INF 5300 - 29.04.2015
Dense motion and flow

Anne Schistad Solberg

- Motion perception
- Motion visualization
- Image similarity measures
- Motion estimation
- Optical flow algorithm

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 **only 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.

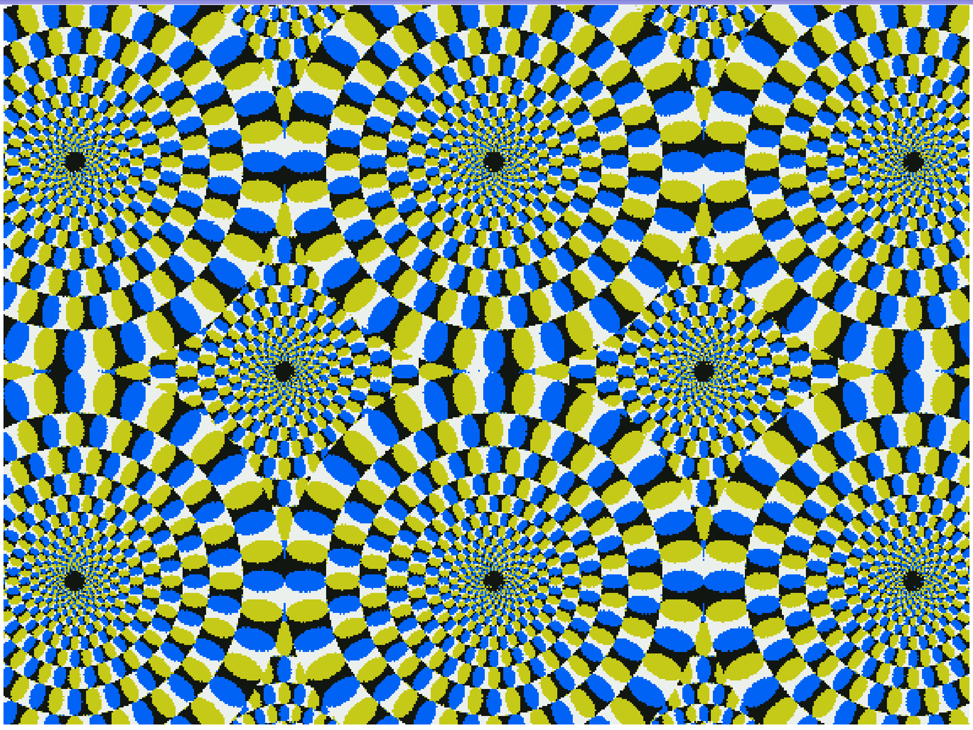
Applications of motion estimation

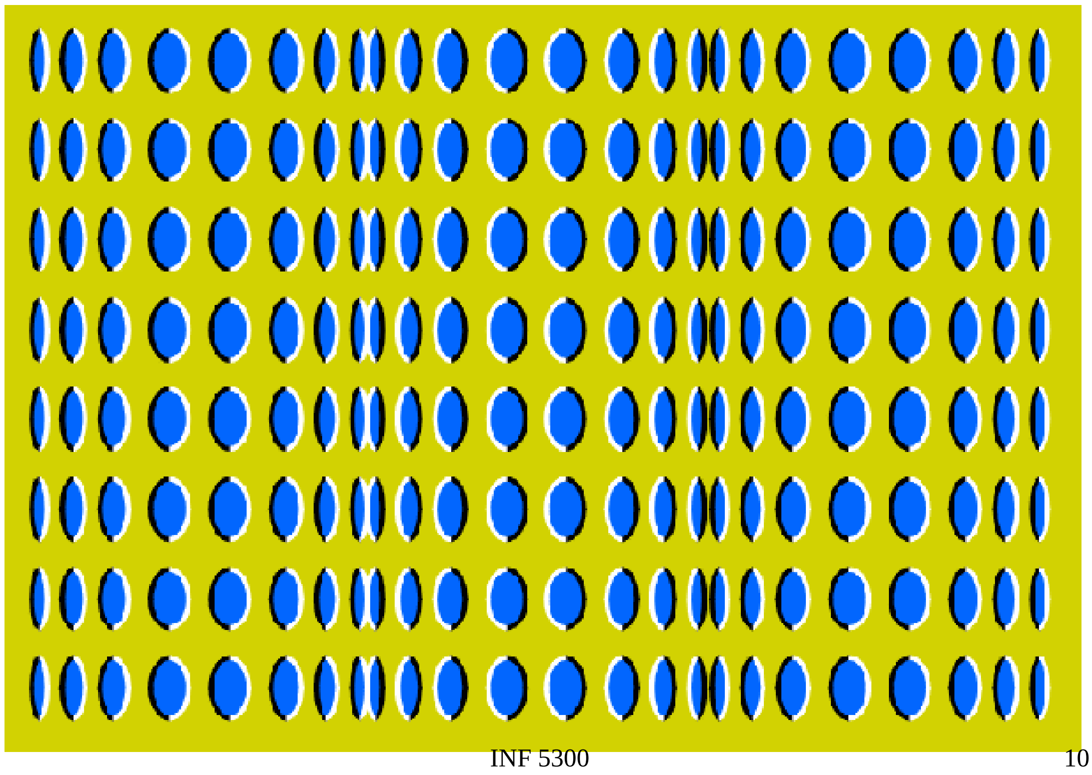
- Video enhancements:
  - Stabilization
  - Denoising
  - Super resolution
- 3D reconstruction: structure from motion
- Video segmentation
- Tracking/recognizing objects
- Learning dynamical models
- Advanced video editing

