INF 5300 - 09.04.2014
Dense motion and flow

Anne Schistad Solberg

- Motion perception
- Motion visualization
- Image similarity measures
- Motion estimation
- Optical flow algorithm
- Slide credits: Several slides adapted from R. Szeliski CSE 576.

Curriculum

- Chapter 8 in Szeliski (except 8.3)
- Additional reading:
  - Good description of optical flow: http://www.cs.utoronto.ca/~jepson/csc420/notes/flowChapter05.pdf
  - Simon Baker and Iain Matthews, Lucas-Kanade 20 Years On: A Unifying Framework, International Journal of Computer Vision 56(3), 221-255, 2004 http://dx.doi.org/10.1023/B:VISI.0000011205.11775.fd
  - Optical flow (wikipedia)
  - Horn-Schunck method (wikipedia)

From last lecture: Image matching

- Last weeks:
  - Extract keypoint features in an image
  - Find the matching features in a different image
  - Do a robust motion (e.g. using RANSAC) to get the geometrical model describing a COMMON transform relating the keypoints in both images.
- Characteristics:
  - Keypoint locations are SPARSE
  - A common motion model is assumed for the entire scene.

This lecture: dense motion

- Motion vectors are now estimated from every point an a image sequence.
- Motion maps are created, and each pixel can have a different motion vector.
- Some regularization of the motion vectors is done to get smooth estimates.
  - No restriction that all pixels move in the same average direction.
- Video normally has high frame rate:
  - Small motion between one fram and the next frame

Why estimate visual motion?

- Visual motion can be annoying
  - Camera instabilities: measure it and remove it
- Visual motion indicates dynamics in the scene
  - Moving objects, behaviour in surveillance cameras
  - Track objects and analyse trajectories
- Visual motion reveals spatial layout
  - Motion parallax

Essential steps in motion estimation

- An error metric to compare the two images must be chosen.
- A search technique to compute the best match is needed.
  - Pyramid search is often used to speed up the process.
- Accurate motion estimates might need subpixel accuracy.

- Regularization is often applied since the motion vectors are not reliable in all regions.
  - For compex motion layered motion models might also be needed.

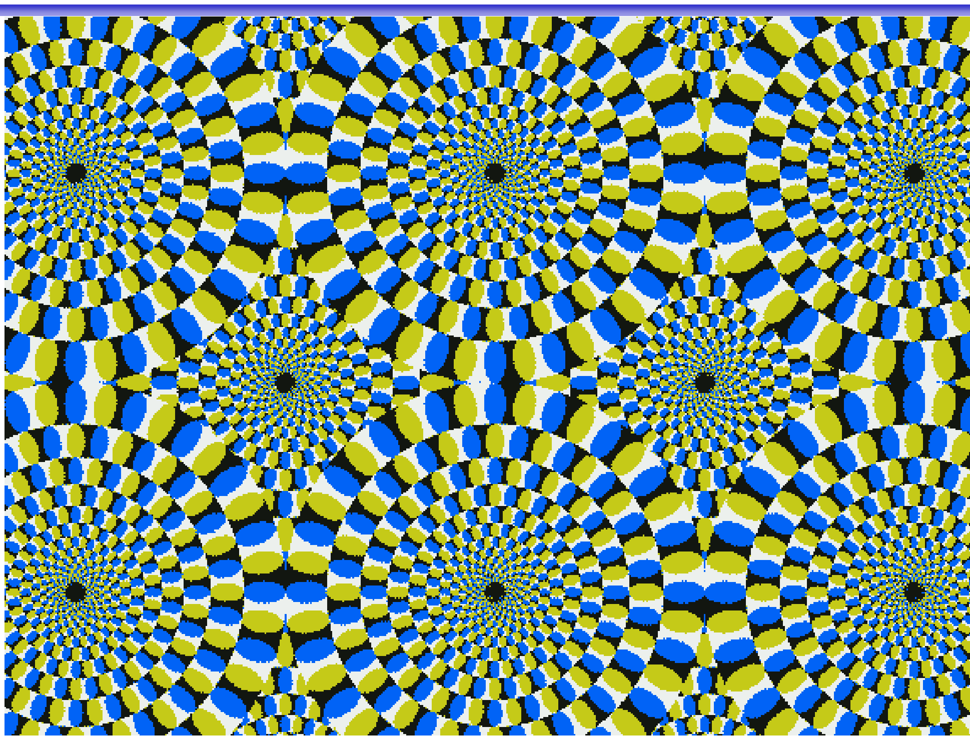
Applications of motion estimation

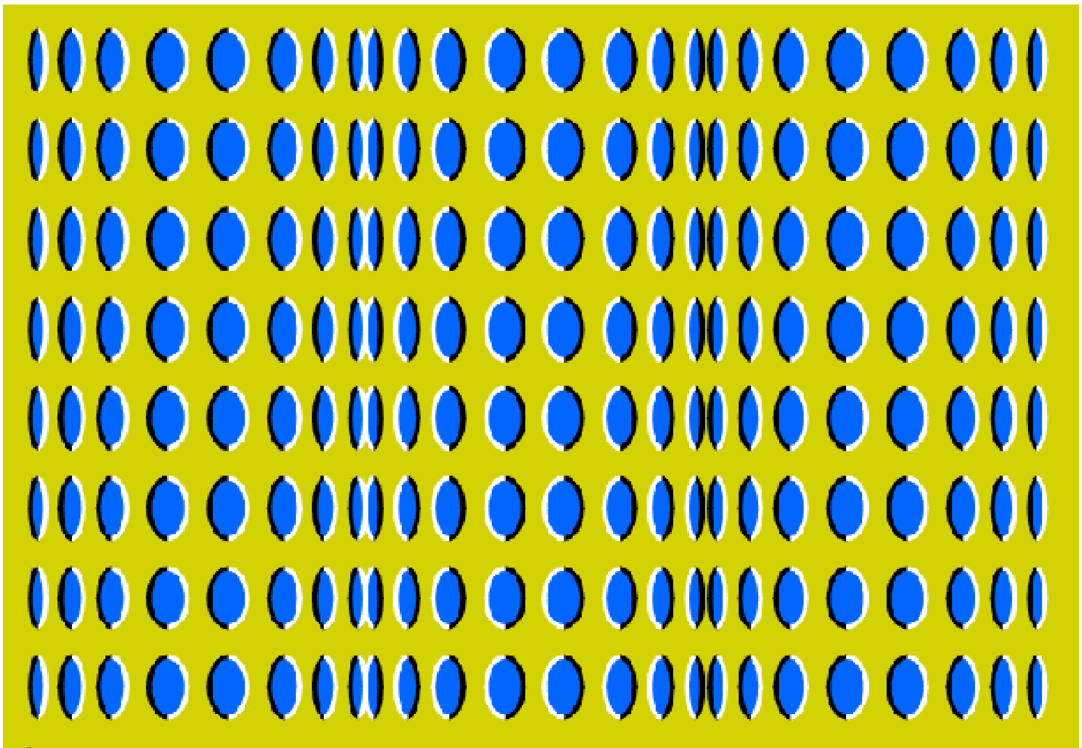
- Video enhancements:
  - Stabilization
  - Denoising
  - Super resolution
- 3D reconstruction: structure from motion
- Video segmentation
- Tracking/recognizing objects
- Learning dynamical models
- Advanced video editing

