Visual Navigation for Flying Robots

Visual Motion Estimation

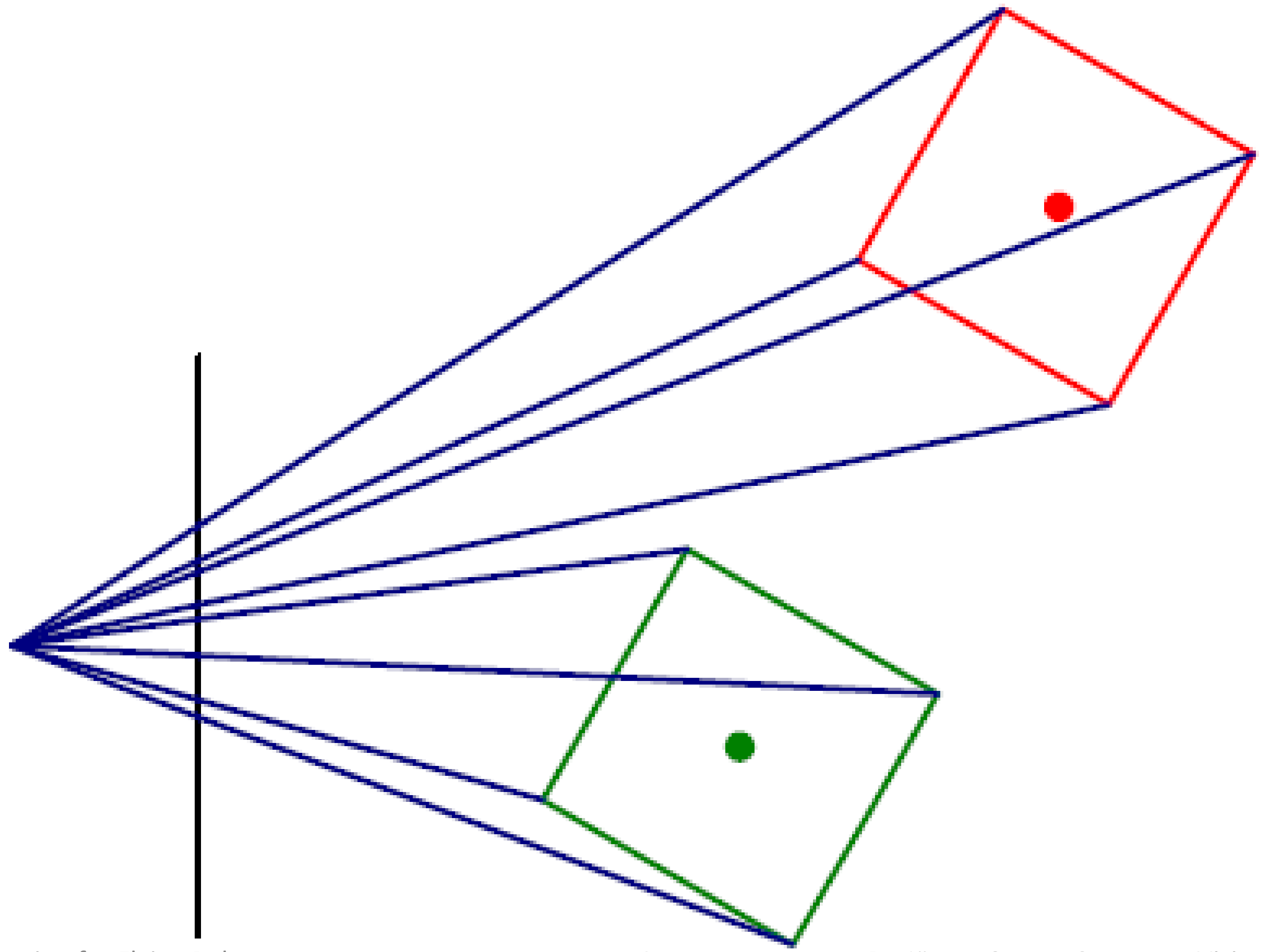
Dr. Jürgen Sturm

Motivation

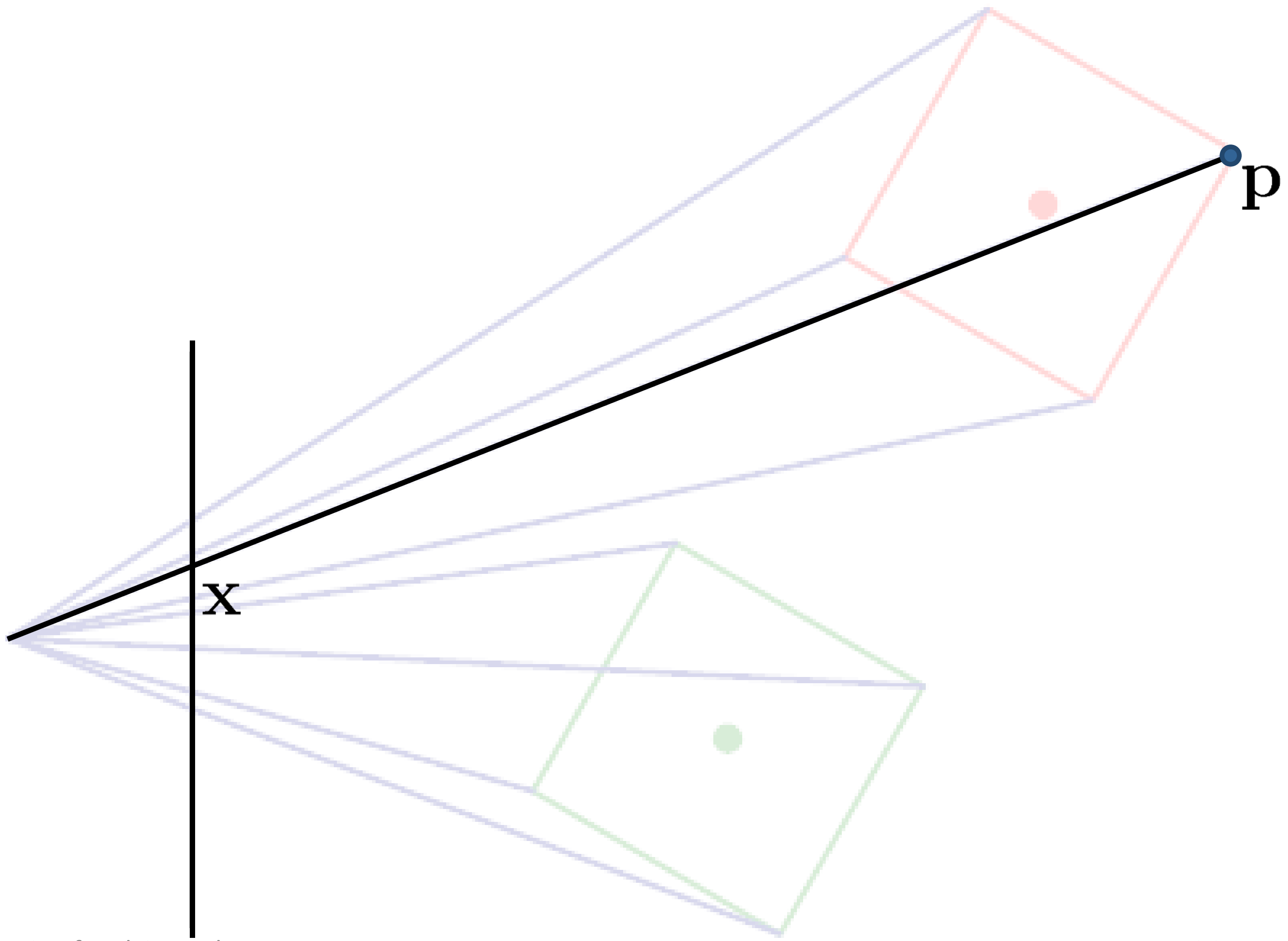
Visual Motion Estimation

- Quick geometry recap
- Image filters
- 2D image alignment
- Corner detectors
- Kanade-Lucas-Tomasi tracker
- 2D motion estimation
- Interesting research papers from ICRA and ROSCon

Recap: Perspective Projection



Recap: Perspective Projection



3D to 2D Perspective Projection

- 3D point $\mathbf{p}$ (in the camera frame)
- 2D point $\mathbf{x}$ (on the image plane)
- Pin-hole camera model