Motion estimation techniques

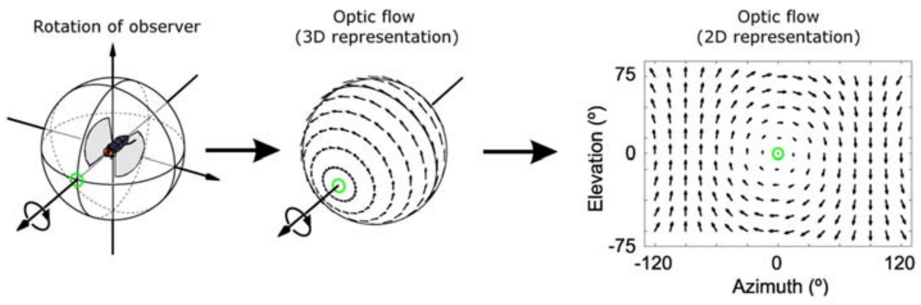
- Direct methods
  - Directly recover image motion at each pixel from spatio-temporal image brightness variations
  - Dense motion fields, but sensitive to appearance variations
  - Suitable for video when image motion is small
  - Computationally expensive
- Feature-based methods
  - Extract visual 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



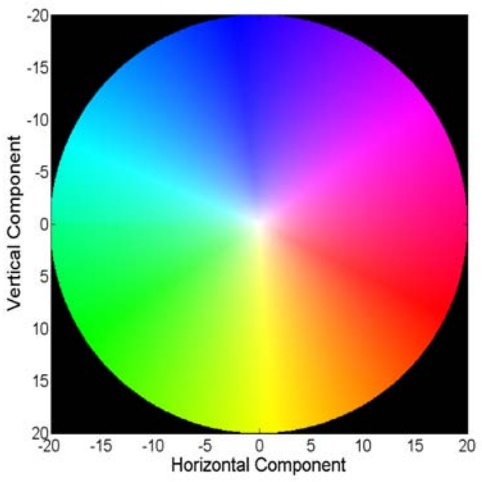
- Optical flow is the apparent motion of objects in a scene caused by the relative motion between an observer (eye or camera) and the scene.
- Parametric motion (e.g. using global geometric transforms) is limited and cannot describe the motion of arbitrary videos.
- Optical flow field: assign a flow 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 criteria

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

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

- Correlation, normalized cross correlation

Computing similarity between image patches

- A simple matching criterion: summed squared difference (**SSD**):

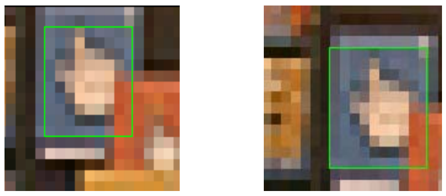
$$E_{SSD}(u) = \sum_i [I_1(x_i + u) - I_0(x_i)]^2$$

- I_0 and I_1 are the two images, **u**=(u,v) the displacement vector.
- Movement can be at the sub-pixel level so interpolation might be needed.
- A measure more robust to outliers is

$$E_{SRD} = \sum_i \rho\big(I_1(x_i + u) + I_0(x_i)\big)$$

$$\rho(x) = \frac{x^2}{1 + \frac{x^2}{a^2}}$$

  - a is a constant called outlier threshold



Computing similarity between image patches

- Spatially varying weights can be needed (e.g. on image boundaries where the weights are zero)

$$E_{WSSD}(u) = \sum_i w_0(x_i) w_1(x_i + u)[I_1(x_i + u) - I_0(x_i)]^2$$

- Differences in bias β and gain α can be handled by a simple linear intensity model between the images:

$$I_1(x+u) = (1+\alpha) I_0(x) + \beta$$

- This criterion handles changes in bias and gain:

$$E_{BG}(u) = \sum_i [I_1(x_i + u) - (1-\alpha) I_0(x_i) - \beta]^2 = \sum_i [\alpha I_0(x_i) - \beta - e_i]^2$$

- To handle bias and gain properly it might be necessary to perform a linear regression, which is computationally more costly.

Computing similarity between image patches

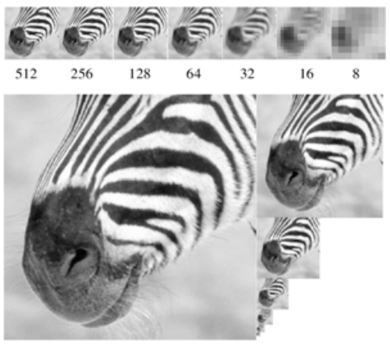
- Normalized cross-correlation can also be used

$$E_{NCC}(u) = \frac{\sum_i [(I_0(x_i) - \bar{I}_0) - (I_1(x_i) - \bar{I}_1)]}{\sqrt{\sum_i (I_0(x_i) - \bar{I}_0)^2} \sqrt{\sum_i (I_1(x_i) - \bar{I}_1)^2}}$$

$\bar{I}_0$ and $\bar{I}_1$ are the mean of the respective patches.