Motion estimation techniques

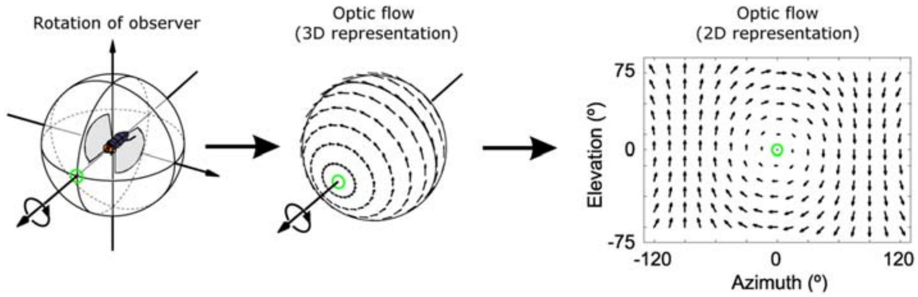
- Direct methods
  - Directly recover image motion at each pixel from spatio-temporal image brightness variations
  - Dense motion fields, but sensitive to appearance variations
  - Suitable for video when image motion is small
  - Computationally expensive
- Feature-based methods
  - Extract visual features (corners, textured areas) and track them over multiple frames
  - Sparse motion fields, but more robust tracking
  - Suitable when image motion is large (10s of pixels)

Seeing motion from a static picture?





Optical flow field



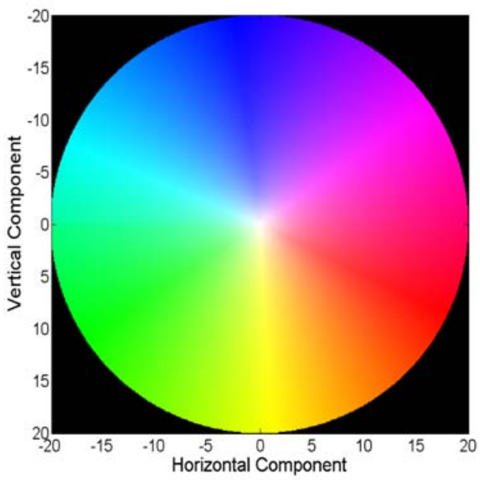
- Optical flow is the apparent motion of objects in a scene caused by the relative motion between an observer (eye or camera) and the scene.
- Parametric motion (e.g. using global geometric transforms) is limited and cannot describe the motion of arbitrary videos.
- Optical flow field: assign a flow vector (u(x,y),v(x,y)) to each pixel (x,y).
- Projection from 3D world to 2D

Visualization of optical flow fields

- Vector fields can be used to visualize sparse motion fields, but are too mess to plot for every pixel.
- Map flow vector to color:
  - – Magnitude: saturation
  - – Orientation: hue

http://hci.iwr.uni-heidelberg.de/Static/correspondenceVisualization/

Image example:
http://people.csail.mit.edu/celiu/OpticalFlow/



Matching brightness patterns

- Brightness constancy assumption:

$$I_L(x,y) = I_R(x+u, y+v) + r + g$$

$$r \sim N(0,\sigma^2), g \sim U(-1,1)$$

Noise r, outlier g (occlusion, lighting change)

- How do we determine correspondences?
  - *block matching* or *SSD* (sum squared differences)

$$E(x,y;d) = \sum_{(x',y') \in N(x,y)} [I_L(x'+d,y') - I_R(x',y')]^2$$

Matching criteria

- What is invariant between the two images?
  - Brightness? Gradients? Phase? Other features?
- Distance metric: (L2,L1, truncated L1, Lorentzian)

