# Optical flow and keypoint tracking

* Given two subsequent frames of a video, the optical flow field indicates the apparent motion of each pixel
* If we have more than two frames, we can track features from one frame to the next by following the optical flow

# Motion is a powerful perceptual cue

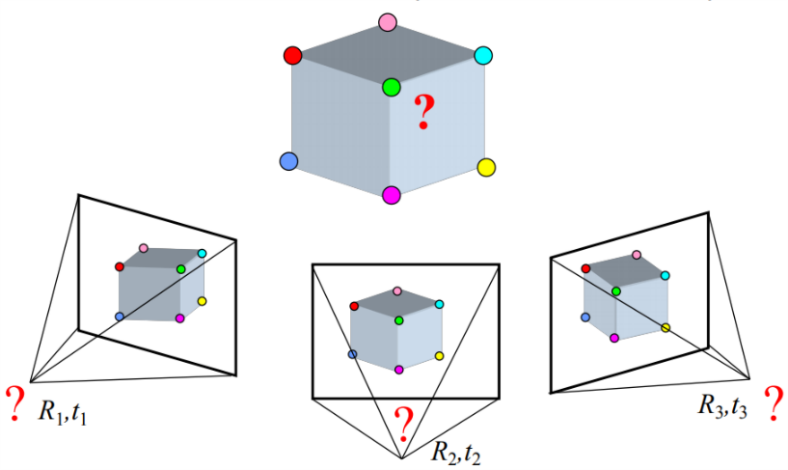
* Sometimes, it is the only cue
* Even “impoverished” motion data can evoke a strong percept

# Uses of motion in computer vision

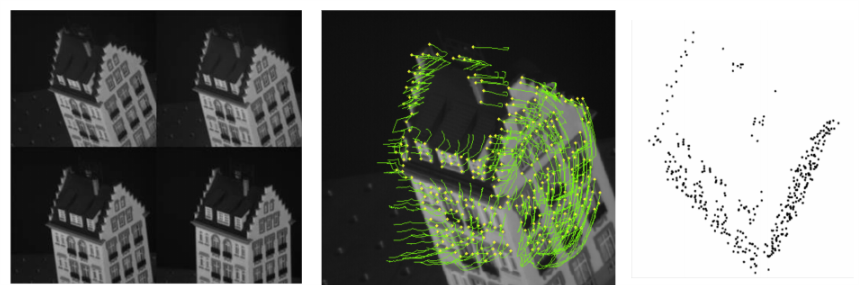
* 3D shape reconstruction
* Object segmentation
* Learning and tracking of dynamic models
* Event and activity recognition

# Preview: Structure from motion

* Given a set of corresponding points in two or more images, compute the camera parameters and the 3D point coordinates

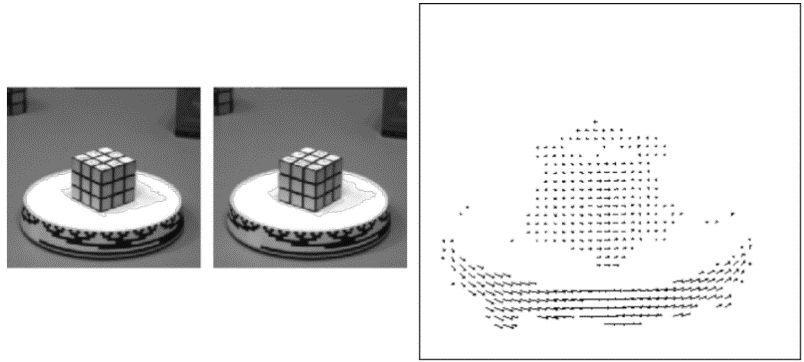


# Keypoint tracking



# Motion field

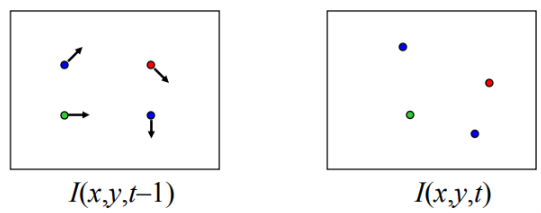
* It is the projection of the 3D scene motion into the image