- Correlation works well if the changes in contrast are large, but not well for homogeneous areas where a good match can be found for many different locations.

Hierarchical search for matches – block matching



- In motion estimation, there are often small motion between frames, so the search is restricted to a small region (e.g. ±16 pixels) from a given position.
- This is called block matching.
- Hiearchical motion estimation is often used to speed up the process.
  - An image pyramid is created by decimation and smoothing, consisting of images.

$I_k^{(l)}(x_j) \leftarrow \tilde{I}_k^{(l-1)}(2x_j)$

$\tilde{I}_k^{(l-1)}(2x_j)$ is a smoothed version of the image at level l - 1

  - At the coarsest level, we do a full search in a window for the displacement u(l) that minimizes

$$I_o^{(l)} - I_1^{(l)}$$

  - This value of the motion vector is then used to predict the displacement at the finer level:

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

Fourier-based matching

- Fast when the patch size is large.
- Based on the fact that the Fourier transform of a shifted signal has the same magnitude but a linearly shifted phase:

$$\mathcal{F}\left\{I_1(x+u)\right\} = \mathcal{F}\left\{I_1(x)\right\}e^{-ju\omega}$$

- Remember that convolution in the spatial domain is equivalent to multilication in the Fourier domain, and correlation in the spatial domain is equivalent to multiplication with the complex conjugate in the Fourier domain.
- Thus, correlation is computed efficiently in the Fourier domain.
- Windowed correlation must be used to handle pixels outside image boundaries in one of the images.
- Fourier-based matching can also be made invariant to rotation and scale by transforming the image to polar coordinates (rotation) or log-polar coordinates (rotation and scale).

Fourier-based matching: Phase correlation

- Phase correlation weights the signal by the magnitude of the Fourier transforms:

$$\mathcal{F}\{E_{PC}(u)\} = \frac{\mathcal{F}\{I_0(x)\}\mathcal{F}^*\{I_0(x)\}}{\|\mathcal{F}\{I_0(x)\}\|\|\mathcal{F}\{I_0(x)\}\|}$$

- For a noiseless signal, this gives a single spike located at the correct value of u.
- Some argue that phase correlation is more accurate the regular cross-correlation.

Lucas and Kanade optical flow

- Good image stabilization requires subpixel accuracy.
- Assume that matching is based on the SSD-criterion.
- Lucas and Kanade did a gradient descent on the SSD function to refine the shift in u based on a Taylor series expansion of $I_1(x_i+u+\Delta u)$.
- Let

$$J_1(x_i+u) = \nabla I_1(x_i+u)$$

- Define

$$e_i = I_1(x_i+u) - I_0(x_i)$$

- The modified SSD criterion is then:

$$E_{LK-SSD}(u+\Delta u) = \sum_i \left[I_1(x_i+u+\Delta u) - I_0(x_i)\right]^2$$

$$\approx \sum_i \left[I_1(x_i+u) + J_1(x_i+u)\Delta u - I_0(x_i)\right]^2$$

$$= \sum_i \left[J_1(x_i+u)\Delta u + e_i\right]^2$$

Incremental refinement/subpixel alignment

- The gradient at a subpixel location (x_i+u) can for example be computed as the simple horizontal and vertical differences between pixel x_i and its horizontal or vertical neighbor.
- We often write $J_1(x_i+u)\Delta u+e_i$ as the brightness constancy constraint equation:

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

- This is also called the optical flow constraint. I_x and I_y are the spatial derivatives and I_t the temporal derivative.

Gradient Constraint (or the Optical Flow Constraint)

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

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 (gradient tensor)

Weighted version

A weight function can be used to weight constraints in the center of the neighborhood with a gaussian function g(x)

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

Minimizing

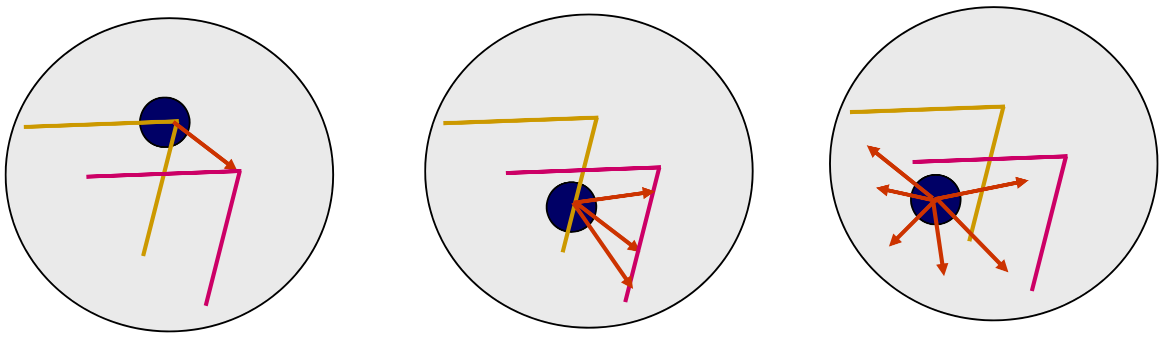
$$\begin{bmatrix} \sum g I_x^2 & \sum g I_x I_y \\ \sum g I_x I_y & \sum g I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} \sum g I_x I_t \\ \sum g I_y I_t \end{pmatrix}$$

Balance spatial gradients by temporal gradients and the shift in u

Note: to compute the derivatives, take care so they center at the same location (e.g. I(x,y)-I(x+1,y) will not center at the same location in all directions.