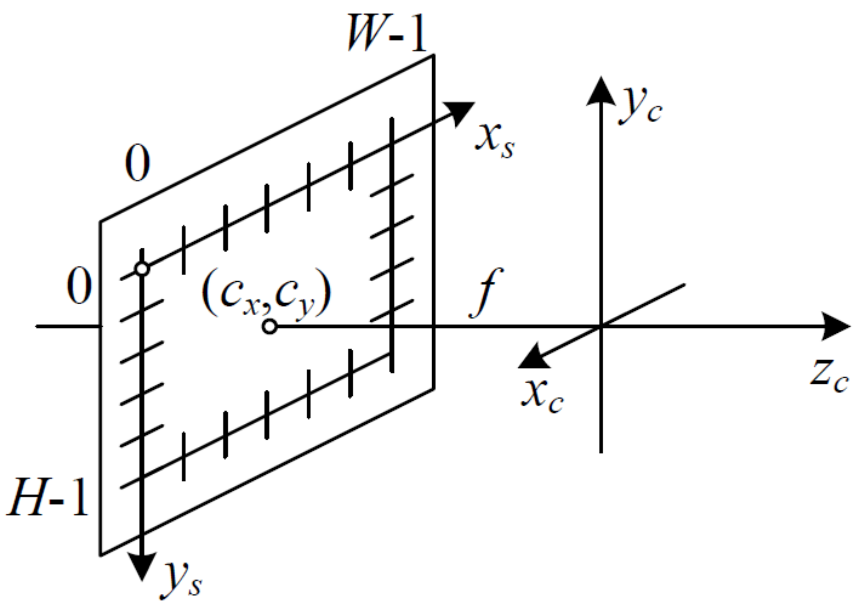
$$\tilde{\mathbf{x}} = \lambda \bar{\mathbf{x}} = \mathbf{p}$$

- Remember, $\tilde{\mathbf{x}}$ is homogeneous, need to normalize

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \quad \Rightarrow \quad \mathbf{x} = \begin{pmatrix} \tilde{x}/\tilde{z} \\ \tilde{y}/\tilde{z} \end{pmatrix}$$

Camera Intrinsics

- So far, 2D point is given in meters on image plane (located in 1m distance from origin)
- But: we want 2D point be measured in pixels (as the sensor does)



Camera Intrinsics

- Need to apply some scaling/offset

$$\tilde{\mathbf{x}} = \underbrace{\begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{\text{intrinsics } K} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\text{projection}} \tilde{\mathbf{p}}$$

- Focal length f_x, f_y
- Camera center c_x, c_y
- Skew s

Image Plane

- Pixel coordinates $\mathbf{x} \in \Omega$
- Image plane $\Omega \subset \mathbb{R}^2$

- Example:
  - Discrete case $\mathbf{x} \in [0, W-1] \times [0, H-1] \subset \mathbb{N}_0^2$ (default in this course)
  - Continuous case $\mathbf{x} \in [0,1] \times [0,1] \subset \mathbb{R}^2$

Image Functions

- We can think of an image as a function $f : \Omega \mapsto \mathbb{R}$
- $f(\mathbf{x})$ gives the intensity at position $\mathbf{x}$
- Color images are vector-valued functions

$$f : \Omega \mapsto \mathbb{R}^3$$

$$f(\mathbf{x}) = \begin{pmatrix} r(\mathbf{x}) \\ g(\mathbf{x}) \\ b(\mathbf{x}) \end{pmatrix}$$

Image Functions

- Realistically, the image function is only defined on a rectangle and has finite range

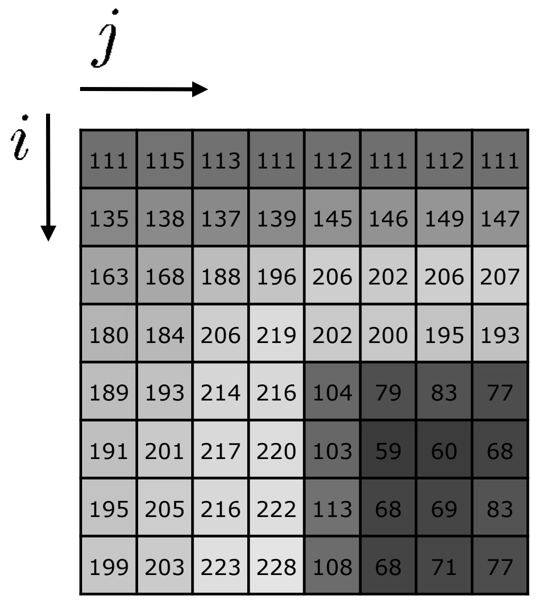
$$f : [0, W-1] \times [0, H-1] \mapsto [0, 1]$$

- Image can be represented as a matrix
- Alternative notations

$$F_{ij}, f(\underset{\uparrow}{i}, \underset{\uparrow}{j}), f(\underset{\uparrow}{x}, \underset{\uparrow}{y}), f(\mathbf{x}), \ldots$$

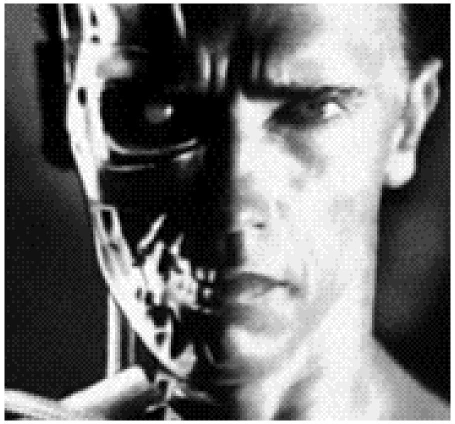
often (row,column)

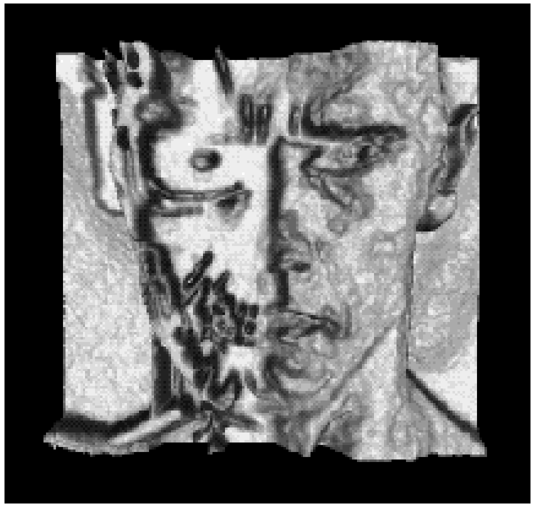
often (column,row)

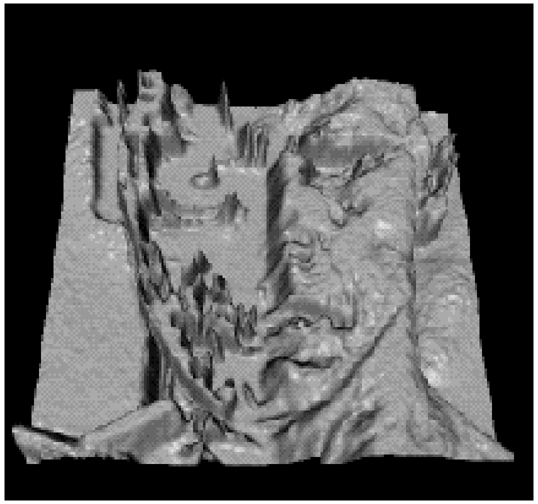


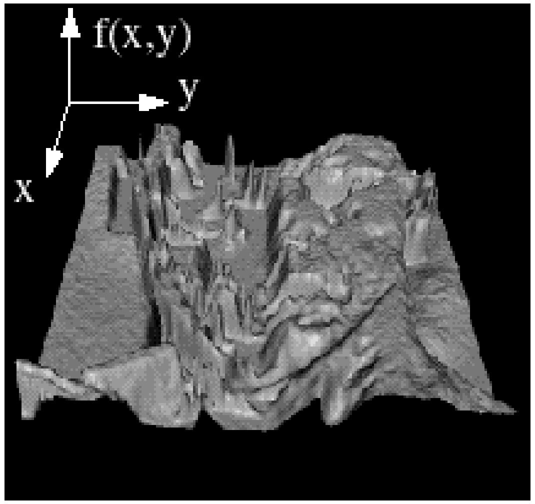
111	115	113	111	112	111	112	111
135	138	137	139	145	146	149	147
163	168	188	196	206	202	206	207
180	184	206	219	202	200	195	193
189	193	214	216	104	79	83	77
191	201	217	220	103	59	60	68
195	205	216	222	113	68	69	83
199	203	223	228	108	68	71	77

Example









Digital Images

- Light intensity is sampled by CCD/CMOS sensor on a regular grid
- Electric charge of each cell is quantized and gamma compressed (for historical reasons)

