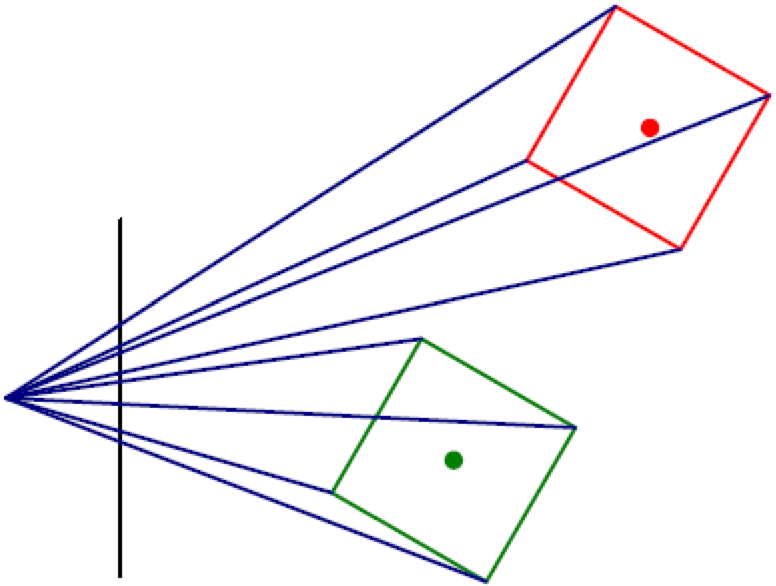
Autonomous Navigation for Flying Robots

Lecture 7.1:
2D Motion Estimation
in Images

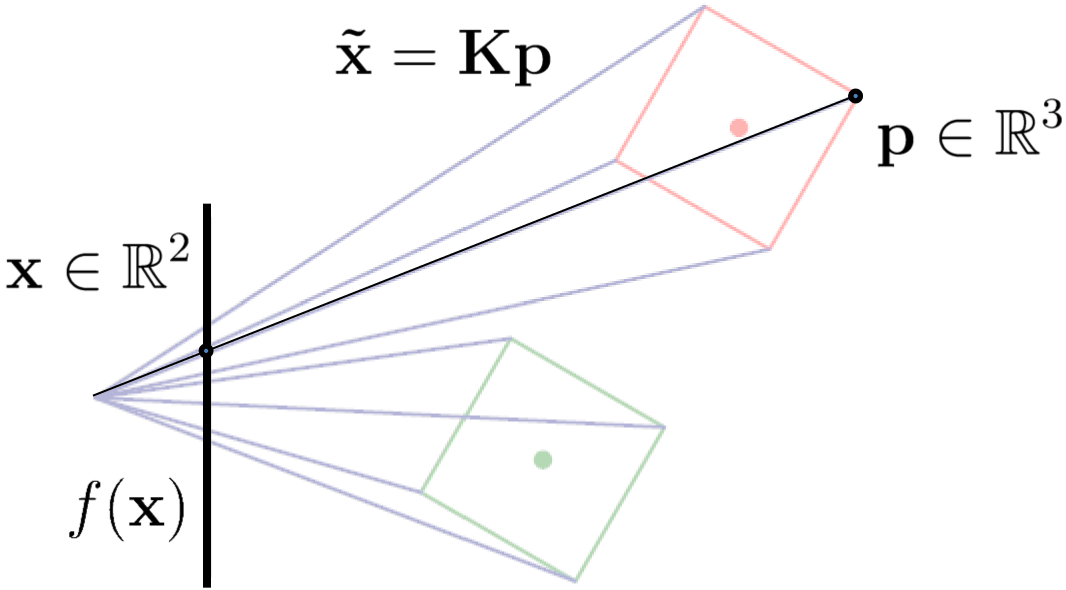
Jürgen Sturm
Technische Universität München

3D to 2D Perspective Projections



Richard Szeliski, Computer Vision: Algorithms and Applications
http://szeliski.org/Book/

3D to 2D Perspective Projections



Richard Szeliski, Computer Vision: Algorithms and Applications
http://szeliski.org/Book/