Local Patch Analysis

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



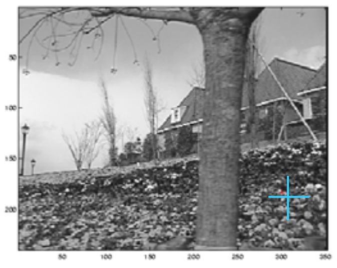
The Aperture Problem

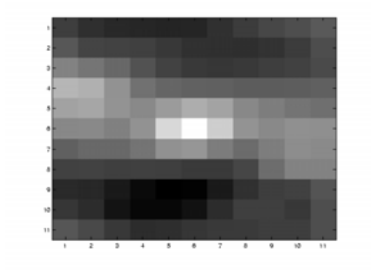
Let $A = \sum (\nabla I)(\nabla I)^T$ and $b = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$

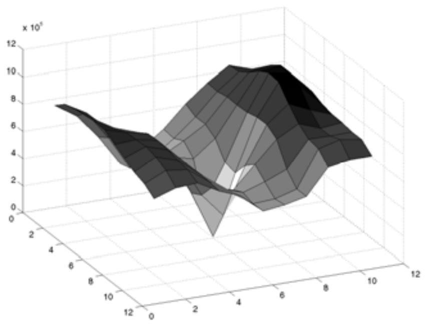
- Algorithm: At each pixel compute U by solving $AU = b$
- *A* is singular if all gradient vectors point in the same direction
  - e.g., along an edge
  - of course, trivially singular if the summation is over a single pixel or there is no texture
  - i.e., only *normal flow* is available (aperture problem)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

SSD Surface – Textured area

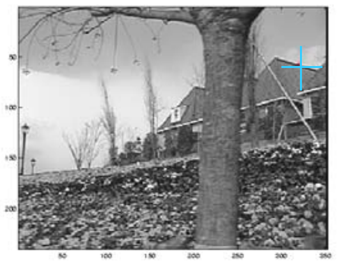
Have you seen this before?
Remember lecture
on keypoint detection

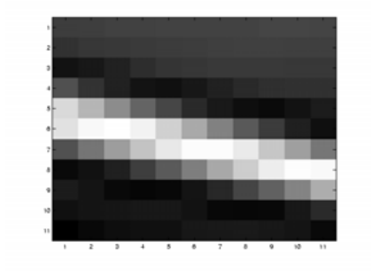


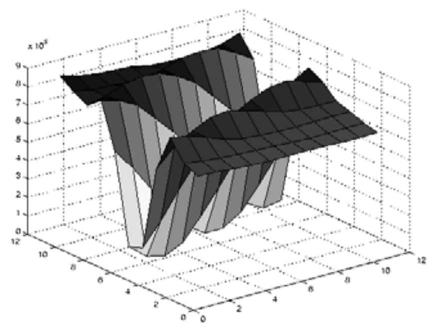




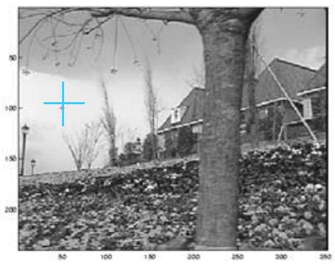
SSD Surface -- Edge

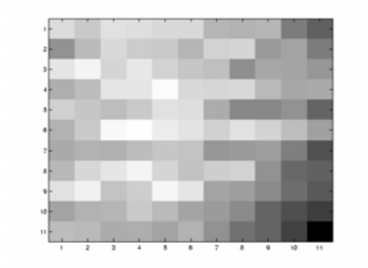


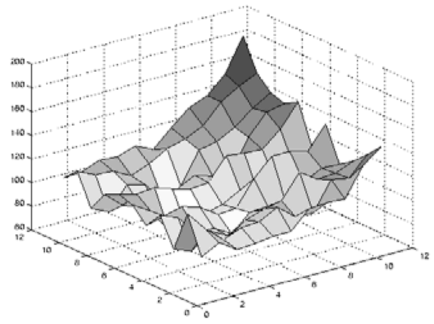




SSD – homogeneous area







Alternative energy functions

- This can also be applied if we have bias/gain shifts:

$$E_{BG}(u) = \sum_i \left[I_1(x_i + u) - (1-\alpha)I_0(x_i) - \beta\right]^2 = \sum_i \left[\alpha I_0(x_i) - \beta - e_i\right]^2$$

- This will result in a 4x4 equation system.
- A
- There is also a version solving the weighted window energy function (both images are weighted, to handle e.g. boundaries):

$$E_{WSSD}(u) = \sum_i w_0(x_i) w_1(x_i + u)\left[I_1(x_i + u) - I_0(x_i)\right]^2$$

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
  - Let ut now explain this