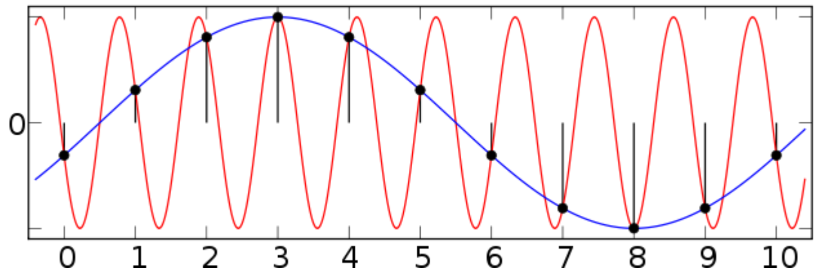
$$V = B^{\frac{1}{\gamma}} \quad \text{with} \quad \gamma = 2.2$$

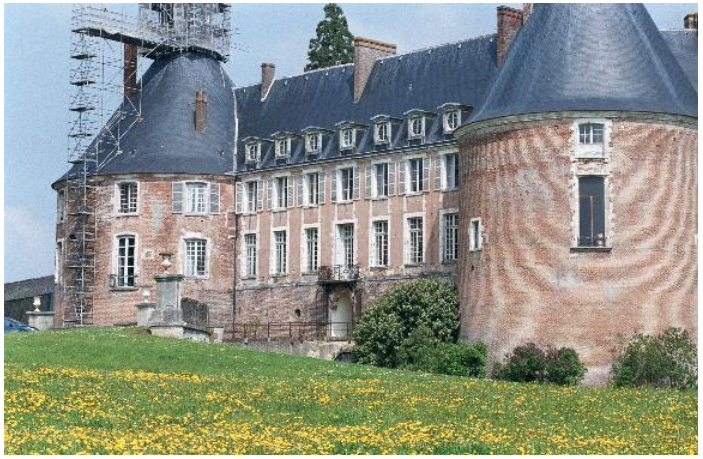
- CRTs / monitors do the inverse $B = V^{\gamma}$
- Almost all images are gamma compressed

→ Double brightness results only in a 37% higher intensity value (!)

Aliasing

- High frequencies in the scene and a small fill factor on the chip can lead to (visually) unpleasing effects
- Examples





Rolling Shutter

- Most CMOS sensors have a rolling shutter
- Rows are read out sequentially
- Sensitive to camera and object motion
- Can we correct for this?

Image Filtering

- We want to remove unwanted sources of variation, and keep the information relevant for whatever task we need to solve

$$\begin{tikzcd} {} \arrow[r, "f(i,j)"] & \boxed{\quad\quad\quad} \arrow[r, "g(i,j)"] & {} \end{tikzcd}$$

- Example tasks:
  de-noising, (de-)blurring, computing derivatives, edge detection, …

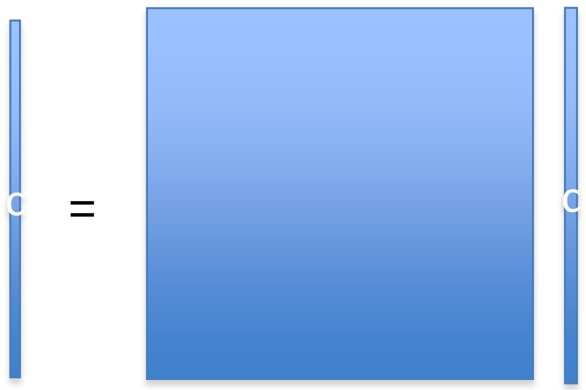
Linear Filtering

- Each output is a linear combination of all the input values

$$g(i,j) = \sum_{k,l} h(i,j,k,l) f(k,l)$$

- In matrix form

G = H F



Spatially Invariant Filtering

- We are often interested in spatially invariant operations

$$g(i,j) = f * h = \sum_{k,l} h(i-k, j-l) f(k,l)$$

- Example

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

$*$

-1	2	-1
-1	2	-1
-1	2	-1

$=$

?

Spatially Invariant Filtering

- We are often interested in spatially invariant operations

$$g(i,j) = f * h = \sum_{k,l} h(i-k, j-l) f(k,l)$$

- Example

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

$*$

-1	2	-1
-1	2	-1
-1	2	-1

$=$

?	?	?	?	?	?	?	?
?	-5	9	-9	21	-12	10	?
?	-29	18	24	4	-7	5	?
?	-50	40	142	-88	-34	10	?
?	-41	41	264	-175	-71	0	?
?	-24	37	349	-224	-120	-10	?
?	-23	33	360	-217	-134	-23	?
?	?	?	?	?	?	?	?

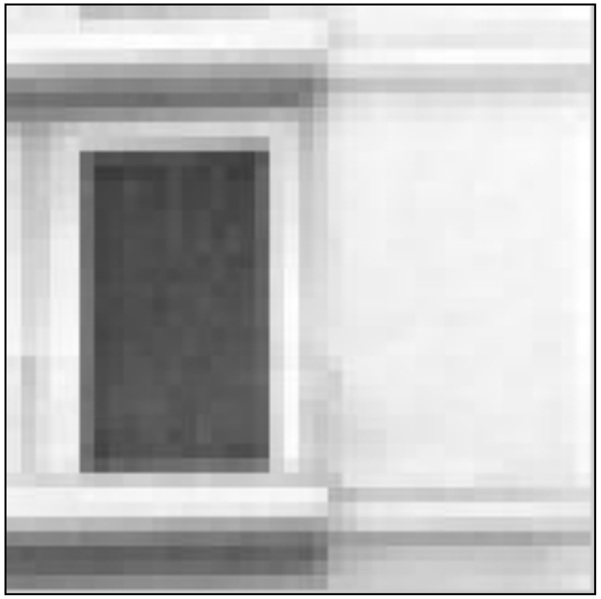
Important Filters

- **Impulses**
- **Shifts**
- Blurring and de-blurring
  - **Gaussian**
  - Bilateral filter
  - Motion blur
- Edges
  - **Finite difference filter**
  - Derivative filter
  - Oriented filters
  - Gabor filter
- …

Impulse

$$g(i,j) = f \underset{\uparrow}{*} h = \sum_{k,l} h(i-k, j-l) f(k,l)$$

convolution operator



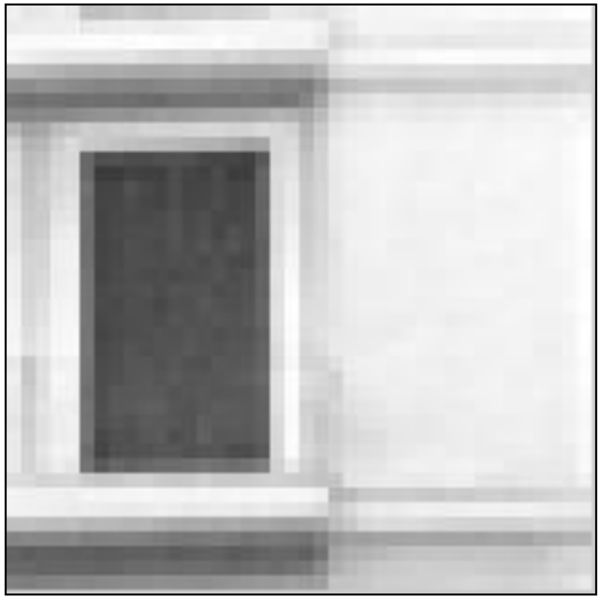
$f(i,j)$

$*$

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

$h(i,j)$

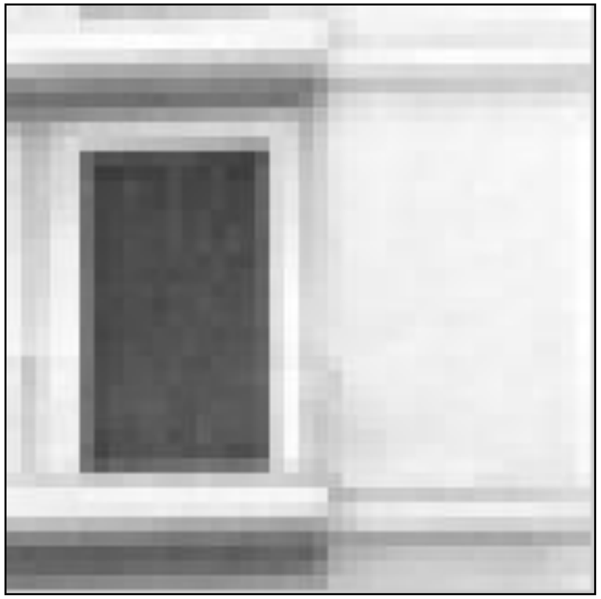
$=$



$g(i,j)$

Image shift (translation)

$$g(i,j) = f * h = \sum_{k,l} h(i-k, j-l) f(k,l)$$



$f(i,j)$

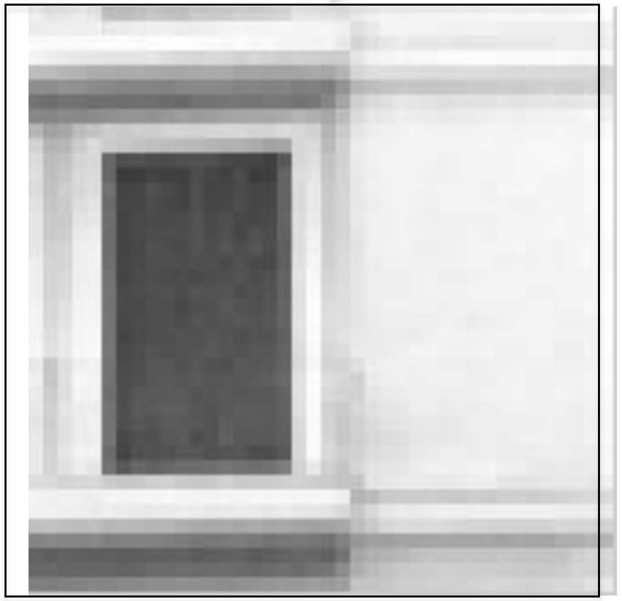
$*$

$$\begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 0&0&0&0&1\\\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline\end{array}$$