$$E(u,v) = \sum_{x,y} \rho\left(I_1(x,y) - I_2(x+u,y+v)\right)$$

- Correlation, normalized cross correlation

Distance metrics

Example: data samples

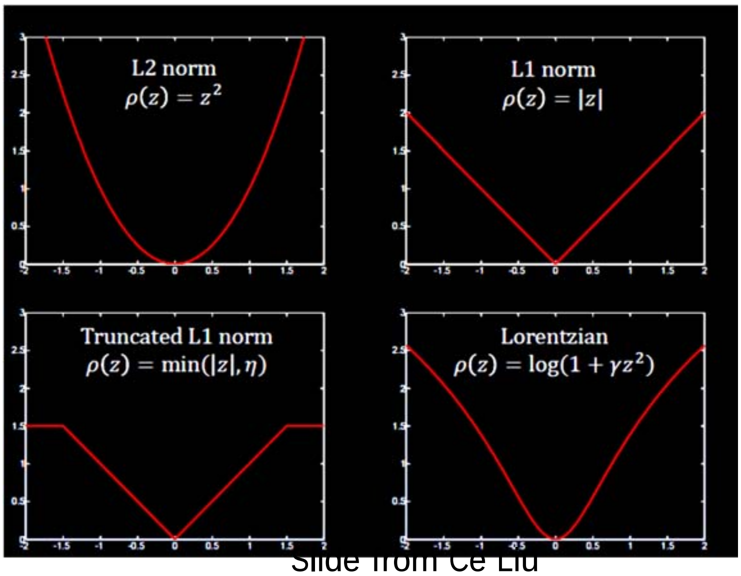
0.95, 1.04, 0.91, 1.02, 1.10, 20.01

L2 norm: error = 4.172

L1 norm: error = 1.038

Truncated L1: error 1.0296

Lorentzian: error 1.0147



Slide from Ce Liu

Alternative error measures

- Spatially varying weights

$$E_{\mathrm{WSSD}}(\boldsymbol{u}) = \sum_i w_0(\boldsymbol{x}_i) w_1(\boldsymbol{x}_i + \boldsymbol{u})[I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2,$$

- Normalized cross-correlation

$$E_{\mathrm{NCC}}(\boldsymbol{u}) = \frac{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]\,[I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \overline{I_1}]}{\sqrt{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]^2}\sqrt{\sum_i [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \overline{I_1}]^2}},$$

$$\overline{I_0} = \frac{1}{N}\sum_i I_0(\boldsymbol{x}_i) \quad \text{and}$$

$$\overline{I_1} = \frac{1}{N}\sum_i I_1(\boldsymbol{x}_i + \boldsymbol{u})$$

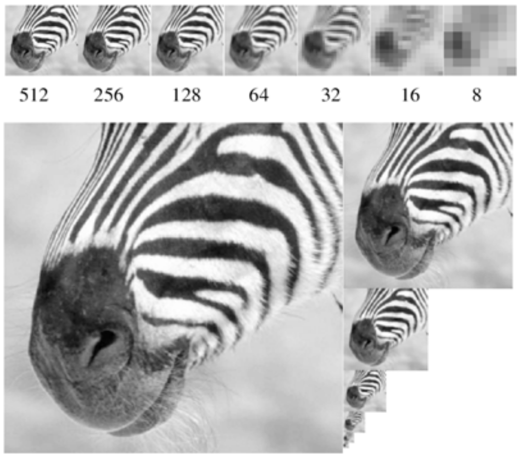
- Normalized SSD score:

$$E_{\mathrm{NSSD}}(\boldsymbol{u}) = \frac{1}{2}\frac{\sum_i \left[[I_0(\boldsymbol{x}_i) - \overline{I_0}] - [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \overline{I_1}]\right]^2}{\sqrt{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]^2 + [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \overline{I_1}]^2}}$$

Hierarchical motion estimation

- Given the cost function, how do we search for the best match?
- Image pyramids (Gaussian) are often used to speed up the process:
  – Search first at a coarse level
  – Refine the fit by a smaller local search at the next finer level.
- Let $I_k^l(x_j)$ be the downsampled and smoothed image at level l, created from the finer level image $I_k^{l-1}(2x_j)$ (see section 3.5 on pyramids)
- Once a suitable motion vector is found at level l, predict the displacement at the next level:

$$\hat{\mathbf{u}}^{(l-1)} \leftarrow 2\hat{\mathbf{u}}^{(l)}$$



Fourier-based alignment

- When the motion is large, searching in a coarser level at the pyramid might not be sufficient.
- Matching in the Fourier domain is an alternative:
  – Correlation can be cone by multiplication by the complex conjugate in the Fourier domain (remember the convolution theorem?)
  – Windowed correlation is often used in addition to this.
  – The Fourier transform after a translation has the same magnitude, but different phase.
- The SSD criterion can also be computed efficiently in the Fourier domain.

The Brightness Constraint

- Brightness Constancy Equation/Find similar patches in two images: $$J(x,y) \approx \boxed{I(x+u(x,y), y+v(x,y))}$$

Or, equivalently, minimize :

$$E(u,v) = (J(x,y) - I(x+u, y+v))^2$$

Linearizing (assuming small *(u,v)*) using Taylor series expansion:

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

I_x and I_y are the horisontal and vertical image gradients

Gradient Constraint (or the Optical Flow Constraint)

$$E(u,v) = (I_x \cdot u + I_y \cdot v + I_t)^2$$

I_t is the temporal gradient

Minimizing: $$\frac{\partial E}{du} = \frac{\partial E}{dv} = 0$$

$$I_x(I_x u + I_y v + I_t) = 0$$

$$I_y(I_x u + I_y v + I_t) = 0$$

In general $I_x, I_y \neq 0$

Hence, $I_x \cdot u + I_y \cdot v + I_t \approx 0$

Least-square problem, see Appendix A.2 for details

Patch Translation [Lucas-Kanade]

Assume a single velocity for all pixels within an image patch

$$E(u,v) = \sum_{x,y\in\Omega} \left(I_x(x,y)u + I_y(x,y)v + I_t\right)^2$$

Minimizing

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}$$

Balance spatial gradients by temporal gradients and the shift in u

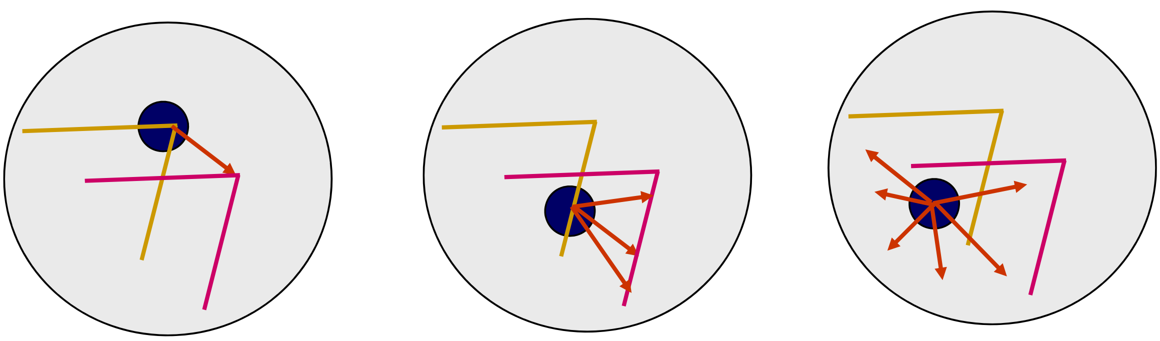
$$\left(\sum \nabla I \nabla I^T\right)\vec{U} = -\sum \nabla I I_t$$

LHS: sum of the 2x2 outer product of the gradient vector

Iterative solutions needed

Local Patch Analysis

- How *certain* are the motion estimates?
- This is similar to finding good keypoints in SIFT.