Images

- We can think of an image as a function $f : \mathbb{R}^2 \mapsto \mathbb{R}$
- $f(\mathbf{x})$ gives the intensity at position $\mathbf{x}$
- Realistically, the image function is only defined on a rectangle and has finite range

  $$f : [0, W-1] \times [0, H-1] \mapsto [0, 1]$$

Digital Images

- Image function is sampled discrete pixel locations
- Image can be represented as a matrix

j

i

111	115	113	111	112	111	112	111
135	138	137	139	145	146	149	147
163	168	188	196	206	202	206	207
180	184	206	219	202	200	195	193
189	193	214	216	104	79	83	77
191	201	217	220	103	59	60	68
195	205	216	222	113	68	69	83
199	203	223	228	108	68	71	77

Problem Statement

- **Given:** two camera images f_0, f_1
- **Goal:** estimate the camera motion $\mathbf{u}$

- For the moment, let's assume that the camera only moves in the xy-plane, i.e., $\mathbf{u} = \begin{pmatrix} u & v \end{pmatrix}^\top$
- Extension to 3D follows

General Idea

1. Define an error metric $E(\mathbf{u})$ that defines how well the two images match given a motion vector
2. Find the motion vector with the lowest error

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} E(\mathbf{u})$$

Error Metrics for Image Comparison

- Sum of Squared Differences (SSD)

$$E_{\mathrm{SSD}}(\mathbf{u}) = \sum_i \left(f_1(\mathbf{x}_i + \mathbf{u}) - f_0(\mathbf{x}_i)\right)^2 = \sum_i e_i^2$$

with displacement $\mathbf{u} = (u\ v)^\top$

and residual errors $e_i = f_1(\mathbf{x}_i + \mathbf{u}) - f_0(\mathbf{x}_i)$

Windowed SSD

- Images (and image patches) have finite size
- Standard SSD has a bias towards smaller overlaps (less error terms)
- Solution: divide by the overlap area
- Root mean square error

$$E_{\mathrm{RMS}}(\mathbf{u}) = \sqrt{E_{\mathrm{SSD}}/A}$$

Cross Correlation

- Maximize the product (instead of minimizing the differences)

$$E_{\mathrm{CC}}(\mathbf{u}) = -\sum_i f_0(\mathbf{x}_i) f_1(\mathbf{x}_i + \mathbf{u})$$

- Normalized cross correlation (between -1..1)

$$E_{\mathrm{NCC}}(\mathbf{u}) = \\ -\sum_i \frac{(f_0(\mathbf{x}_i) - \operatorname{mean} f_0)(f_1(\mathbf{x}_i + \mathbf{u}) - \operatorname{mean} f_1)}{\sqrt{\operatorname{var} f_0 \operatorname{var} f_1}}$$

- Less sensitive to illumination changes

General Idea

1. Define an error metric $E(\mathbf{u})$ that defines how well the two images match given a motion vector
2. **Find the motion vector with the lowest error**

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} E(\mathbf{u})$$

Finding the minimum

- Full search (e.g., ±16 pixels)
- Gradient descent
- Hierarchical motion estimation

Motion Estimation

- Perform Gauss-Newton minimization on the SSD energy function (Lucas and Kanade, 1981)
- Gauss-Newton minimization
  - Linearize residuals w.r.t. to camera motion
  - Yields quadratic cost function
  - Build normal equations and solve linear system

Motion Estimation

- Error function

$$E_{\mathrm{SSD}}(\mathbf{u}) = \sum_i \left(f_1(\mathbf{x}_i + \mathbf{u}) - f_0(\mathbf{x}_i)\right)^2 = \sum_i e_i^2$$

- Linearize in u

$$E_{\mathrm{SSD}}(\mathbf{u} + \Delta\mathbf{u}) = \sum_i \left(f_1(\mathbf{x}_i + \mathbf{u} + \Delta\mathbf{u}) - f_0(\mathbf{x}_i)\right)^2$$