$h(i,j)$

$=$

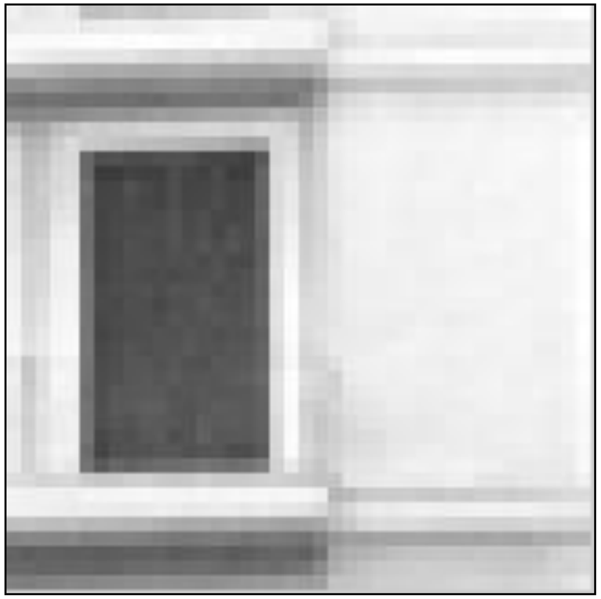
2 pixels $\rightarrow$



$g(i,j)$

Image rotation

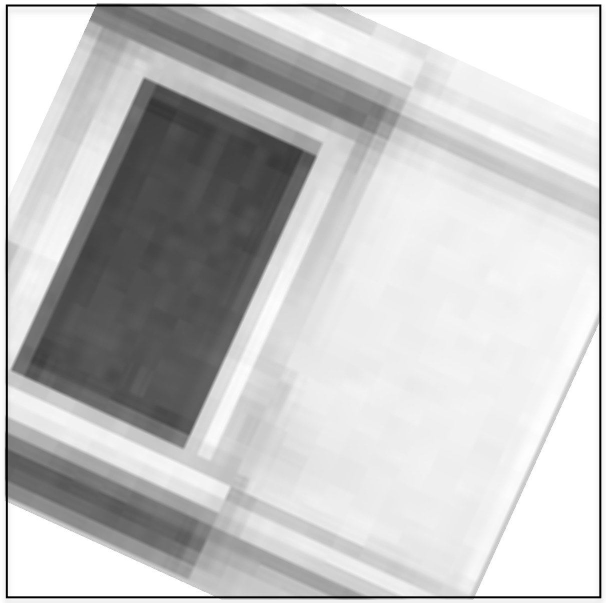
$$g(i,j) = f * h = \sum_{k,l} h(i-k, j-l) f(k,l)$$



$f(i,j)$

$*$? $=$

$h(i,j)$

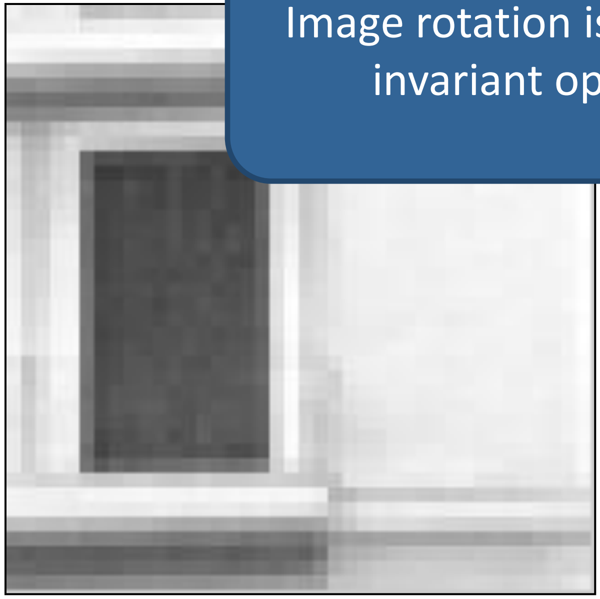


$g(i,j)$

Image rotation

$$g(i,j) = f * h = \sum_{k,l} h(i-k, j-l) f(k,l)$$

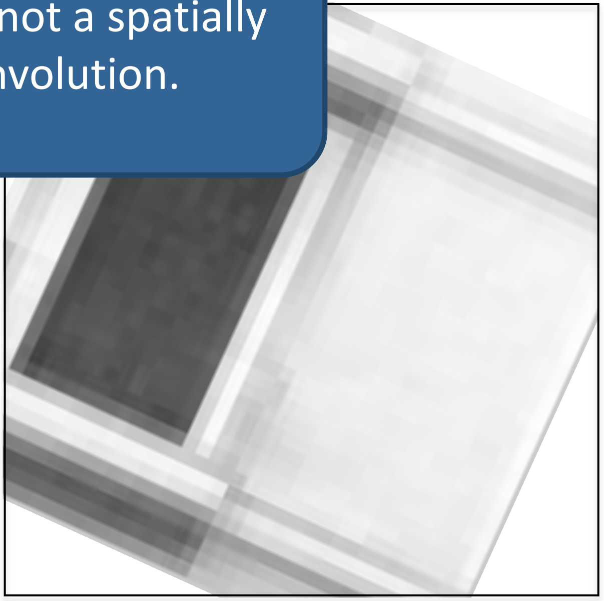
Image rotation is a linear operator (why?), but not a spatially invariant operation (why?). There is no convolution.