# Optical flow

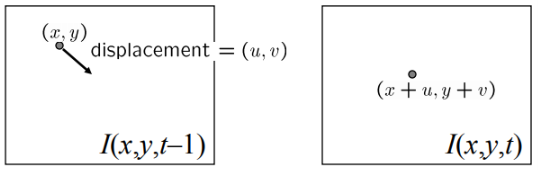
* It is the apparent motion of brightness patterns in the image
* Ideally, optical flow would be the same as the motion field
* Have to be careful: apparent motion can be caused by lighting changes without any actual motion
  + Think of a uniform rotating sphere under fixed lighting vs a stationary sphere under moving illumination

# Estimating optical flow



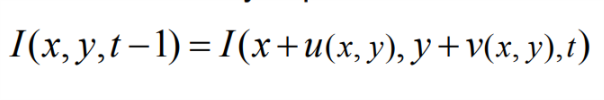
* Given two subsequent frames, estimate the apparent motion field u(x,y) and v(x,y) between them
* Key assumptions
  + Brightness constancy: projection of the same point looks the same in every frame
  + Small motion: points do not move very far
  + Spatial coherence: points move like their neighbors

# The brightness constancy constraint

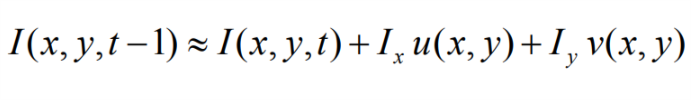


* Brightness constancy equation:

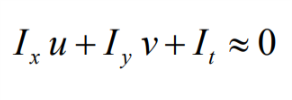
[Ix, Iy are the brightness gradients in their corresponding directions]



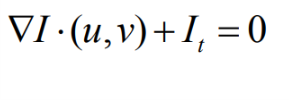
* Linearizing the right side using Taylor expansion:



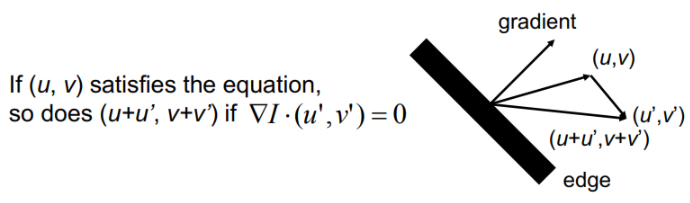
Derivation with respect time…



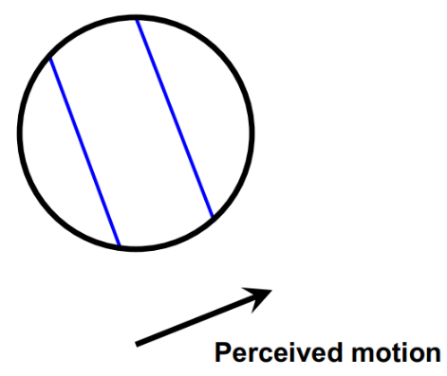
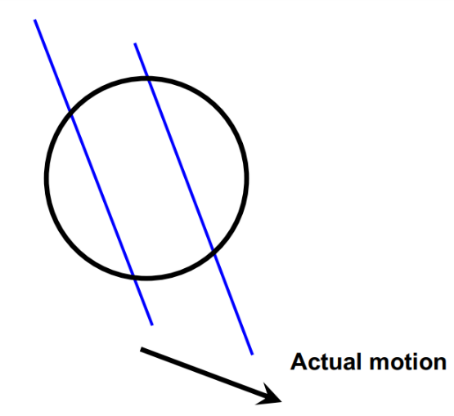
* How many equations do you have per pixel?
  + One equation, two unknowns
* What does this constraint mean?



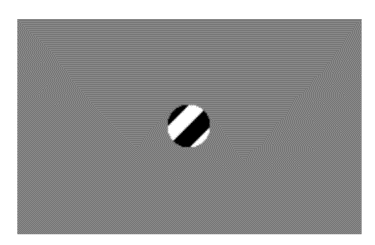
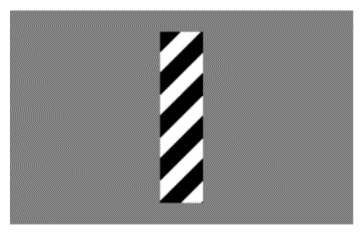
* + The dot product is the value of the projection of the shortest vector to the longest vector
* The component of the flow perpendicular o the gradient (i.e., parallel to the edge) is unknown



