracking & Optical Flow

Challenges:

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

Feature Tracking & Optical Flow

Feature tracking: Extract visual features (corners, textured areas) and “track” them over multiple frames

Problems: Sparse, inexact feature alignment, low accuracy

Optical Flow: Recover image motion at each pixel from spatio-temporal image brightness variations

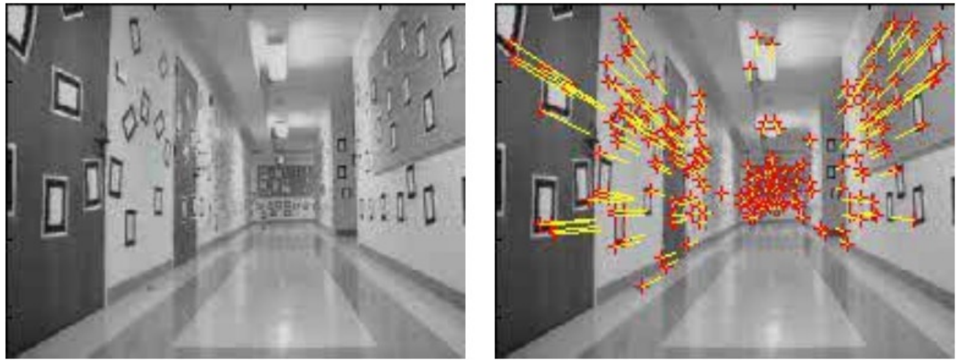
Advantage: Scale/rotation invariant, lighting invariant, can handle large movements (not really, not accurately for sure)

Registration using Lucas-Kanade Method

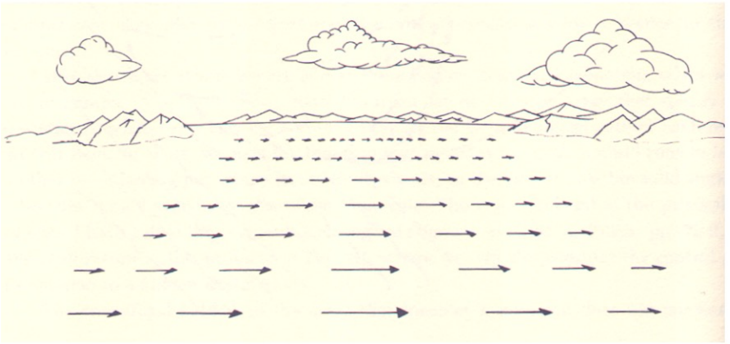
An iterative image registration technique with an application to stereo vision

Feature Tracking & Optical Flow

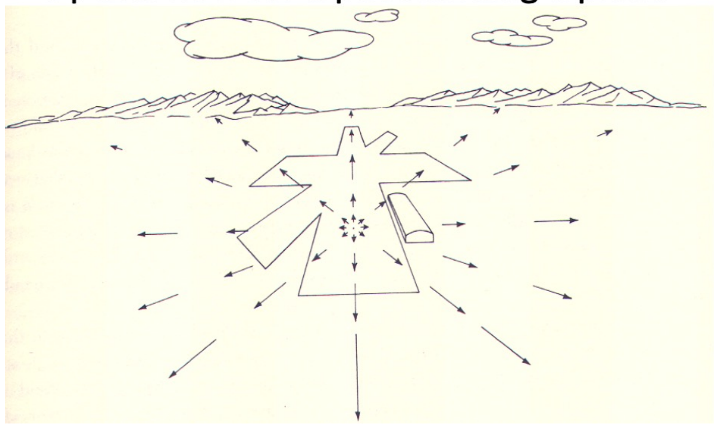
Feature-tracking



Optical flow from the side window of a car

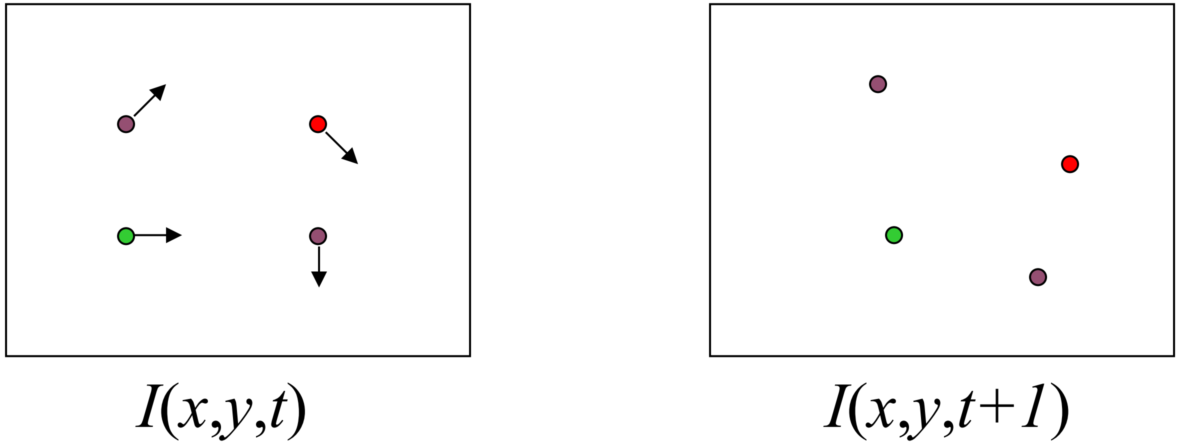


Optical flow for a pilot landing a plane



Applications:
3D reconstruction, Object segmentation, Motion Segmentation, Tracking, Super-resolution, Activity recognition