$f(i,j)$

$*$? $=$

$h(i,j)$



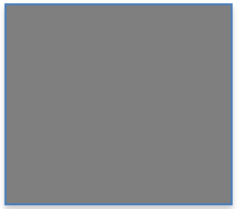
$g(i,j)$

Rectangular Filter

$$g(i,j) = f * h = \sum_{k,l} h(i-k, j-l) f(k,l)$$

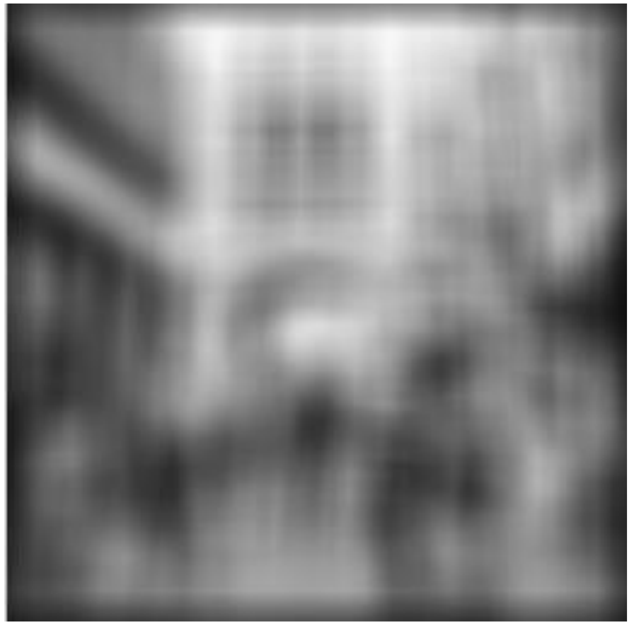
$f(i,j)$

$*$



$h(i,j)$

$=$



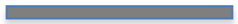
$g(i,j)$

Rectangular Filter

$$g(i,j) = f * h = \sum_{k,l} h(i-k, j-l) f(k,l)$$

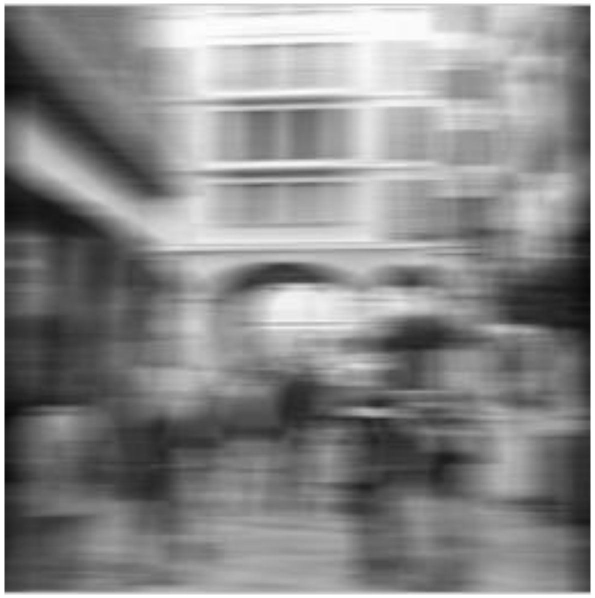
$f(i,j)$

$*$



$h(i,j)$

$=$



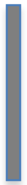
$g(i,j)$

Rectangular Filter

$$g(i,j) = f * h = \sum_{k,l} h(i-k, j-l) f(k,l)$$

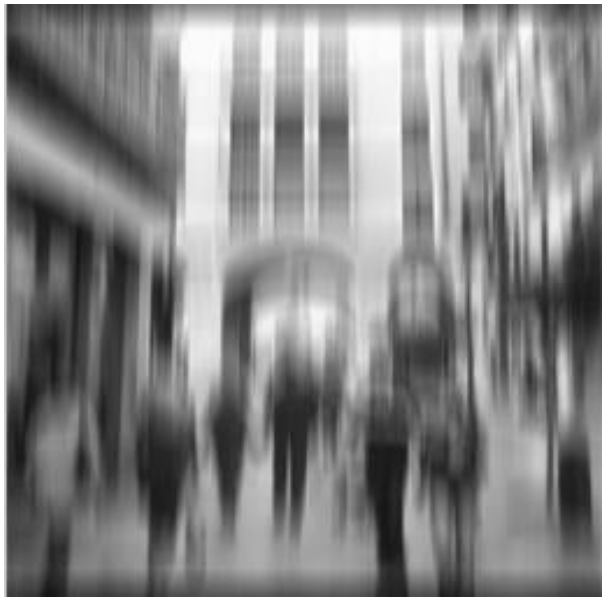
$f(i,j)$

$*$



$h(i,j)$

$=$



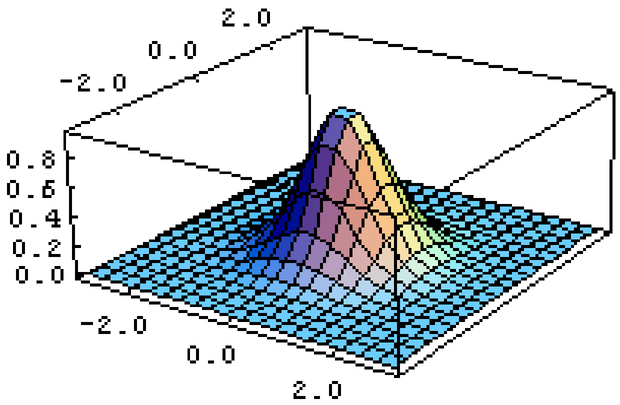
$g(i,j)$

Gaussian Blur

- Gaussian distribution

$$g_\sigma(i, i) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2 + j^2}{2\sigma^2}\right)$$

- Example of resulting kernel



Gaussian Blur

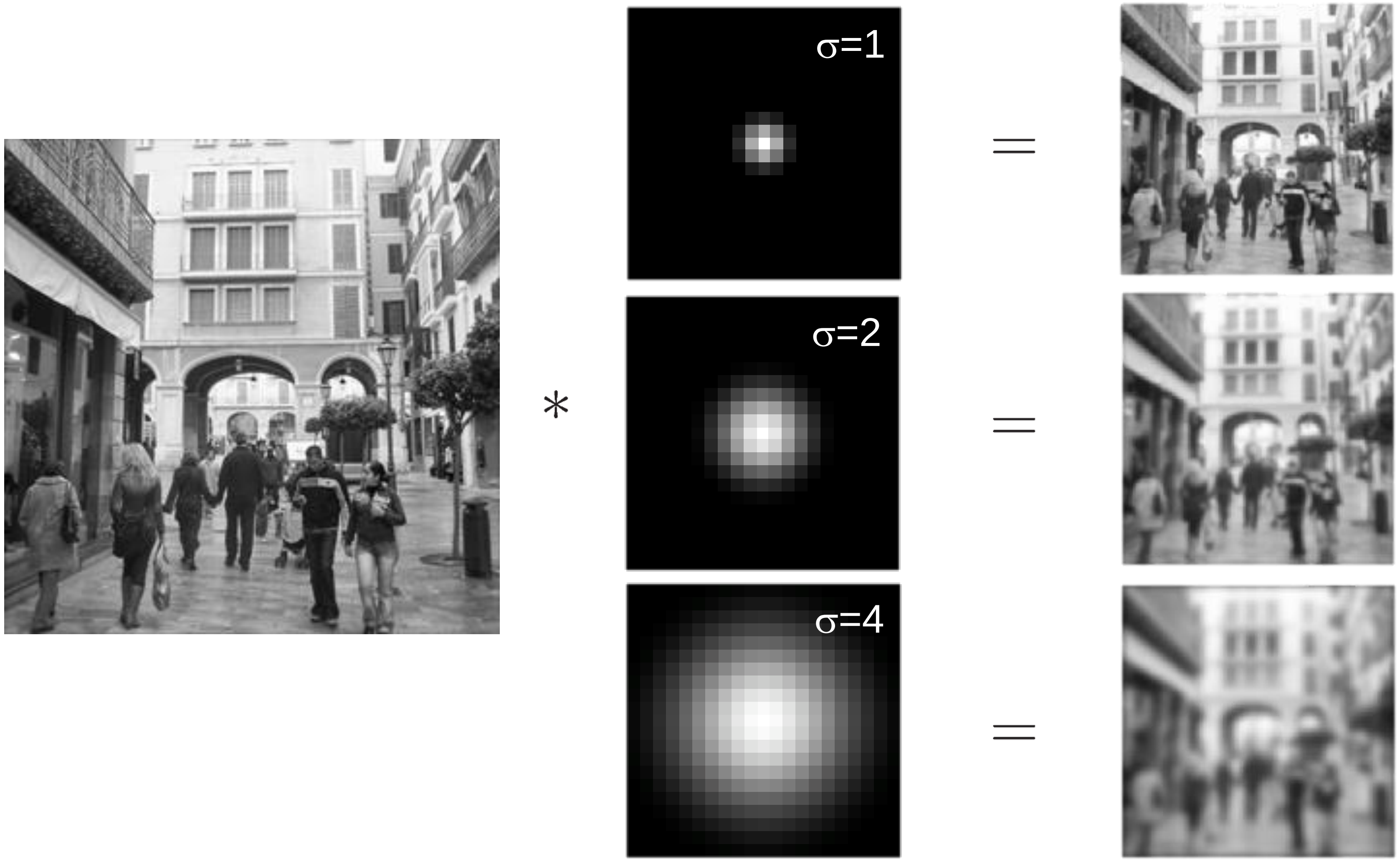
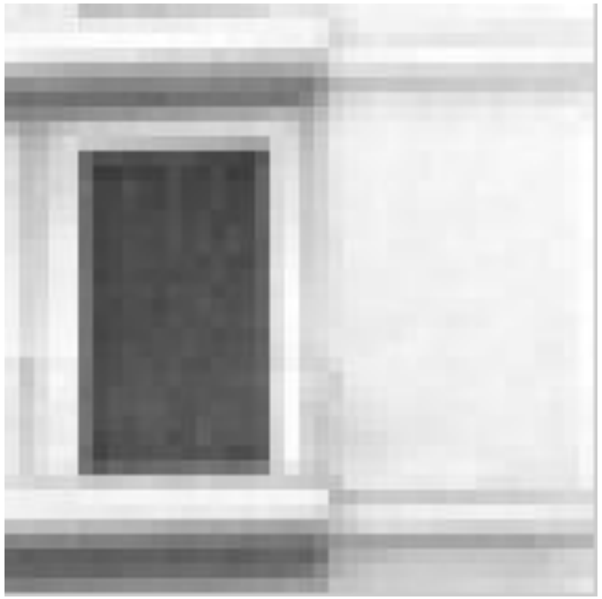


Image Gradient

- The image gradient $\nabla f = \left(\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y} \right)^\top$ points in the direction of increasing intensity (steepest ascend)