# The aperture problem

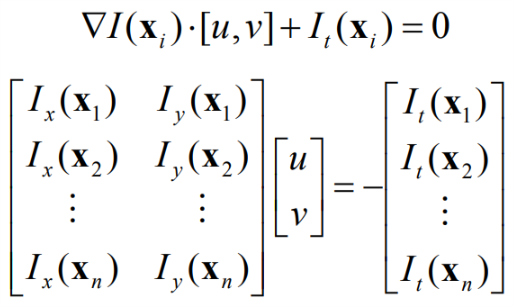
 

# The barber pole illusion

# Solving the aperture problem

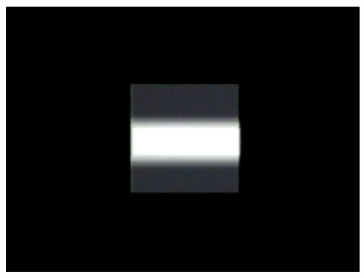
* How to get more equations for a pixel?
* Spatial coherence constraint: pretend the pixel’s neighbor have the same (u,v)
  + E.g., if we use a 5x5 window, that gives us 25 equations per pixel



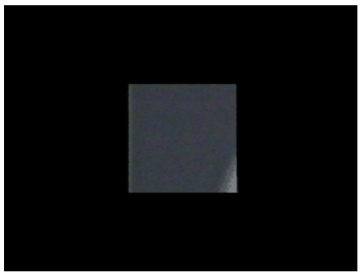
* Least squares problem
* When is it solvable?
  + What I the window contains just a single straight edge?
  + When you have at least two linearly independent equations

# Conditions for solvability

* Bad case: single straight edge

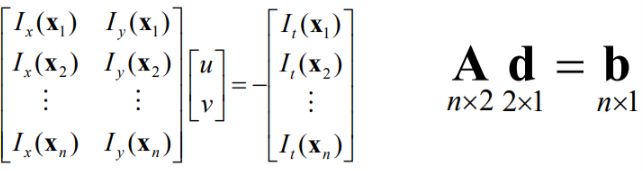


* Good case

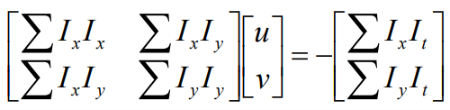


# Lucas-Kanade flow

Linear least squares problem



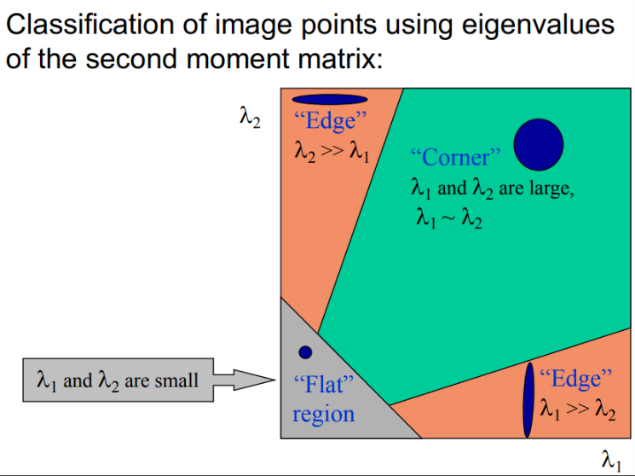
Solution given by (ATA)b = ATb



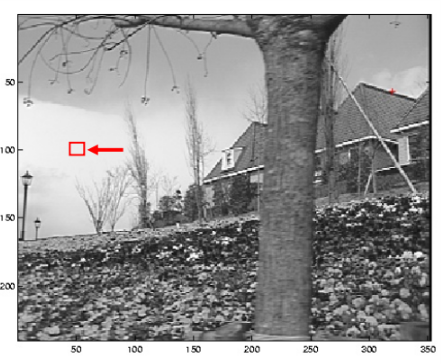
The summations are over all pixels

* Recall the Harris corner detector: M = ATA is second moment matrix
* We can figure out whether the system is solvable by looking at the eigenvalues of the second moment matrix
  + The eigenvectors and eigenvalues of M relate to edge direction and magnitude
  + The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change, and the other eigenvector is orthogonal to it