Optical Flow

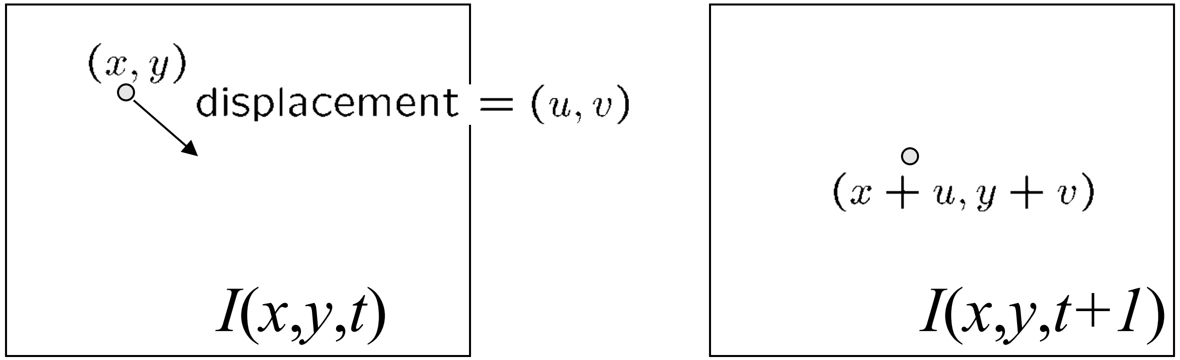


Given two subsequent frames, estimate the point translation

Key assumptions of Lucas-Kanade Tracker

- **Brightness constancy:** projection of the same point looks the same in every frame
- **Small motion:** points do not move very far
- **Spatial coherence:** points move like their neighbors

Optical Flow: Brightness Constancy



Brightness Constancy Equation: $I(x,y,t) = I(x+u, y+v, t+1)$

Take Taylor expansion of $I(x+u,\ y+v,\ t+1)$ at (x,y,t) to linearize the right side:

Image derivative along x Difference over frames

$$I(x+u, y+v, t+1) \approx I(x,y,t) + \boxed{I_x} \cdot u + I_y \cdot v + \boxed{I_t}$$

$$I(x+u, y+v, t+1) - I(x,y,t) = I_x \cdot u + I_y \cdot v + I_t$$

So: $I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \rightarrow \nabla \mathrm{I} \cdot [\mathrm{u}\ \ \mathrm{v}]^{\mathrm{T}} + \mathrm{I}_{\mathrm{t}} = 0$

Optical Flow: Brightness Constancy

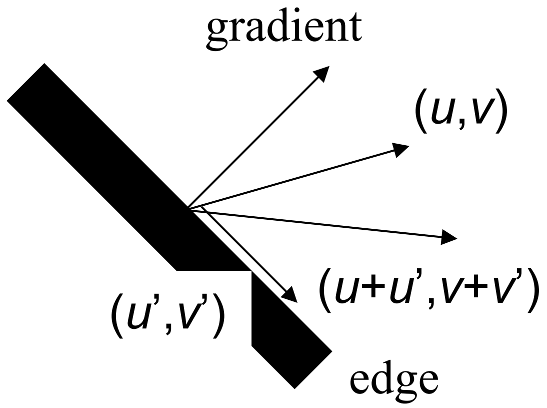
$$\nabla I \cdot [u \;\; v]^T + I_t = 0$$

1 equation, 2 unknowns (u,v), can't solve

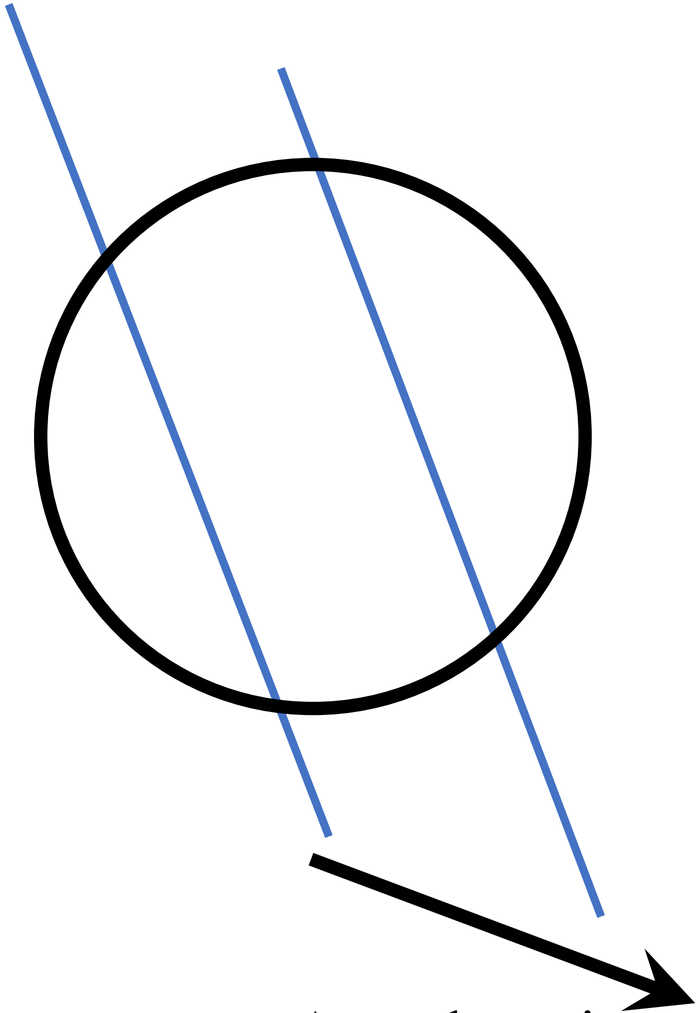
The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If (u, v) satisfies the equation, so does $(u+u', v+v')$ if