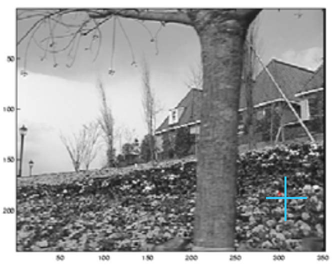
The Aperture Problem

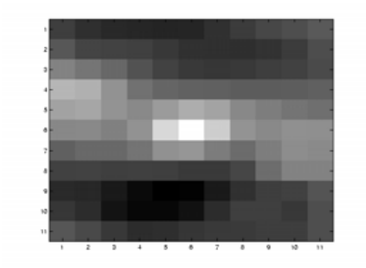
Let $M=\sum(\nabla I)(\nabla I)^T$ and $b=\begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$

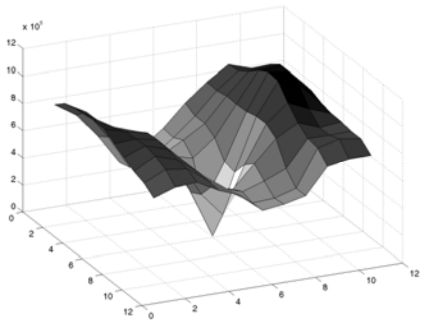
- Algorithm: At each pixel compute U by solving $MU=b$
- *M* is singular if all gradient vectors point in the same direction
  - e.g., along an edge
  - of course, trivially singular if the summation is over a single pixel or there is no texture
  - i.e., only *normal flow* is available (aperture problem)
- Corners and textured areas are OK

SSD Surface – Textured area

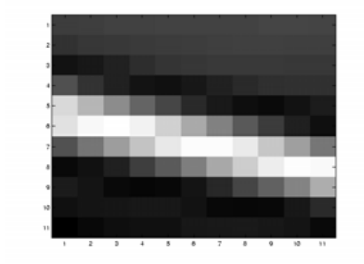
Have you seen this before?
Remember lecture
on keypoint detection