# Recall: second moment matrix

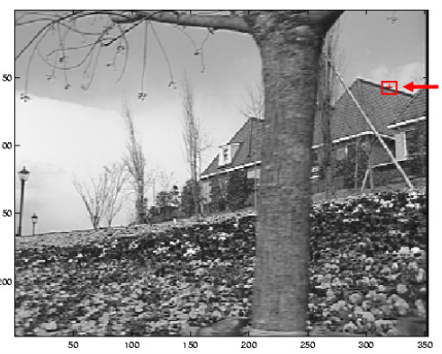


# Uniform region



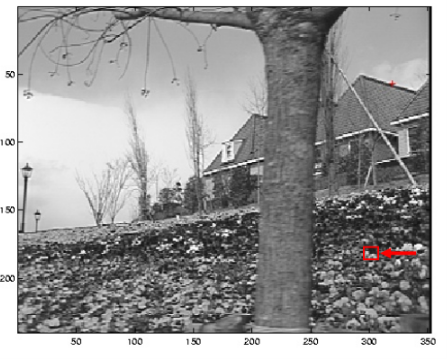
* Gradients have small magnitude
* Small lambda1 and lambda2
* System is ill-conditioned

# Edge



* Gradients have one dominant direction
* Large lambda1, small lambda2
* System is ill-conditioned

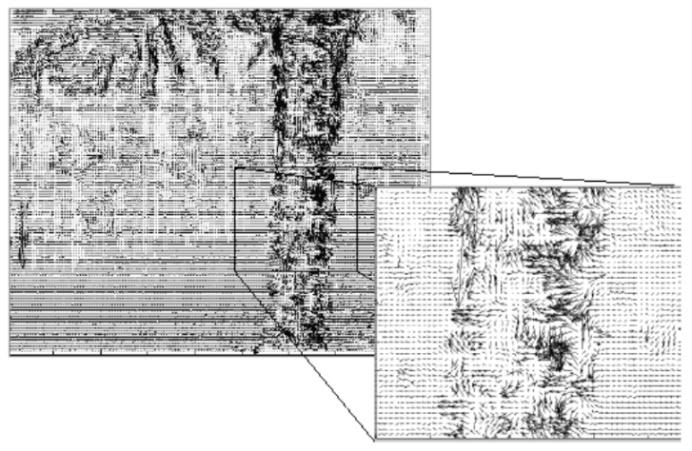
# High-texture or corner region



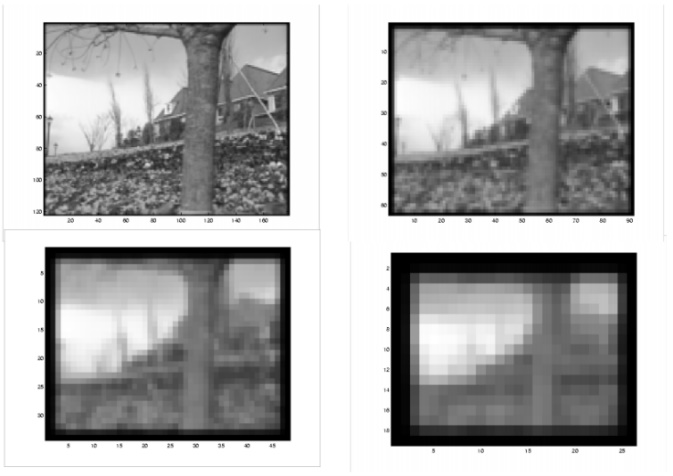
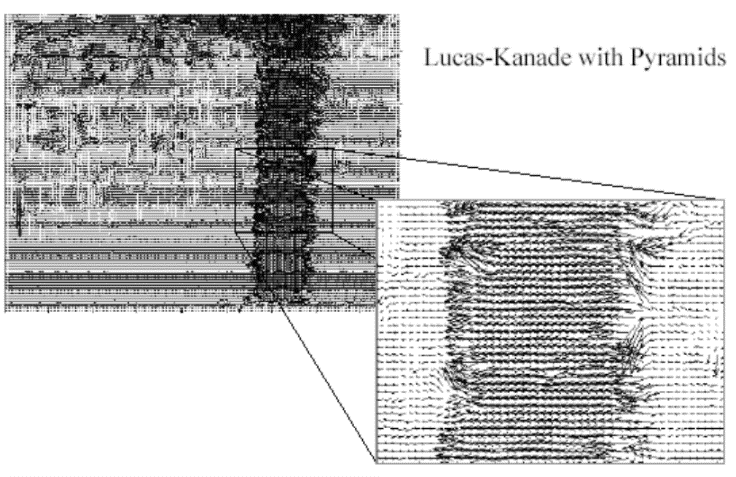
* Gradients have different directions, large magnitudes
* Large lambda1, large lambda2
* System is well-conditioned