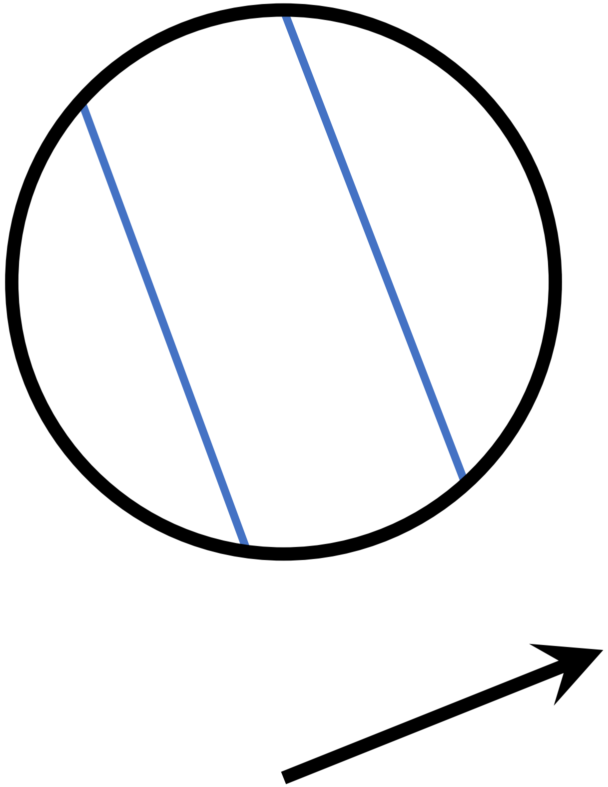
$$\nabla I \cdot [u' \;\; v']^T = 0$$



Optical Flow: Aperture Problem

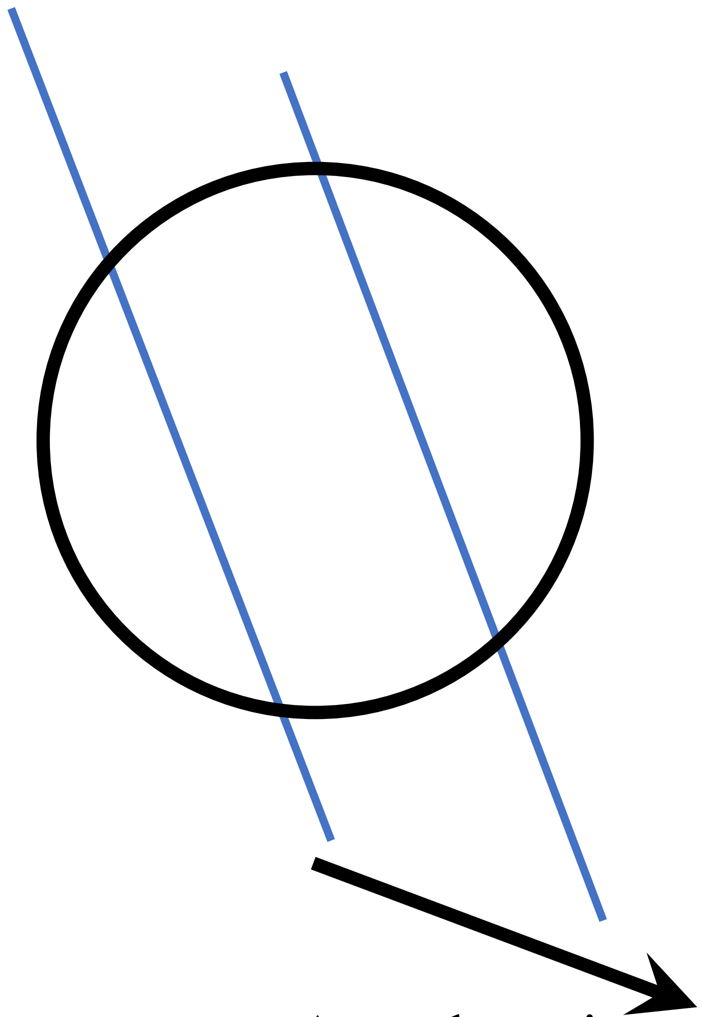


Actual motion

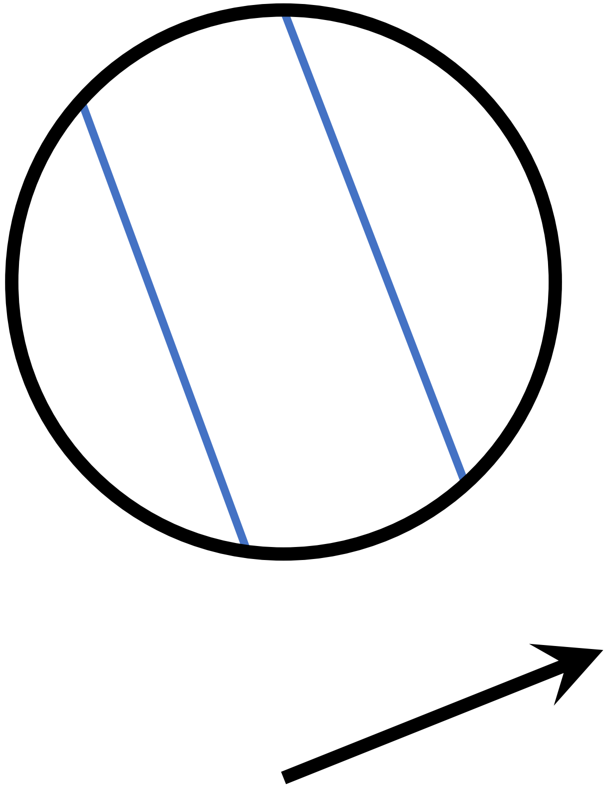


Perceived motion

Optical Flow: Aperture Problem



Actual motion



Perceived motion

Solution:
Look at neighbors

Optical Flow: Barber Pole Illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

Optical Flow: Barber Pole Illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

Optical Flow: Solving the issue

Get more equations for a pixel.

Spatial coherence constraint: Assume the pixel's neighbors have the same (u,v)

If we use a 5x5 window, that gives us 25 equations per pixel

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \;\; v]$$

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}$$