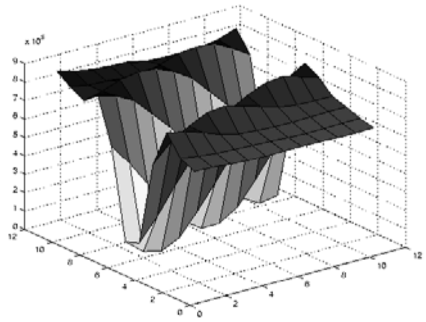




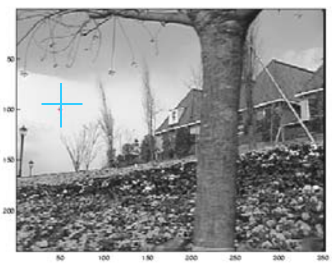


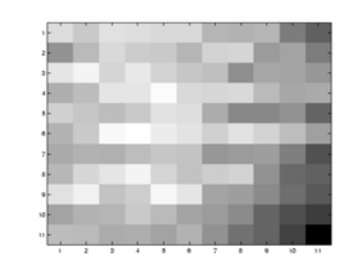
SSD Surface -- Edge

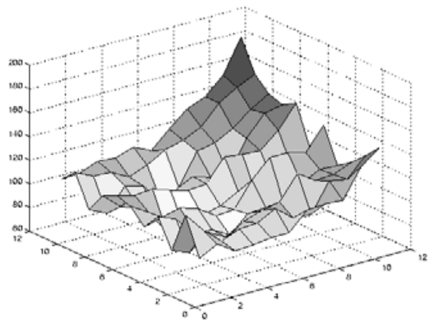




SSD – homogeneous area



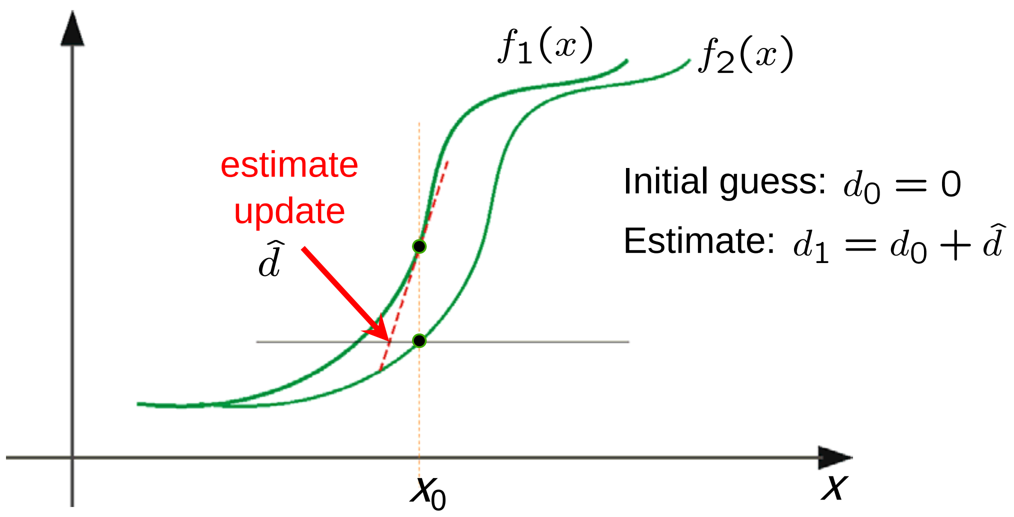




Refining the search to sub-pixel accuracy

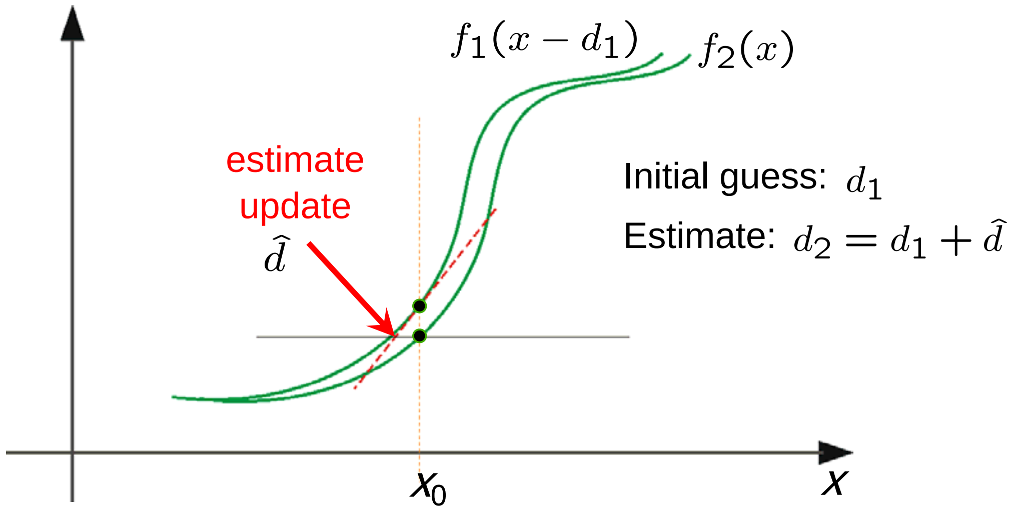
- Estimate velocity at each pixel using one iteration of Lucas and Kanade estimation.
- Many applications, like image stabilization and stitching, require sub-pixel accuracy in matching.
- Refine this estimate by repeating the process
- Remember that the Taylor series expansion ignored the higher order terms
  - The accuracy of the estimate is bounded by the magnitude of the displacement and the second derivative of I.
- If we undo the motion, and reapply the estimator to the warped signal to find the residual motion left
  - Do this iteratively until the residual motion is small

Optical Flow: Iterative Estimation



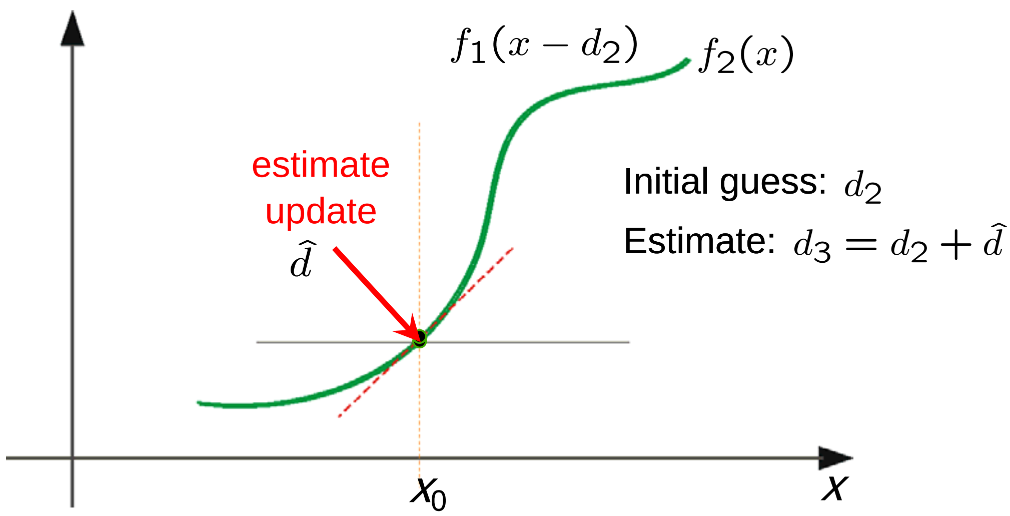
(using *d* for *displacement* here instead of *u*)