# Example of optical flow estimation

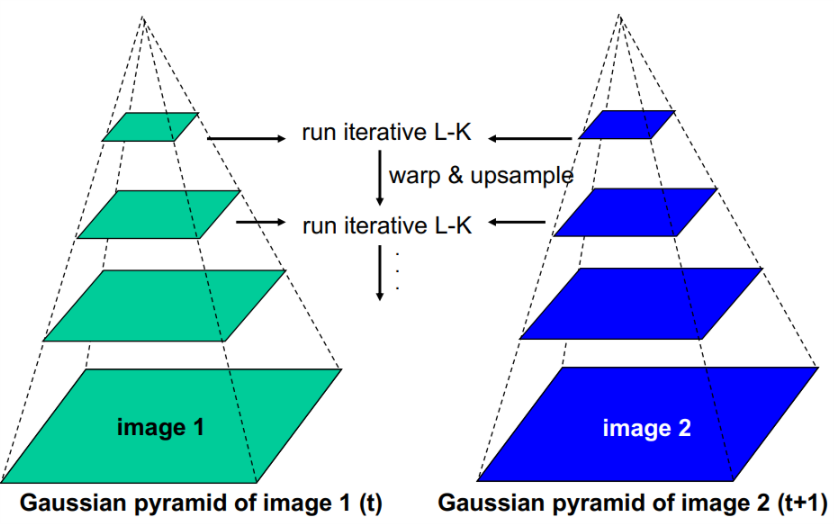
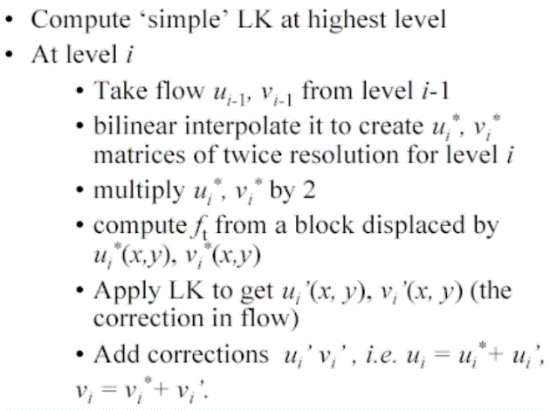
From the previous “sequence of images”



# Multi-resolution estimation

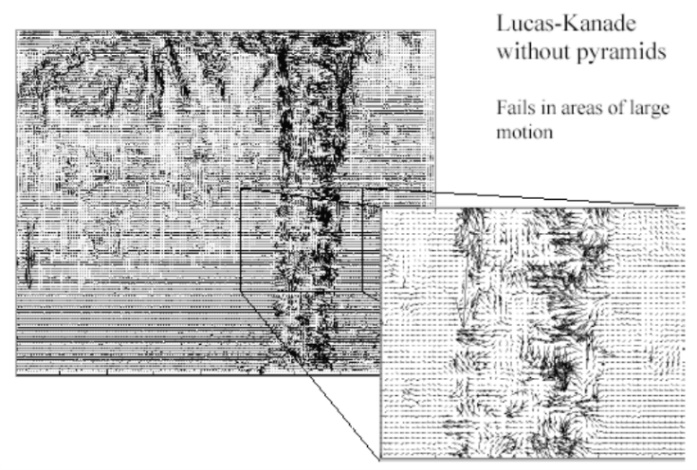
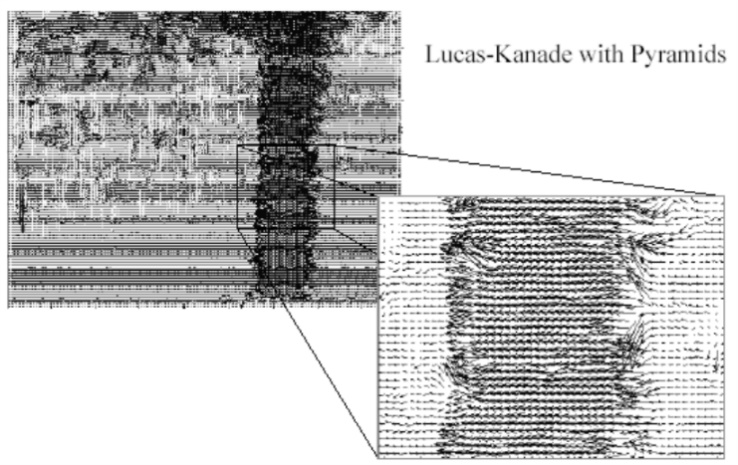
# Coarse-to-fine optical flow estimation