$$\Rightarrow \underset{25\text{x}2}{A} \; \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

$$(A^T A) \; d = A^T b$$

Optical Flow: Solving the issue

When is this solvable? I.e., what are good points to track? Remember: Harris Corners

- $\mathbf{A^TA}$ should be invertible
- $\mathbf{A^TA}$ should not be too small due to noise
  – eigenvalues λ_1 and λ_2 of $\mathbf{A^TA}$ should not be too small
- $\mathbf{A^TA}$ should be well-conditioned
  – λ_1/λ_2 should not be too large (λ_1 = larger eigenvalue)

Low-texture region

– gradients are small magnitude
– small eigenvalues

Edge

– gradients are large and small
– large and small eigenvalue

High-texture region

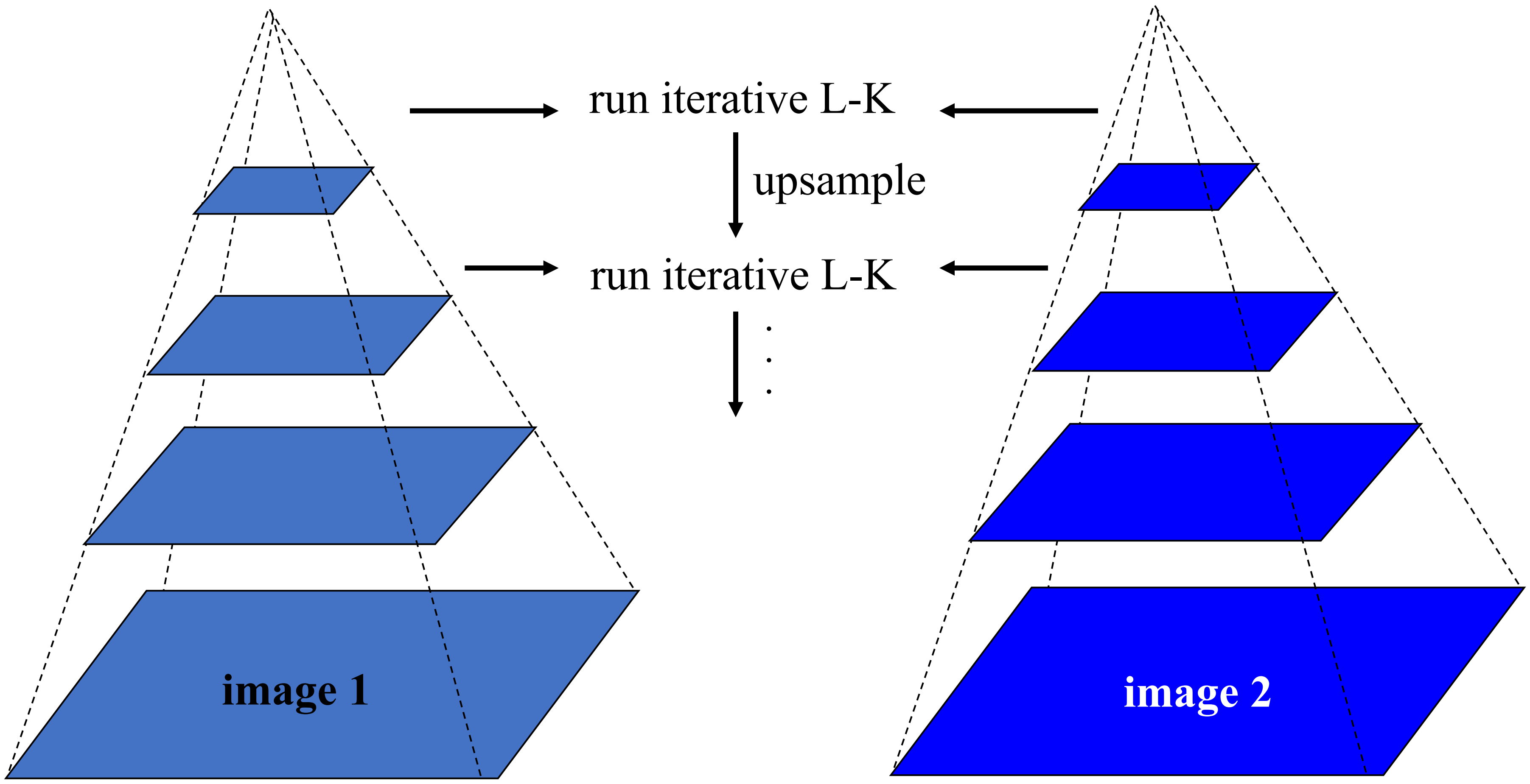
– gradients are different, large magnitudes
– large eigenvalues