Iterative refinement explained in 1D

- Let $f_1(x)$ and $f_2(x)$ be the 1D signals at two time instances.
- We assumed linear motion locally in a Taylor series expansion where we ignored second-order terms:

$$f_1(x-d) = f_1(x) - df_1^{'}(x) + O(d^2 f_1^{''})$$

which gives the approximation of d

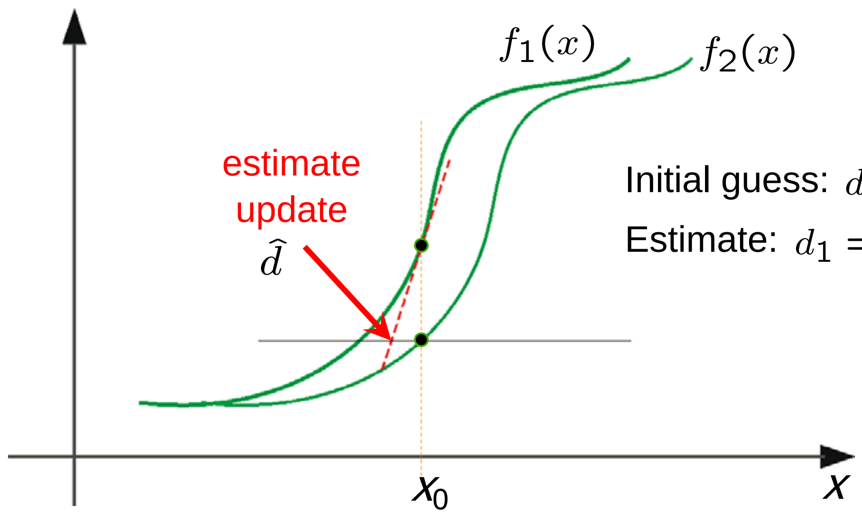
$$\hat{d} = \frac{f_1(x) - f_2(x)}{f_1^{'}(x)}$$

- The accuracy of this estimates depends on magnitude of d and the second derivative of f1:

$$|d - \hat{d}| \le \frac{d^2 |f_1^{''}(x)|}{2|f_1^{'}(x)|}$$

- If we apply iterations in a Gauss-Newton style we use the current estimate to undo the motion, and reapply the estimator to the warped signal iteratively.