Motion Estimation

- Taylor expansion of energy function

$$E_{\text{SSD}}(\mathbf{u}+\Delta\mathbf{u}) = \sum_i (f_1(\mathbf{x}_i+\mathbf{u}+\Delta\mathbf{u}) - f_0(\mathbf{x}_i))^2$$

$$\approx \sum_i (f_1(\mathbf{x}_i+\mathbf{u}) + \mathbf{J}_1(\mathbf{x}_i+\mathbf{u})\Delta\mathbf{u} - f_0(\mathbf{x}_i))^2$$

$$= \sum_i (\mathbf{J}_1(\mathbf{x}_i+\mathbf{u})\Delta\mathbf{u} + e_i)^2$$

with $\mathbf{J}_1(\mathbf{x}_i+\mathbf{u}) = \nabla f_1(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_i+\mathbf{u}} = \left(\frac{\partial f_1(\mathbf{x})}{\partial x}, \frac{\partial f_1(\mathbf{x})}{\partial y}\right)^\top\Big|_{\mathbf{x}=\mathbf{x}_i+\mathbf{u}}$

Least Squares Minimization

- Goal: Minimize

$$E(\mathbf{u} + \Delta\mathbf{u}) \approx \sum_i (\mathbf{J}_1(\mathbf{x}_i + \mathbf{u})\Delta\mathbf{u} + e_i)^2$$

- Solution: Compute derivative and set to zero

$$\frac{\partial E(\mathbf{u} + \Delta\mathbf{u})}{\partial \Delta\mathbf{u}} = 2\mathbf{A}\Delta\mathbf{u} + 2\mathbf{b} \stackrel{!}{=} 0$$

with $\mathbf{A} = \sum_i \mathbf{J}_1^\top(\mathbf{x}_i + \mathbf{u})\mathbf{J}_1(\mathbf{x} + \mathbf{u})$

and $\mathbf{b} = \sum_i e_i \mathbf{J}_1^\top(\mathbf{x}_i + \mathbf{u})$

Least Squares Minimization

Step 1: Compute A,b from image gradients using

$$\mathbf{A} = \begin{pmatrix} \sum f_x^2 & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^2 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} \sum f_x f_t \\ \sum f_y f_t \end{pmatrix}$$

with $f_x = \dfrac{\partial f_1(\mathbf{x})}{\partial x}, f_y = \dfrac{\partial f_1(\mathbf{x})}{\partial y}$

and $f_t = \dfrac{\partial f_t(\mathbf{x})}{\partial t} \left[\approx f_1(\mathbf{x}) - f_0(\mathbf{x}) \right]$

Least Squares Minimization

Step 2: Solve linear system

$$\mathbf{A}\Delta\mathbf{u} = -\mathbf{b}$$

$$\Rightarrow \qquad \Delta\mathbf{u} = -\mathbf{A}^{-1}\mathbf{b}$$

Note: All of the required computations are super-fast!

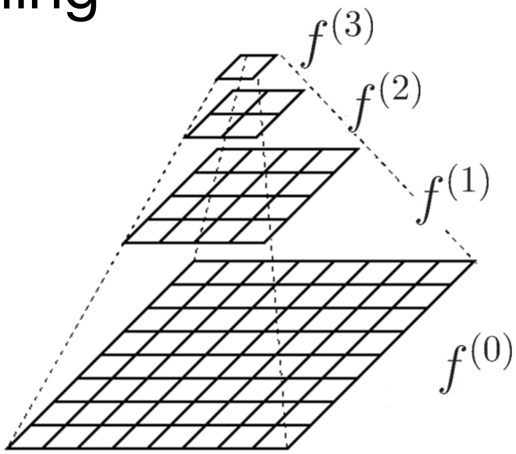
In step 1: image gradients + summation to build A,b

In step 2: solve a 2x2 linear equation

Hierarchical Motion Estimation

- Construct image pyramid by downsampling

$$f_k^{(l+1)}(\mathbf{x}_i) \leftarrow f_k^{(l)}(2\mathbf{x}_i)$$



- Estimate motion on coarse level
- Use as initialization for next finer level

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

Covariance of the Estimated Motion

- Assuming (small) Gaussian noise in the images

$$f_{\text{obs}}(\mathbf{x}_i) = f_{\text{true}}(\mathbf{x}_i) + \epsilon_i$$

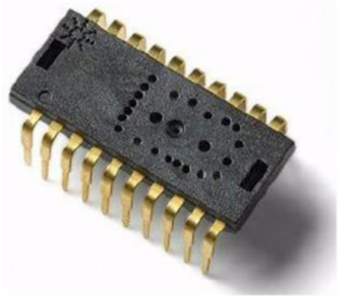
  with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$