Optical Flow- Lucas-Kanade

Assumptions:

1. Brightness consistency
2. Spatial coherence
3. Small motion (not really!)

Dealing with larger movements: coarse-to-fine registration



Gaussian pyramid of image 1 (t)

Gaussian pyramid of image 2 (t+1)

Optical Flow: State of the Art

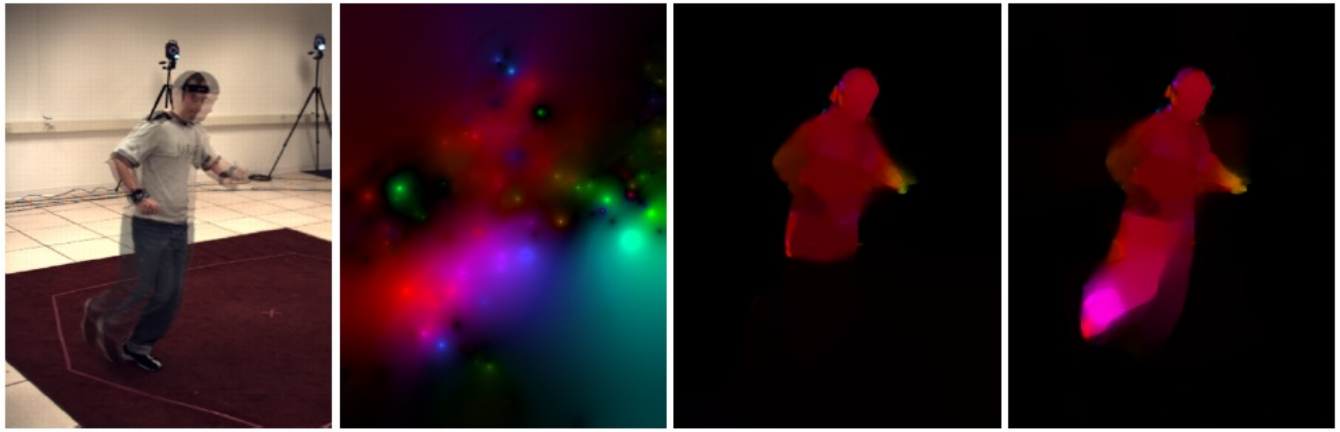
Start with something similar to Lucas-Kanade

+ gradient constancy

+ energy minimization with smoothing term

+ region matching

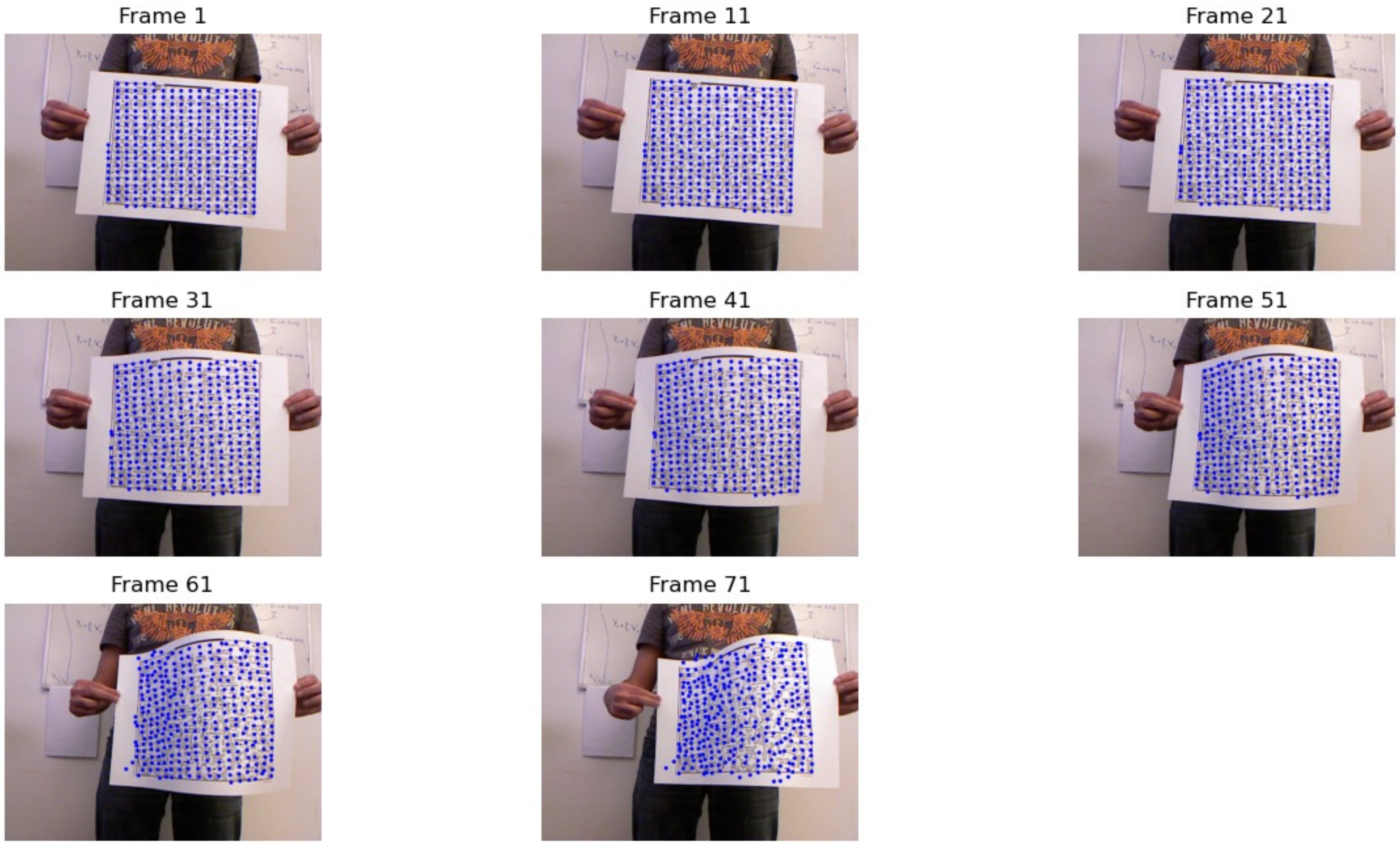
+ keypoint matching (long-range)