Optical Flow: Iterative Estimation



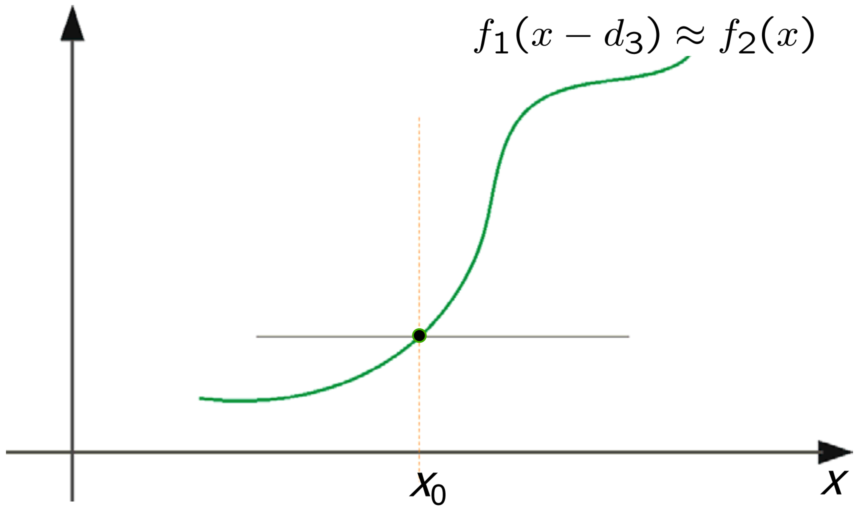
Szeliski

Optical Flow: Iterative Estimation



Szeliski

Optical Flow: Iterative Estimation



Szeliski

Optical Flow: Iterative Estimation

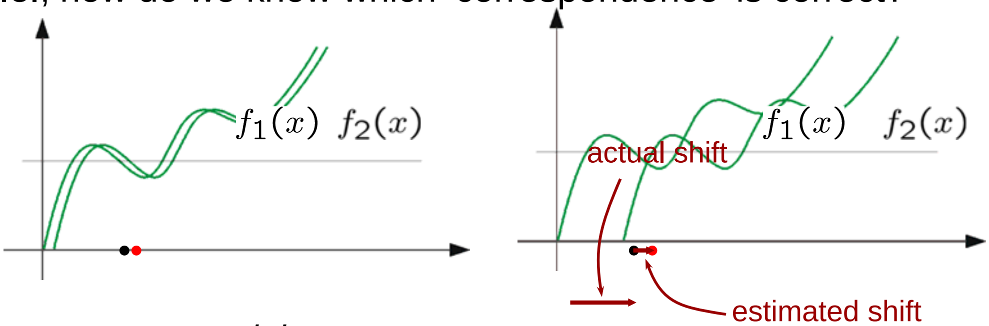
- Some Implementation Issues:
  - Warping is not easy (ensure that errors in warping are smaller than the estimate refinement)
  - Warp one image, take derivatives of the other so you don't need to re-compute the gradient after each iteration.
  - Often useful to low-pass filter the images before motion estimation (for better derivative estimation, and linear approximations to image intensity)

Szeliski

Optical Flow: Aliasing

Temporal aliasing causes ambiguities in optical flow because images can have many pixels with the same intensity.

I.e., how do we know which 'correspondence' is correct?



nearest match is correct (no aliasing)

nearest match is incorrect (aliasing)

To overcome aliasing: coarse-to-fine estimation.
At a coarse scale, the image is blurred and the motion velocity small.

The coarse-scale estimate is used to stabilize the finer scale motion.

Limits of the gradient method

Fails when intensity structure in window is poor

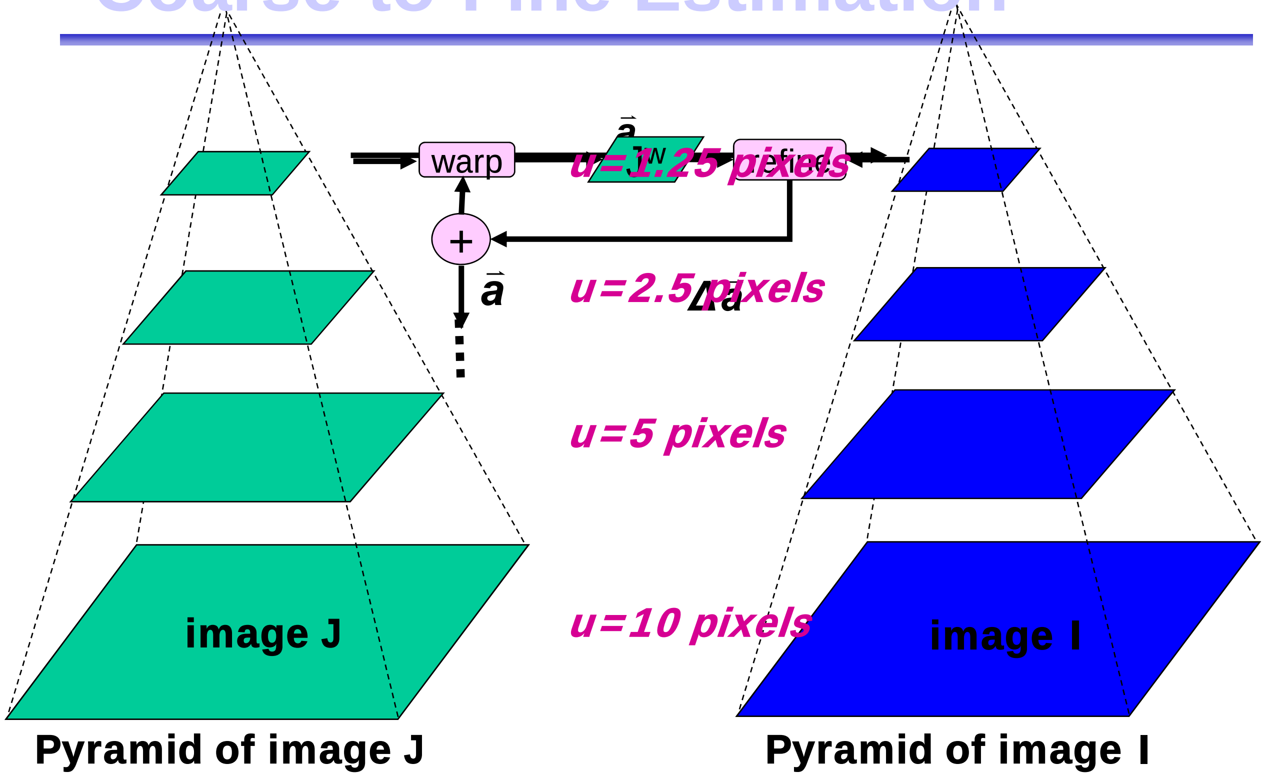
Fails when the displacement is large (typical operating range is motion of 1 pixel)