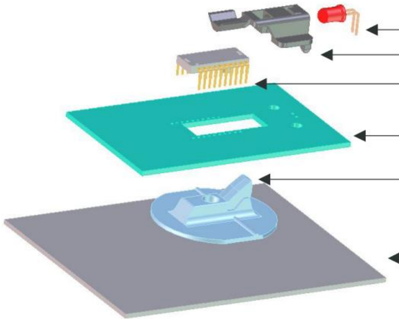
- … results in uncertainty in the motion estimate with covariance (e.g., useful for Kalman filter)

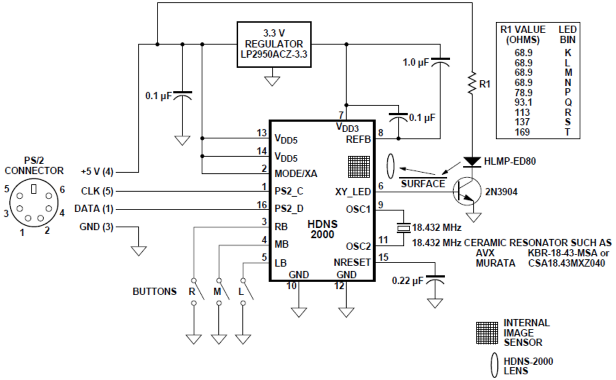
$$\mathbf{\Sigma}_u = \sigma^2 \mathbf{A}^{-1}$$

Optical Computer Mouse (since 1999)

- E.g., ADNS3080 from Agilent Technologies, 2005
  - 6400 fps
  - 30x30 pixels
  - 4 USD







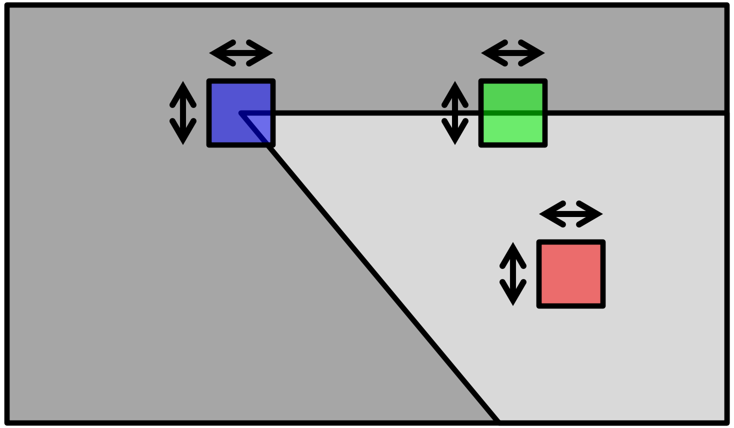
http://www.alldatasheet.com/datasheet-pdf/pdf/203607/AVAGO/ADNS-3080.html

Image Patches

- Sometimes we are interested of the motion of small image patches
- **Problem:** some patches are easier to track than others
- Which patches are easy/difficult to track?
- How can we recognize “good” patches?