Optical Flow: Iterative Estimation

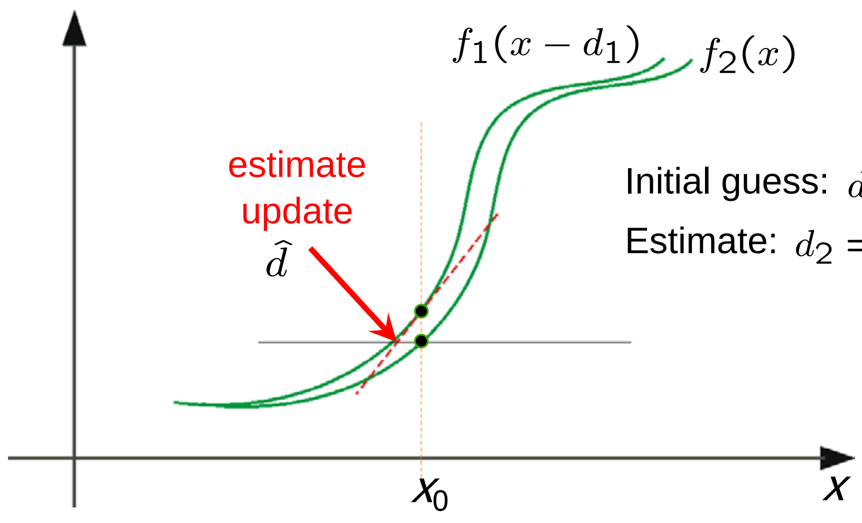


Initial guess: $d_0 = 0$

Estimate: $d_1 = d_0 + \hat{d}$

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

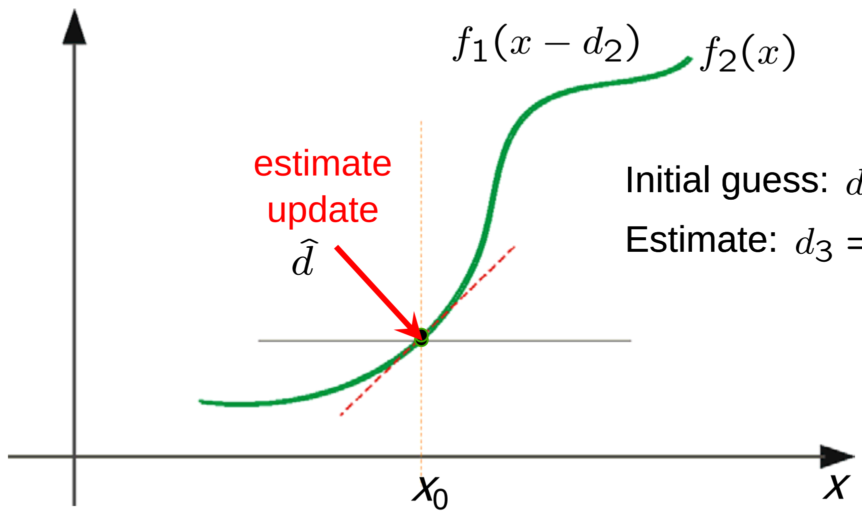
Optical Flow: Iterative Estimation



Initial guess: d_1

Estimate: $d_2 = d_1 + \hat{d}$

Optical Flow: Iterative Estimation

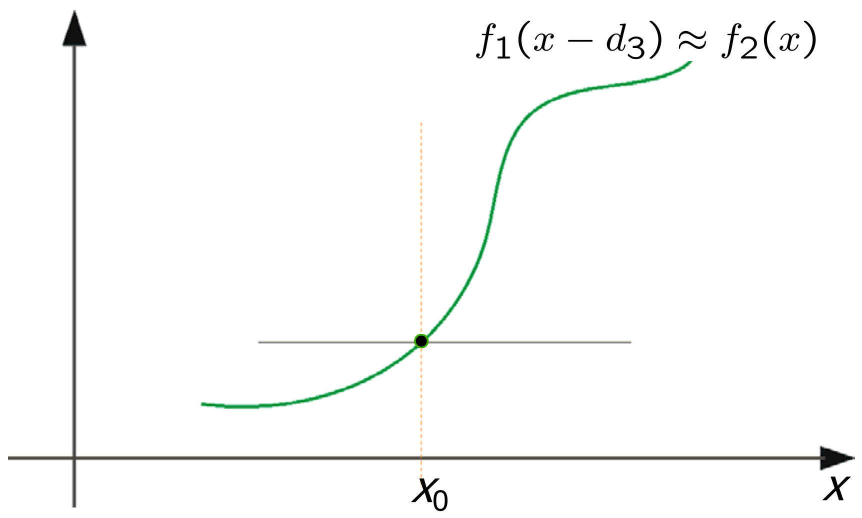


Initial guess: d_2

Estimate: $d_3 = d_2 + \hat{d}$

Szeliski

Optical Flow: Iterative Estimation



Szelisk

Optical Flow: Iterative Estimation

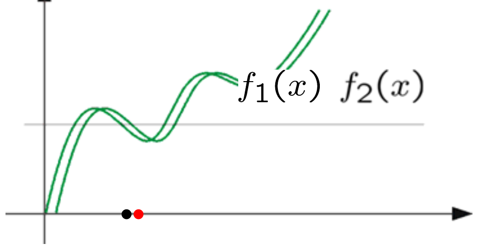
- Some Implementation Issues:
  - Warping is not easy (ensure that errors in warping are smaller than the estimate refinement)
  - Warp one image, take derivatives of the other so you don't need to re-compute the gradient after each iteration.
  - Often useful to low-pass filter the images before motion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

Szeliski

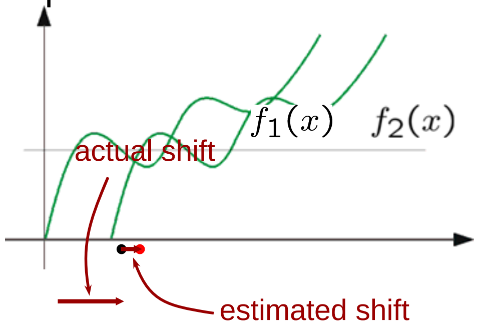
Optical Flow: Aliasing

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)

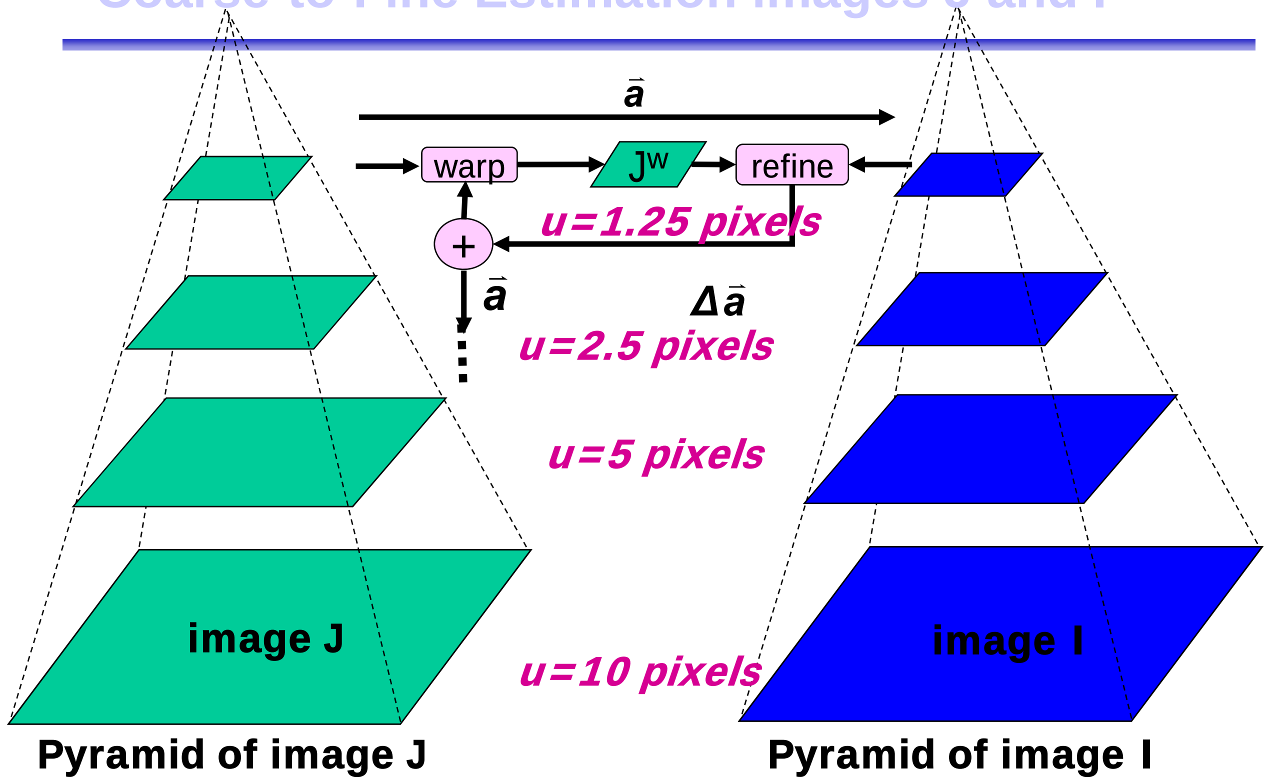
Linearization of brightness is suitable only for small displacements