# Iterative Refinement

1. Iterative Lukas-Kanade algorithm
   1. Estimate displacement at each pixel by solving Lucas-Kanade equations
2. Warp I(t) towards I(t+1) using the estimated flow field
   1. Basically, just interpolation
3. Repeat until convergence

# Optical flow results

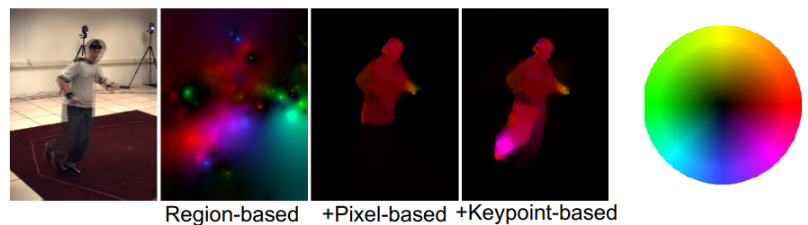
# Errors in Lucas-Kanade

* The motion is large (larger than a pixel)
  + Coarse-to-fine estimation
  + Iterative refinement
  + Exhaustive neighborhood search (feature matching)
* A point does not move like its neighbors
  + Motion segmentation
* Brightness constancy does not hold

# Large displacement optical flow

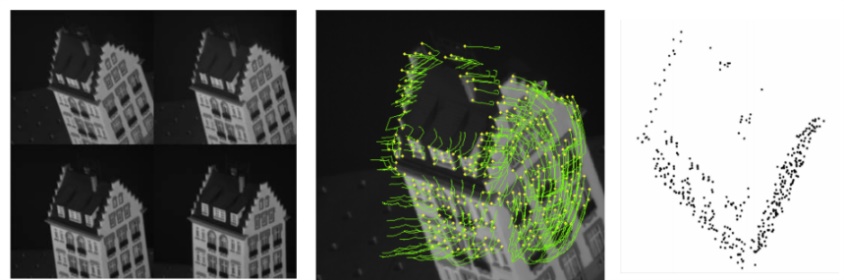
Start with something similar to Lucas-Kanade

* Gradient constancy
* Energy minimization with smoothing term
* Region matching
* Keypoint matching (long-range)



# Feature tracking

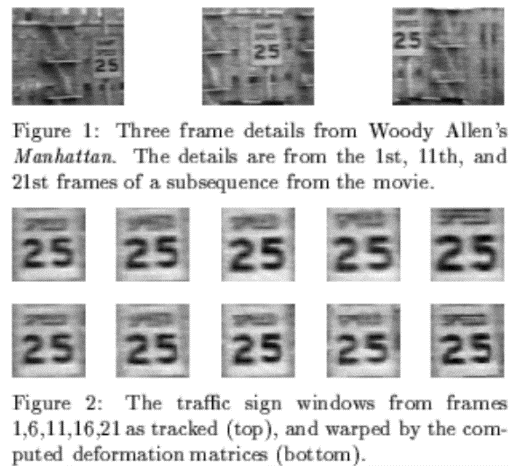
* If we have more than two images, we can track a feature from one frame to the next by following the optical flow
* Challenges
  + Finding good features to track
  + Adding and deleting tracks



# Shi-Tomasi feature tracker

* Find good features using eigenvalues of second-moment matrix
  + Key idea: good features to track are the ones whose motion can be estimated reliably
* From frame to frame, track with Lucas-Kanade
  + This amounts to assuming a translation model for frame-to-frame feature environment
* 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

# Tracking example



# Summary of KLT tracking

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

# Implementation issues

* Windows 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 windows
  + Common to apply weights so that center matters more (e.g., with Gaussian)

# Why not just do local template matching?

* Slow (need to check more locations)
* Does not give subpixel alignment (or becomes much slower)
  + Even pixel alignment may not be good enough to prevent drift
* May be useful as a step-in tracking if there are large movements

# Summary

* Major contributions from Lucas, Kanade, Shi, Tomasi
  + Tracking feature points
  + Optical flow
* Key ideas
  + By assuming brightness constancy, truncated Taylor expansion leads to simple and fast patch matching across frames
  + Coarse-to-fine registration