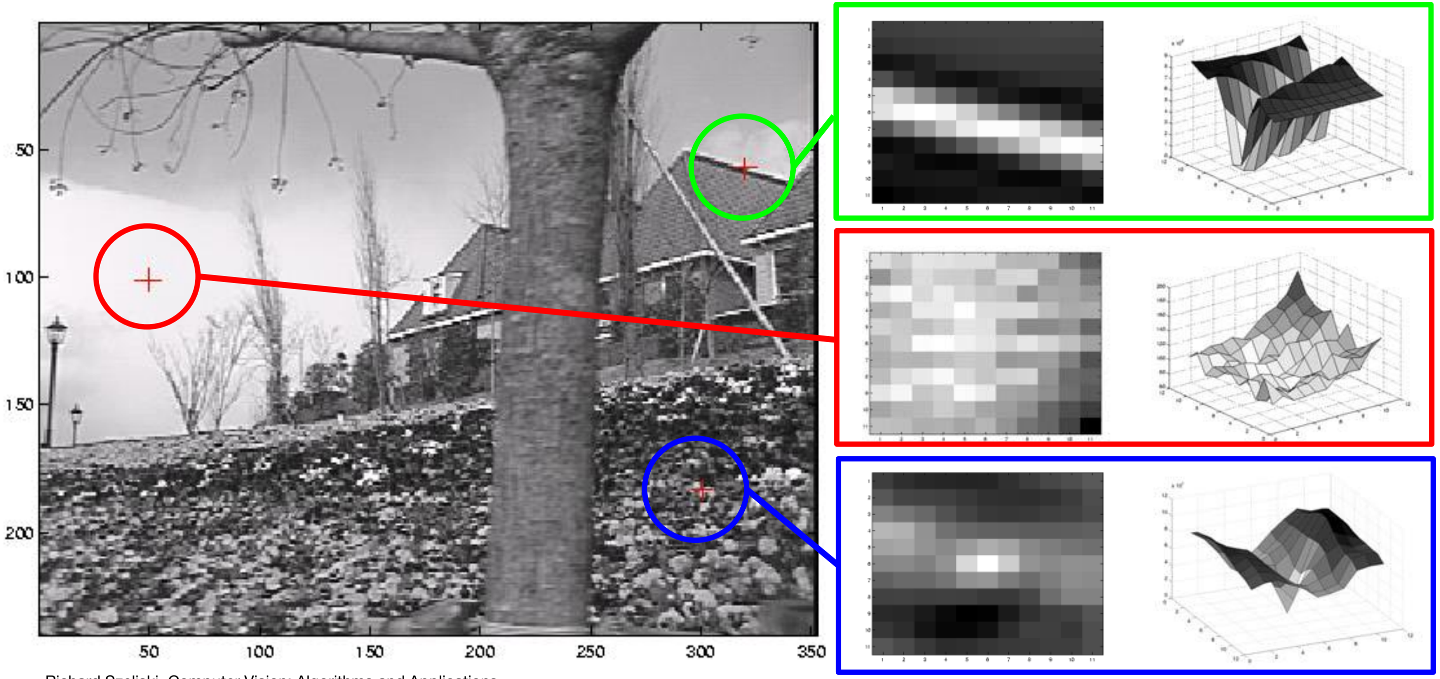
Image Patches

- Sometimes we are interested of the motion of a small image patches
- **Problem:** some patches are easier to track than others



Example

- Let's look at the shape of the energy



Richard Szeliski, Computer Vision: Algorithms and Applications
http://szeliski.org/Book/

Corner Detection

- **Idea:** Inspect eigenvalues λ_1, λ_2 of matrix A (Hessian)
  - λ_1, λ_2 small → no point of interest
  - λ_1 large, λ_2 small → edge
  - λ_1, λ_2 large → corner

$$\mathbf{A} = \begin{pmatrix} \sum f_x^2 & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^2 \end{pmatrix}$$

Corner Detection

- Harris detector (does not need eigenvalues)

$$\lambda_1\lambda_2 > \kappa\,(\lambda_1 + \lambda_2)^2 \Leftrightarrow \det(A) > \kappa\ \mathrm{trace}^2(A)$$

- Shi-Tomasi (or Kanade-Lucas)

$$\min(\lambda_1, \lambda_2) > \kappa$$

Other Detectors

- Förstner detector:
  localize corner with sub-pixel accuracy
- FAST corners:
  learn decision tree, minimize number of tests → super fast
- Difference of Gaussians (DoG):
  scale-invariant detector used for SIFT
- …

Kanade-Lucas-Tomasi (KLT) Tracker

- Algorithm
  1. Find (Shi-Tomasi) corners in first frame and initialize tracks
  2. Track from frame to frame
  3. Delete track if error exceeds threshold
  4. Initialize additional tracks when necessary
  5. Repeat step 2-4

KLT Tracker

- KLT tracker is highly efficient (real-time on CPU) but provides only sparse motion vectors
- Dense optical flow methods require GPU

Demo of a KLT Tracker

Visual Servoing Platform (ViSP), 2010. http://www.irisa.fr/lagadic/visp/visp.html
https://www.youtube.com/watch?v=a0B2nBj4FAM

Lessons Learned

- 2D motion estimation
- Cost functions
- Optical computer mouse
- Corner detectors
- KLT Tracker

- Next: Visual odometry