Region-based +Pixel-based +Keypoint-based

Large displacement optical flow, Brox et al., CVPR 2009

Optical Flow: State of the Art



PIPS: ECCV-2022

Feature Tracking: Shi-Tomasi Feature Tracker

Find good features using eigenvalues of second-moment matrix (e.g., Harris detector or threshold on the smallest eigenvalue)

- Key idea: “good” features to track are the ones whose motion can be estimated reliably

Track from frame to frame with Lucas-Kanade

- This amounts to assuming a translation model for frame-to-frame feature movement

Check consistency of tracks by *affine* registration to the first observed instance of the feature

- Affine model is more accurate for larger displacements
- Comparing to the first frame helps to minimize drift

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Feature Tracking: Shi-Tomasi Feature Tracker

Figure 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 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Feature Tracking: Shi-Tomasi Feature Tracker

1. Find a good point to track (harris corner)
2. Use intensity second moment matrix and difference across frames to find displacement
3. Iterate and use coarse-to-fine search to deal with larger movements
4. When creating long tracks, check appearance of registered patch against appearance of initial patch to find points that have drifted

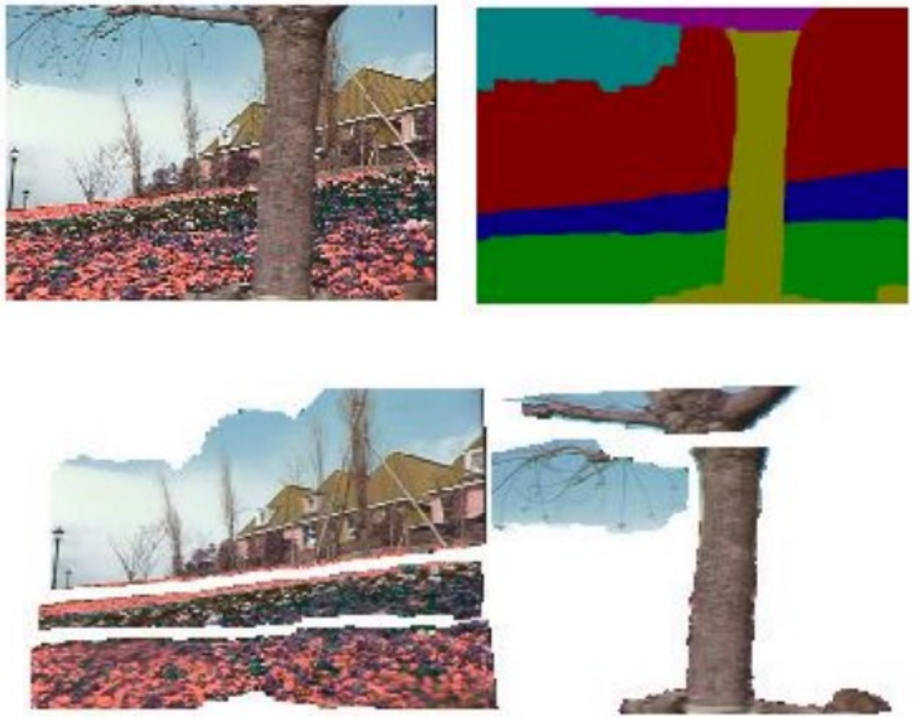
Implementation issues:

- Window size
  - Small window more sensitive to noise and may miss larger motions (without pyramid)
  - Large window more likely to cross an occlusion boundary (and it's slower)
  - 15x15 to 31x31 seems typical
- Weighting the window
  - Common to apply weights so that center matters more (e.g., with Gaussian)

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Motion Segmentation

- Create layers (with coherent affine motion) and track them



J. Wang and E. Adelson. Layered Representation for Motion Analysis. *CVPR 1993*.

Affine Motion

$$u(x,y) = a_1 + a_2 x + a_3 y$$
$$v(x,y) = a_4 + a_5 x + a_6 y$$

- Substituting into the brightness constancy equation:

$$\boxed{I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \approx 0}$$

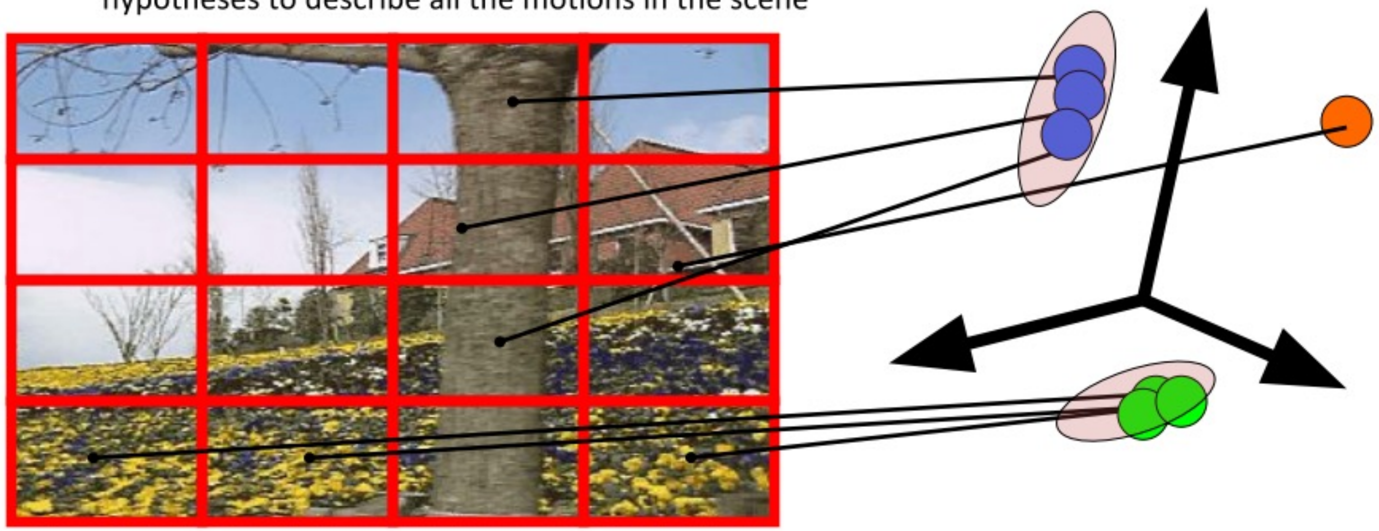
- Each pixel provides 1 linear constraint in 6 unknowns
- Least squares minimization:

$$\boxed{Err(\vec{a}) = \sum \left[I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \right]^2}$$

Source: Silvio Savarese

How to estimate layers?

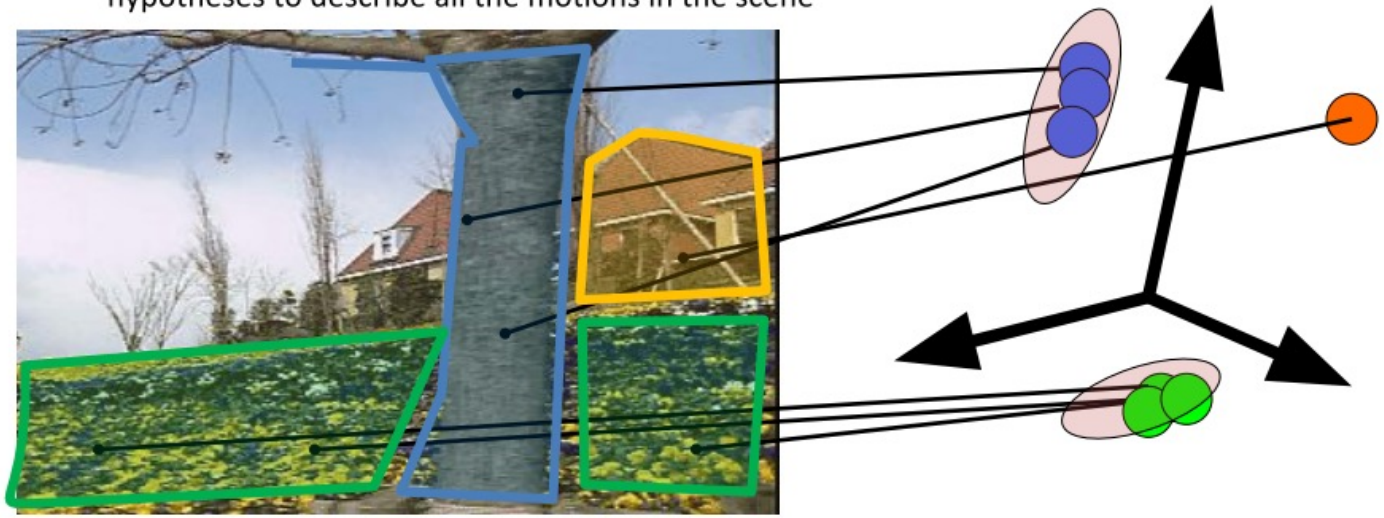
1. Obtain a set of initial affine motion hypotheses
   - Divide the image into blocks and estimate affine motion parameters in each block by least squares
   - Eliminate hypotheses with high residual error
2. Map into motion parameter space
3. Perform k-means clustering on affine motion parameters
   - Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene



Source: Silvio Savarese

How to estimate layers?