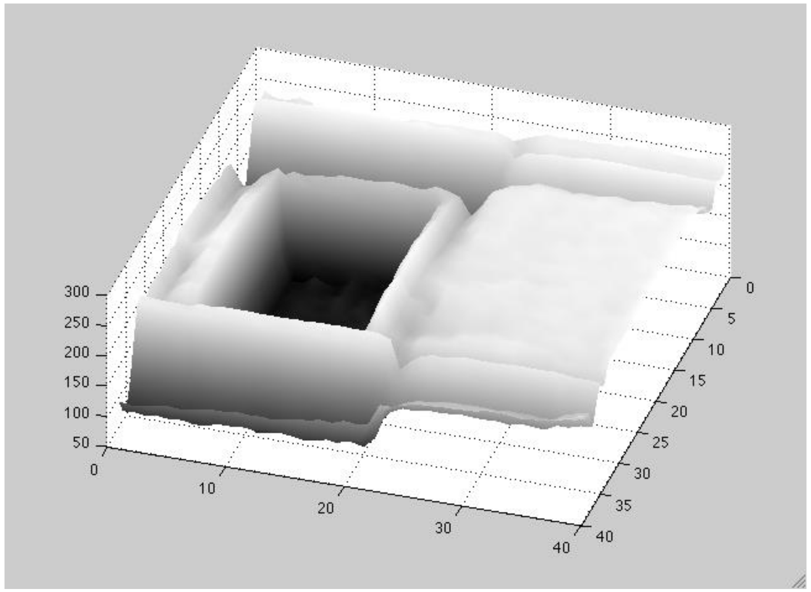
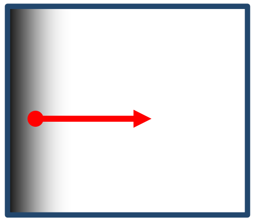
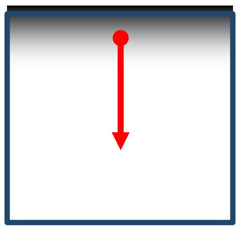
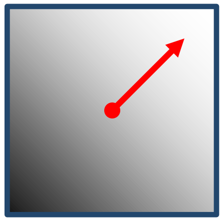


Image Gradient

- The image gradient $\nabla f = \left(\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y}\right)^\top$ points in the direction of increasing intensity (steepest ascend)







$$\nabla f = \left(\frac{\partial f}{\partial x}, 0\right)^\top \quad \nabla f = \left(0, \frac{\partial f}{\partial y}\right)^\top \quad \nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)^\top$$

Image Gradient

- Gradient direction (related to edge orientation)

$$\theta = \operatorname{atan2}\left(\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}\right)$$

- Gradient magnitude (edge strength)

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Image Gradient

How can we differentiate a digital image $f(x, y)$?

- Option 1: Reconstruct a continuous image, then take gradient
- **Option 2: Take discrete derivative (finite difference filter)**
- Option 3: Convolve with derived Gaussian (derivative filter)

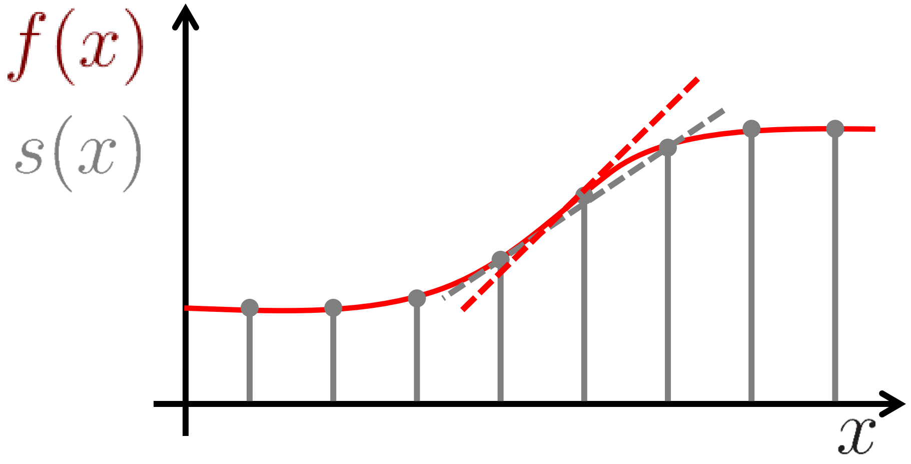
Finite difference

- First-order central difference

$$\frac{\partial f}{\partial x}(x,y) \approx \frac{f(x+1,y) - f(x-1,y)}{2}$$

- Corresponding convolution kernel:

-.5	0	.5



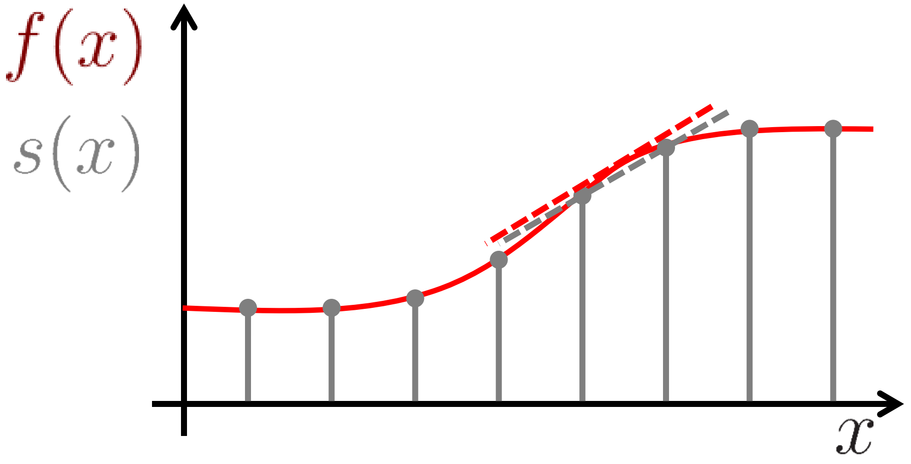
Finite difference

- First-order central difference (half pixel)

$$\frac{\partial f}{\partial x}(x,y) \approx f(x+0.5,y) - f(x-0.5,y)$$

- Corresponding convolution kernel:

-1	1



Second-order Derivative

- Differentiate again to get second-order central difference

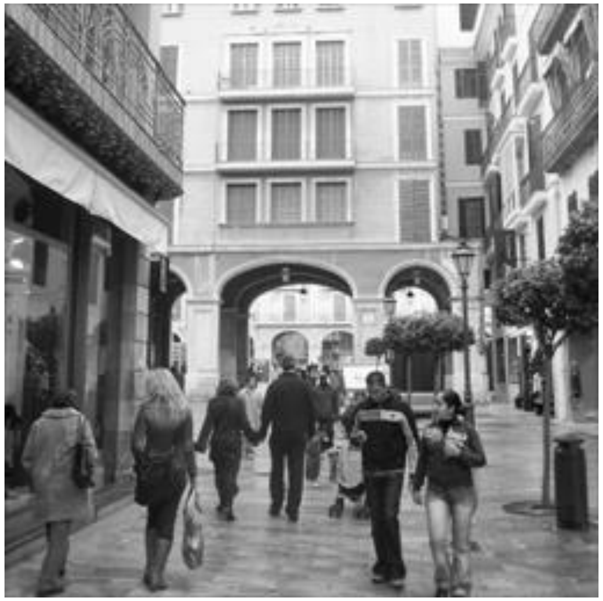
$$\frac{\partial f(x)}{\partial x^2} \approx f(x+1) - 2f(x) + f(x-1)$$

Corresponding convolution kernel:

1	-2	1

Example

$$g(i,j) = f * h = \sum_{k,l} h(i-k, j-l) f(k,l)$$



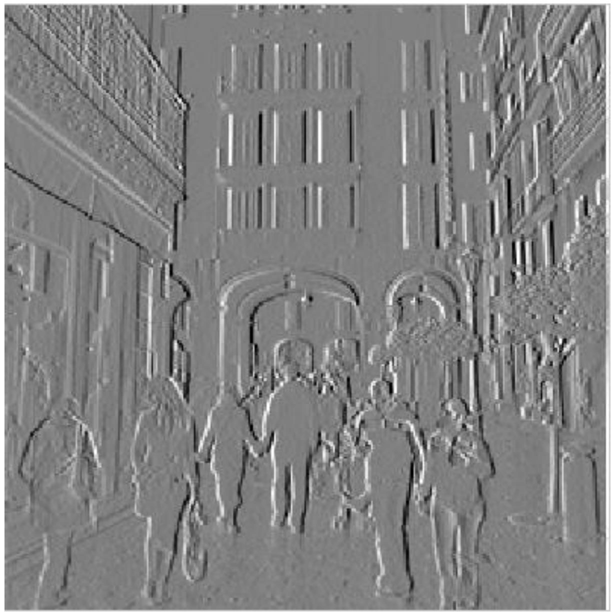
$f(i,j)$

$*$

-1	1

$h(i,j)$

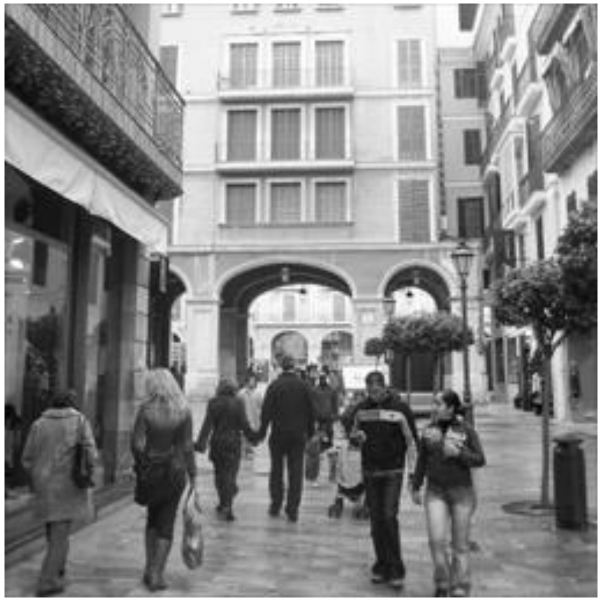
$=$



$g(i,j)$

Example

$$g(i,j) = f * h = \sum_{k,l} h(i-k, j-l) f(k,l)$$



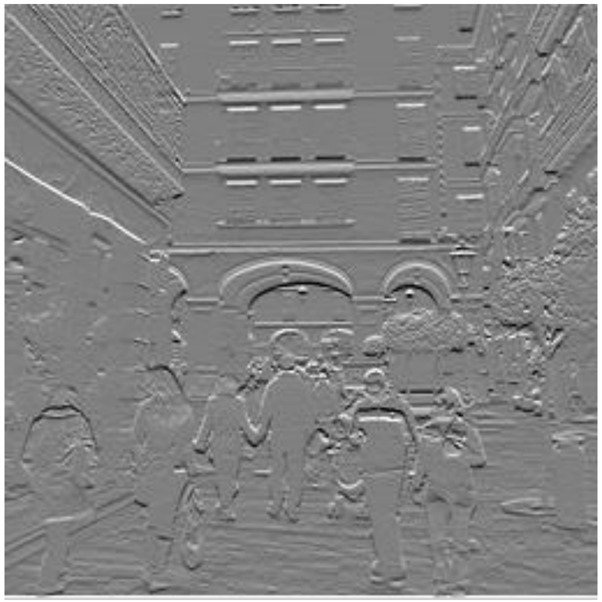
$f(i,j)$

$*$

$$\begin{array}{|c|}\hline -1 \\ \hline 1 \\ \hline \end{array}$$

$h(i,j)$

$=$



$g(i,j)$

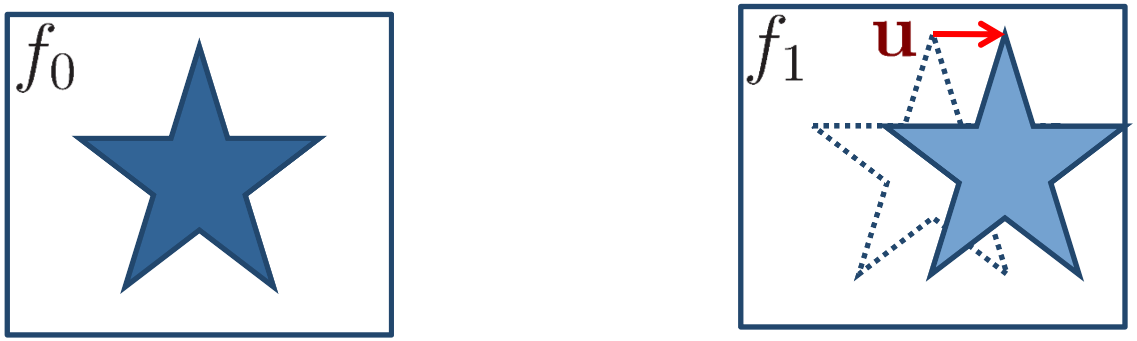
(Dense) Motion Estimation

- 2D motion

- 3D motion

Problem Statement

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

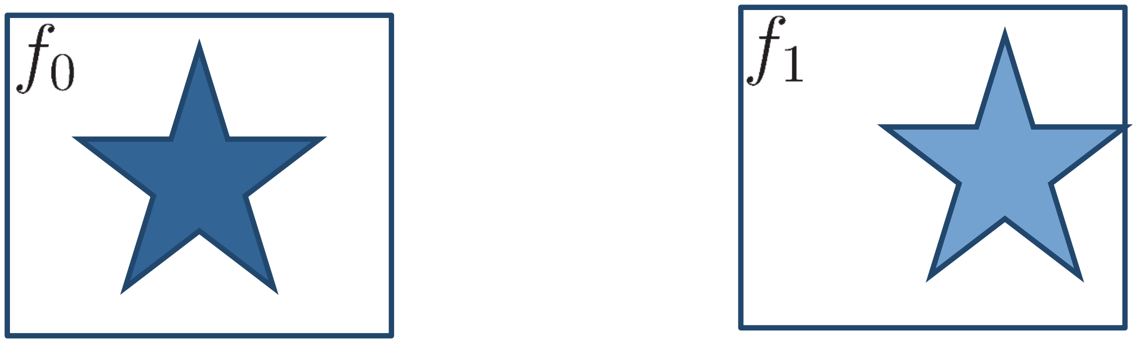


- For the moment, let's assume that the camera only moves in the xy-plane, i.e., $\mathbf{u} = (u\ v)^\top$
- Extension to 3D follows

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)

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

with displacement $\mathbf{u} = (u \ v)^\top$
and residual errors $e_i = f_1(\mathbf{x}_i + \mathbf{u}) - f_0(\mathbf{x}_i)$

Robust Error Metrics

- SSD metric is sensitive to outliers
- Solution: apply a (more) robust error metric

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

Robust Error Metrics

- Sum of absolute differences (SAD, L1 norm)

$$\rho_{\text{SAD}}(e) = |e|$$

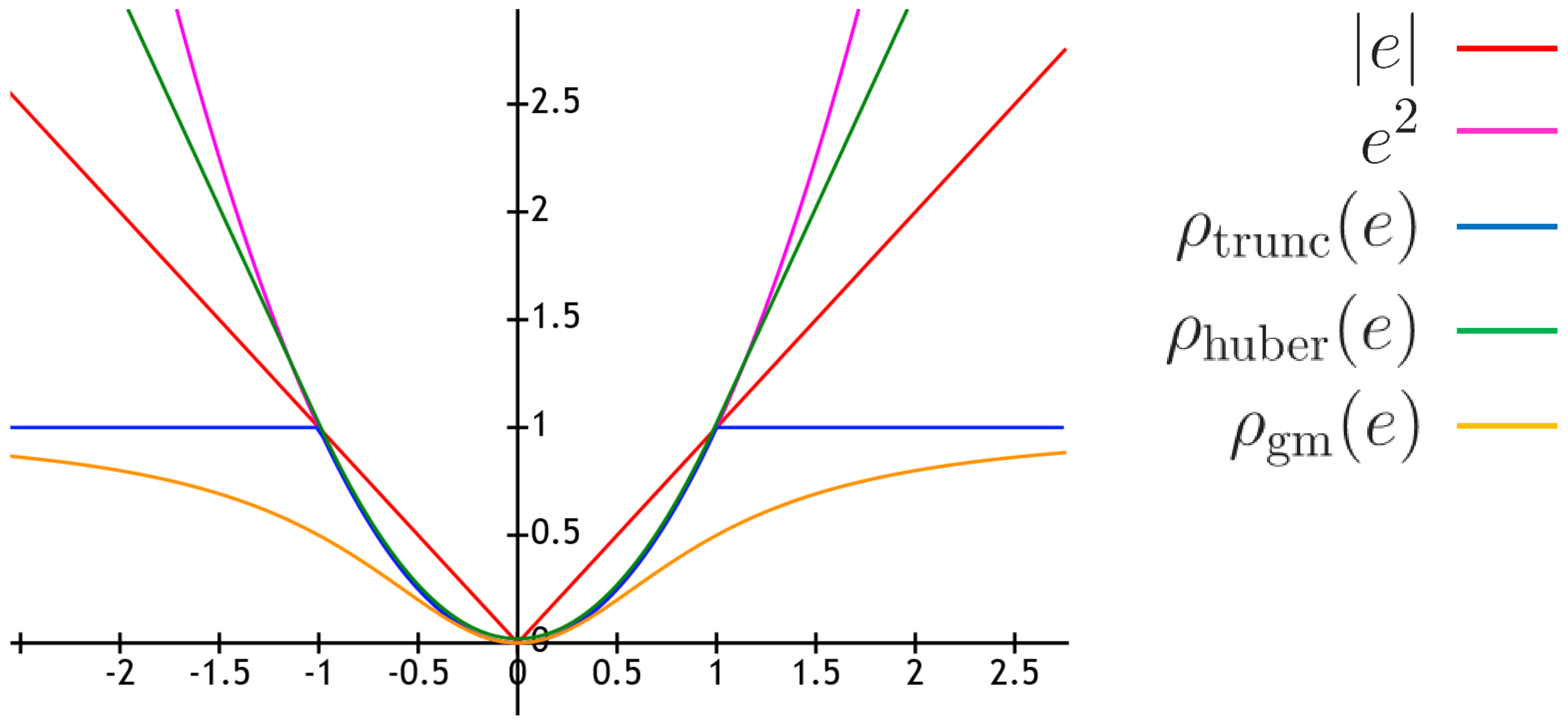
- Sum of truncated errors

$$\rho_{\text{trunc}}(e) = \begin{cases} e^2 & \text{if } |e| < b \\ b^2 & \text{otherwise} \end{cases}$$