- Also, brightness is not strictly constant in images

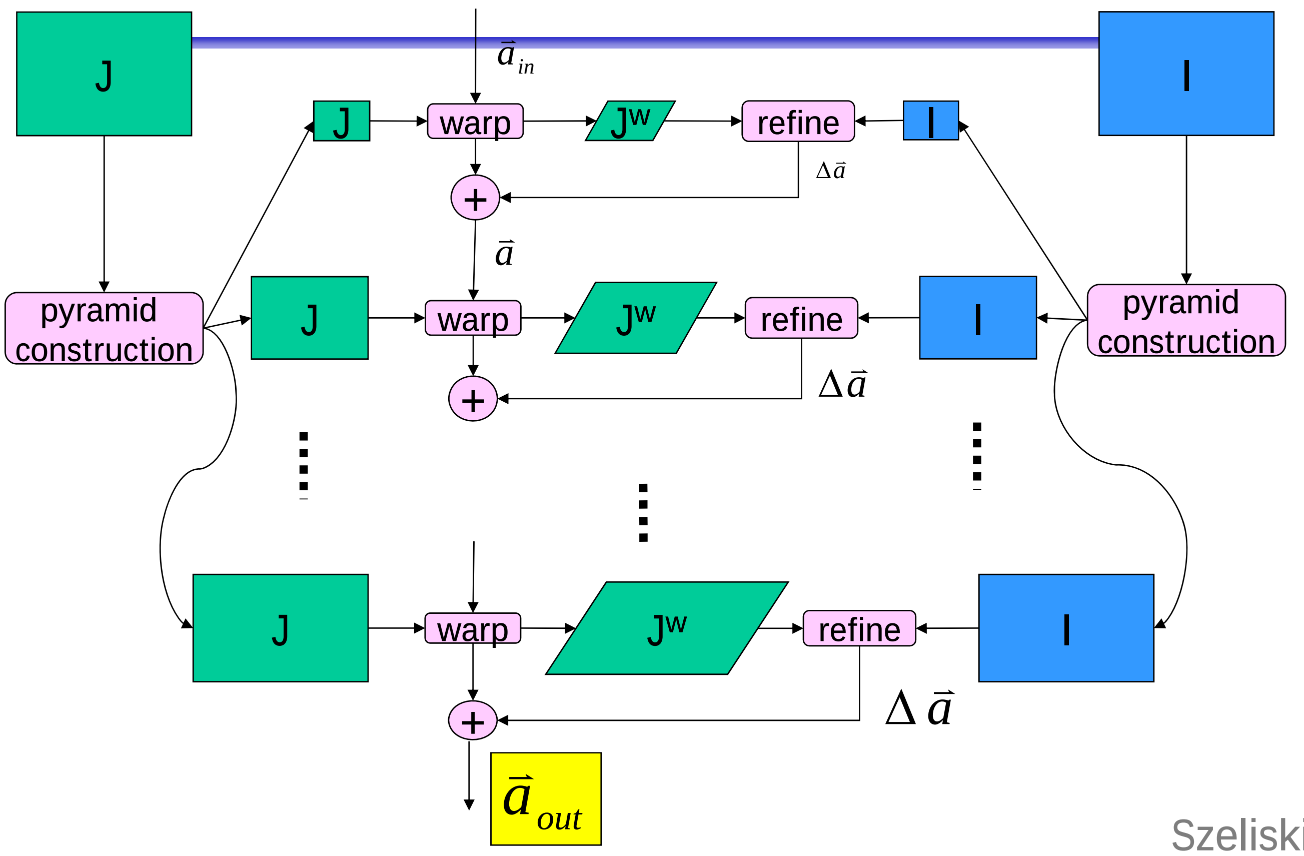
actually less problematic than it appears, since we can pre-filter images to make them look similar

Szelis

Coarse-to-Fine Estimation images J and I



Coarse-to-Fine Estimation



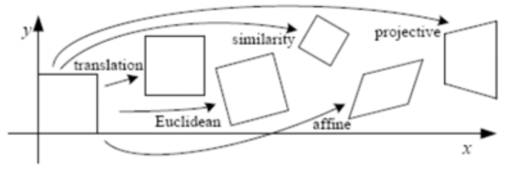
Szeliski

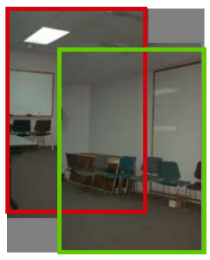
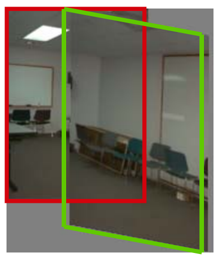
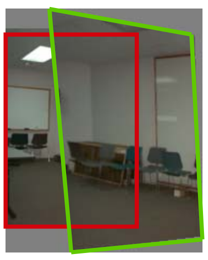
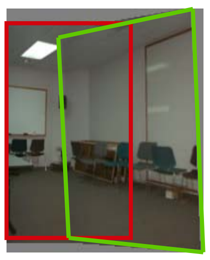
Parametric motion models (8.2)