Linearization of brightness is suitable only for small displacements

- Also, brightness is not strictly constant in images

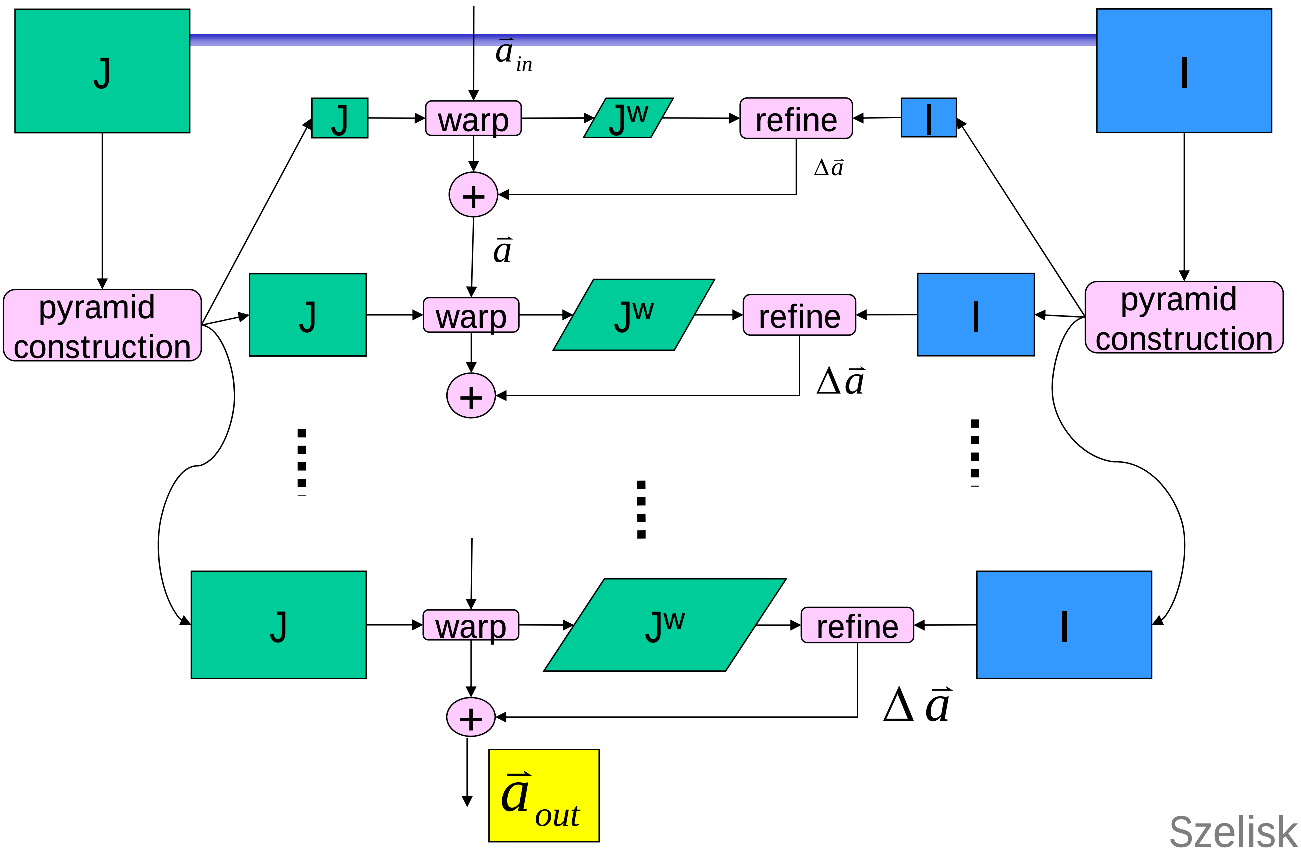
actually less problematic than it appears, since we can pre-filter images to make them look similar

Coarse-to-Fine Estimation



Szeliski

Coarse-to-Fine Estimation



Szeliski

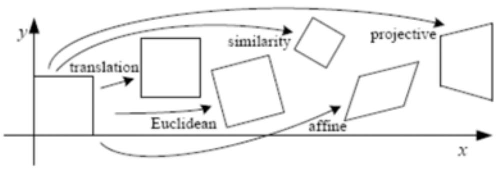
Parametric motion models (8.2)

- *2D Models:*
- Affine
- Quadratic
- Planar projective transform (Homography)

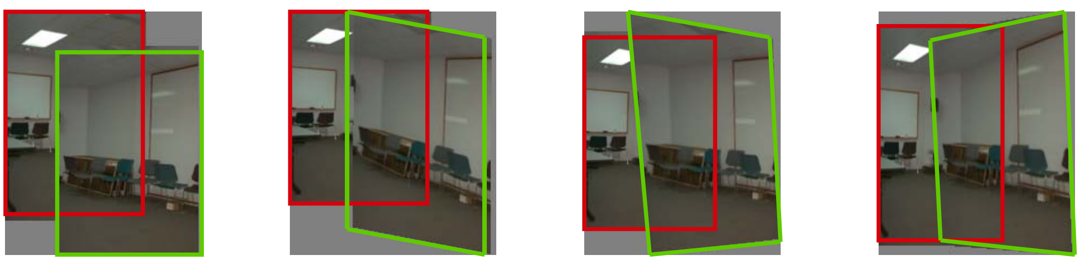
- *3D Models (see the book):*
- Instantaneous camera motion models
- Homography+epipole
- Plane+Parallax

Szeliski

Motion models



Translation	Affine	Perspective	3D rotation



2 unknowns	6 unknowns	8 unknowns	3 unknowns

Szelisl

Example: 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 B.C. Equation:

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

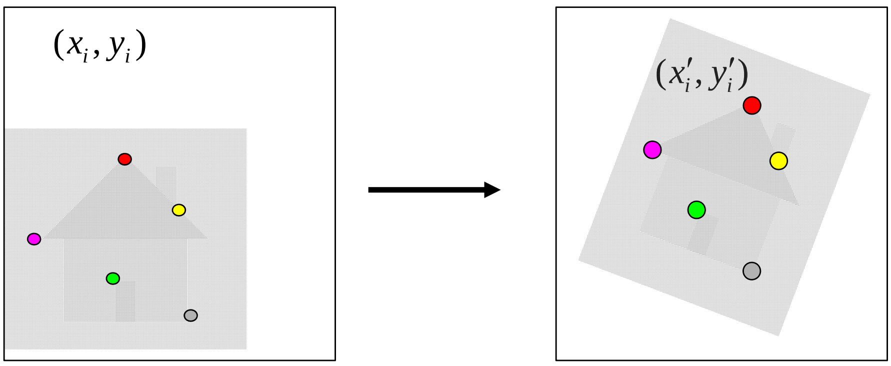
Each pixel provides 1 linear constraint in 6 *global* unknowns

Least Square Minimization (over all pixels):

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

Szeliski

Relation to last lecture: “Alignment”: Assuming we know the correspondences, how do we get the transformation?