1. Obtain a set of initial affine motion hypotheses
   - Divide the image into blocks and estimate affine motion parameters in each block by least squares
   - Eliminate hypotheses with high residual error
2. Map into motion parameter space
3. Perform k-means clustering on affine motion parameters
   - Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene



Source: Silvio Savarese

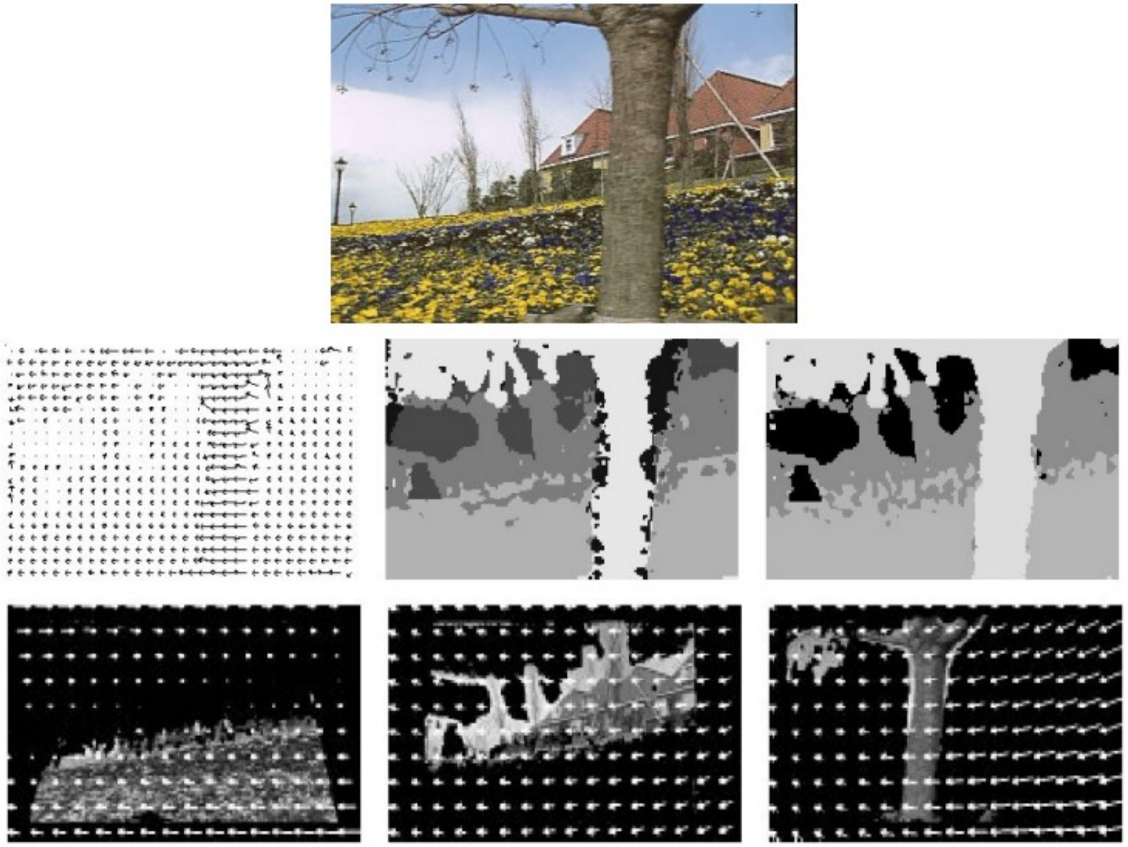
How to estimate layers?

1. Obtain a set of initial affine motion hypotheses
   - Divide the image into blocks and estimate affine motion parameters in each block by least squares
   - Eliminate hypotheses with high residual error
   - Map into motion parameter space
   - Perform k-means clustering on affine motion parameters
   - Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene

2. Iterate until convergence:
   - Assign each pixel to best hypothesis
   - Pixels with high residual error remain unassigned
   - Perform region filtering to enforce spatial constraints
   - Re-estimate affine motions in each region

Source: Silvio Savarese

How to estimate layers?



J. Wang and E. Adelson. Layered Representation for Motion Analysis. *CVPR 1993*.

Source: Silvio Savarese

Course Project: Rubik's cube.

Use a webcam to detect all small squares, obtain cube orientation and reconstruct the cube.

Goals:

1. Detect a cube in the video stream through webcam. Be careful of multiple configurations. (30%)
2. Detect all small squares. The application should be able to count number of how many squares of each color are visible and how many faces are visible. (30%)
3. Detect the cube orientation. Display the information. (15%)
4. Generate 3D model (15%)

To submit: User manual, (OPENCV/MATLAB) code + GUI, well-documented code

Online Demo in first week of January