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

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

Motion models



Translation	Affine	Perspective	3D rotation
			
2 unknowns	6 unknowns	8 unknowns	3 unknowns

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 brightness constraint equation:

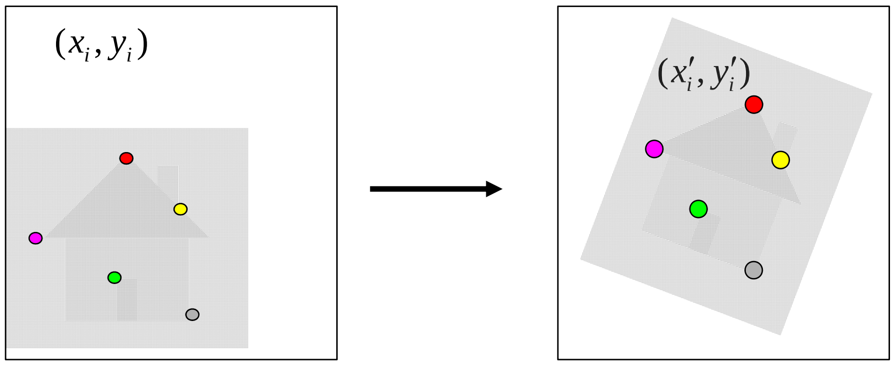
$$\boxed{I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \approx 0}$$

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

Least Square Minimization (over all pixels):

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

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

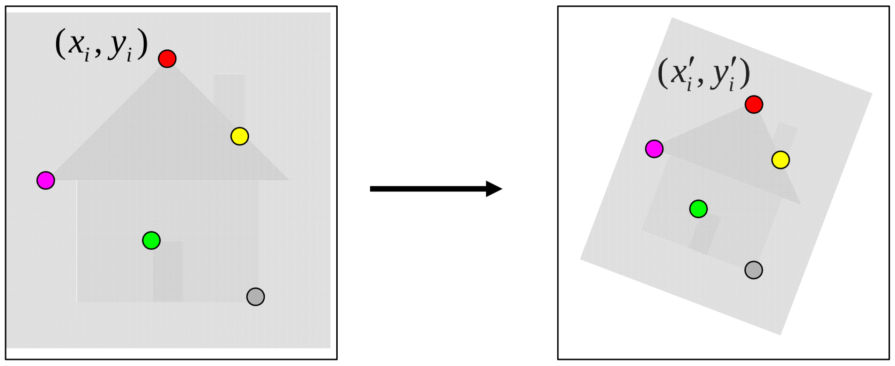


e.g., affine model in abs. coords...

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

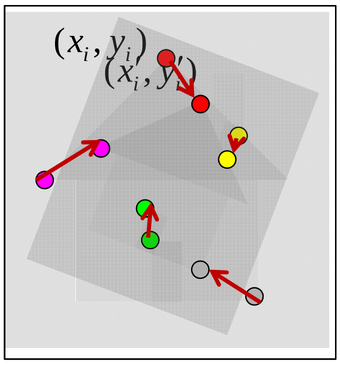
- ***Expressed in terms of absolute coordinates of corresponding points…***
- ***Generally presumed features separately detected in each frame***

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

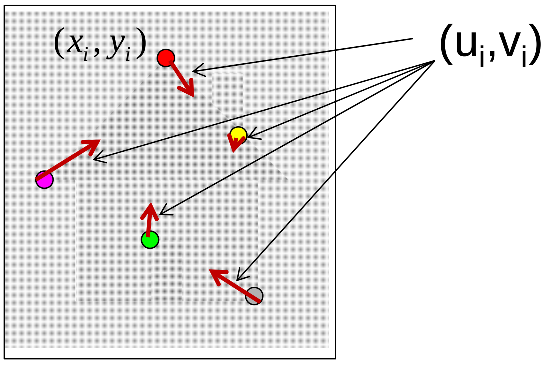


- ***Sparse or dense in first frame***
- ***Search in second frame***
- ***Motion models expressed in terms of position change***

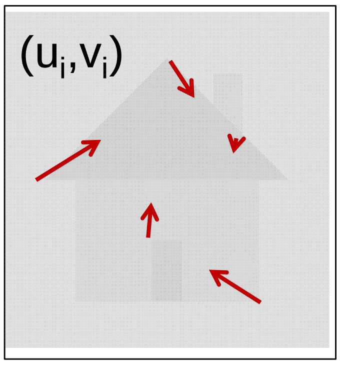
Parametric motion: Two views presumed in temporal sequence...**track** or analyze **spatio-temporal gradient**



- ***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***



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

$$u = x' - x, \quad v = y' - y$$

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, e.g. by correlation/block matching (no sub-pixel).
- The gradient method linearizes the intensity function and solves for the estimate.

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)

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

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