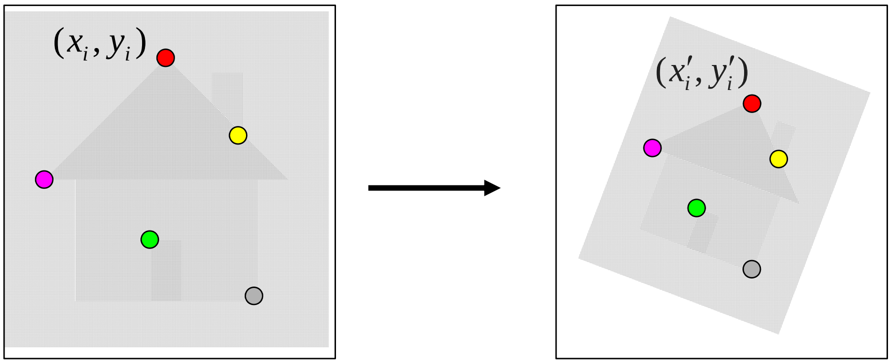


e.g., affine model in abs. coords…

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

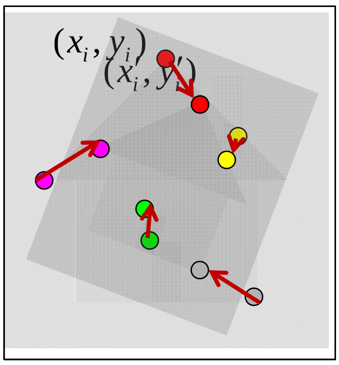
- ***Expressed in terms of absolute coordinates of corresponding points…***
- ***Generally presumed features separately detected in each frame***

Flow: Two views presumed in temporal sequence... **track** or analyze **spatio-temporal gradient**

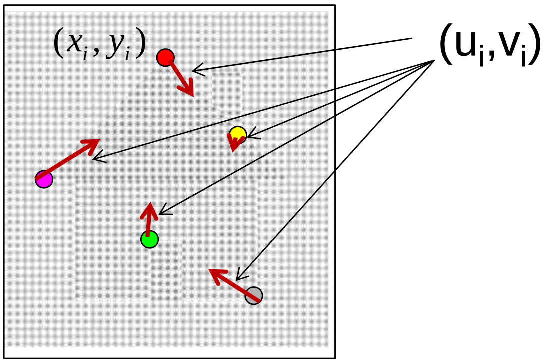


- ***Sparse or dense in first frame***
- ***Search in second frame***
- ***Motion models expressed in terms of position change***

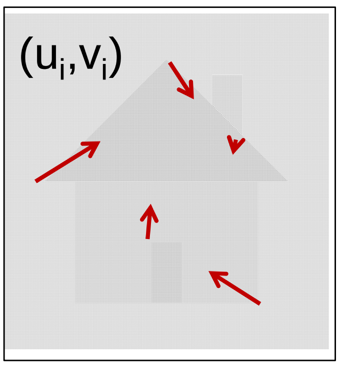
Parametric motion: Two views presumed in temporal sequence...**track** or analyze **spatio-temporal gradient**



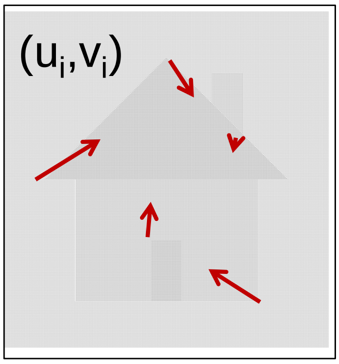
- ***Sparse or dense in first frame***
- ***Search in second frame***
- ***Motion models expressed in terms of position change***



- *Sparse or dense in first frame*
- *Search in second frame*
- *Motion models expressed in terms of position change*



- *Sparse or dense in first frame*
- *Search in second frame*
- *Motion models expressed in terms of position change*



Previous Alignment model:

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Now, Displacement model:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} a_2 & a_3 \\ a_5 & a_6 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} a_1 \\ a_4 \end{bmatrix}$$

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

- ***Sparse or dense in first frame***
- ***Search in second frame***
- ***Motion models expressed in terms of position change***

Other 2D Motion Models

Quadratic – instantaneous approximation to planar motion

$$u = q_1 + q_2 x + q_3 y + q_7 x^2 + q_8 xy$$
$$v = q_4 + q_5 x + q_6 y + q_7 xy + q_8 y^2$$

Projective – exact planar motion

$$x' = \frac{h_1 + h_2 x + h_3 y}{h_7 + h_8 x + h_9 y}$$
$$y' = \frac{h_4 + h_5 x + h_6 y}{h_7 + h_8 x + h_9 y}$$
and
$$u = x' - x, \quad v = y' - y$$

Szeliski

Discrete Search vs. Gradient Based

- Consider image I translated by u_0, v_0

$$I_0(x,y) = I(x,y)$$
$$I_1(x+u_0, y+v_0) = I(x,y) + \eta_1(x,y)$$
$$E(u,v) = \sum_{x,y} (I(x,y) - I_1(x+u, y+v))^2$$
$$= \sum_{x,y} (I(x,y) - I(x-u_0+u, y-v_0+v) - \eta_1(x,y))^2$$

- The discrete search method simply searches for the best estimate.
- The gradient method linearizes the intensity function and solves for the estimate

Szeliski

Correlation and SSD

- For larger displacements, do template matching
  - Define a small area around a pixel as the template
  - Match the template against each pixel within a search area in next image.
  - Use a match measure such as correlation, normalized correlation, or sum-of-squares difference
  - Choose the maximum (or minimum) as the match
  - Sub-pixel estimate (Lucas-Kanade)

Szelisk

Shi-Tomasi feature tracker

1. Find good features (min eigenvalue of 2×2 Hessian)
2. Use Lucas-Kanade to track with pure translation
3. Use affine registration with first feature patch
4. Terminate tracks whose dissimilarity gets too large
5. Start new tracks when needed

Szelis

Learning goals – motion estimation

- Understand representation and visualization of motion vectors.
- Understand the brightness similarity criterion.
- Know different patch similarity measures.
- Understand the gradient constraint.
- Know the basic steps in the optical flow algorithm
- Know strenghts and limitations of optical flow