- Geman-McClure

$$\rho_{\text{gm}}(e) = \frac{x^2}{1 + x^2}$$

Robust Error Metrics



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

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

Exposure Differences

- Images might be taken with different exposure (auto shutter, white balance, …)
- Bias and gain model

$$f_1(\mathbf{x}+\mathbf{u}) = (1+\alpha) f_0(\mathbf{x}) + \beta$$

- With SSD we get

$$E_{\mathrm{BG}}(\mathbf{u}) = \sum_i \left(f_1(\mathbf{x}_i+\mathbf{u}) - (1+\alpha) f_0(\mathbf{x}_i) + \beta\right)^2$$
$$= \sum_i \alpha f_0(\mathbf{x}) + \beta - e_i^2$$

Cross-Correlation

- Maximize the product (instead of minimizing the differences)

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

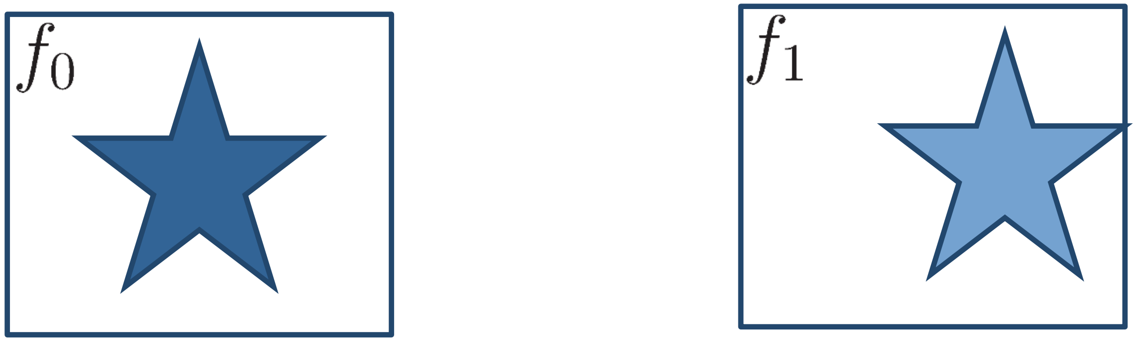
- Normalized cross-correlation (between -1..1)

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

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**

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



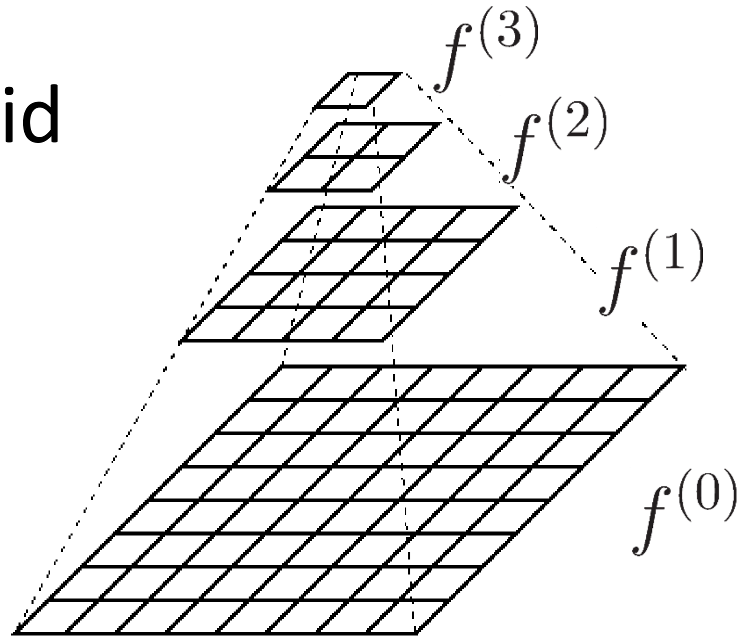
Finding the minimum

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

Hierarchical motion estimation

- Construct image pyramid

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



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

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

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

- Taylor expansion of energy function

$$
\begin{aligned}
E_{\mathrm{LK-SSD}}(\mathbf{u}+\Delta\mathbf{u}) &= \sum_i (f_1(\mathbf{x}_i+\mathbf{u}+\Delta\mathbf{u}) - f_0(\mathbf{x}_i))^2 \\
&\approx \sum_i (f_1(\mathbf{x}_i+\mathbf{u}) + J_1(\mathbf{x}+\mathbf{u})\Delta\mathbf{u} - f_0(\mathbf{x}_i))^2 \\
&= \sum_i (J_1(\mathbf{x}+\mathbf{u})\Delta\mathbf{u} + e_i)^2
\end{aligned}
$$

with $J_1(\mathbf{x}_i+\mathbf{u}) = \nabla f_1(\mathbf{x}_i+\mathbf{u}) = (\frac{\partial f_1}{\partial x}, \frac{\partial f_1}{\partial y})(\mathbf{x}_i+\mathbf{u})$

Least Squares Minimization

- Goal: Minimize

$$E(\mathbf{u}+\Delta\mathbf{u}) = \sum_i (J_1(\mathbf{x}_i+\mathbf{u})\Delta\mathbf{u} + e_i)^2$$

- Solution: Compute derivative and set to zero

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

with $A = \sum_i J_1^\top(\mathbf{x}_i+\mathbf{u}) J_1(\mathbf{x}+\mathbf{u})$

and $\mathbf{b} = \sum_i e_i J_1^\top(\mathbf{x}_i+\mathbf{u})$

Least Squares Minimization

1. Compute A,b from image gradients using

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

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

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

2. Solve $A\Delta\mathbf{u} = -\mathbf{b}$

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

All of these computation are super fast!

Covariance of the Estimated Motion

- Assuming (small) Gaussian noise in the images

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

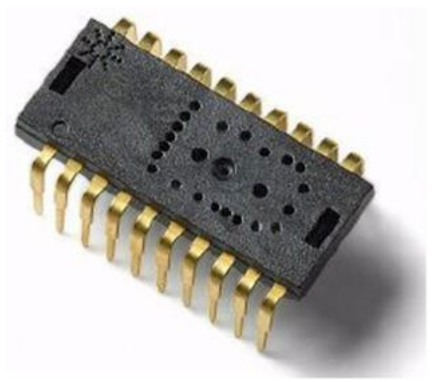
with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$

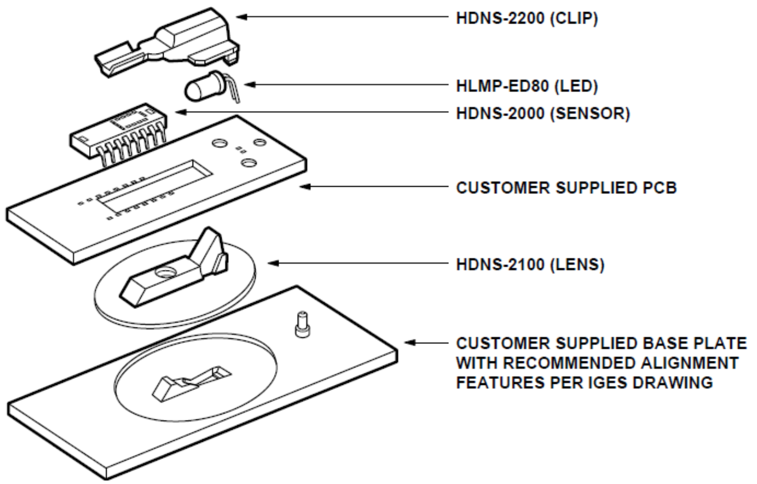
- … results in uncertainty in the motion estimate with covariance (e.g., useful for Kalman filter)

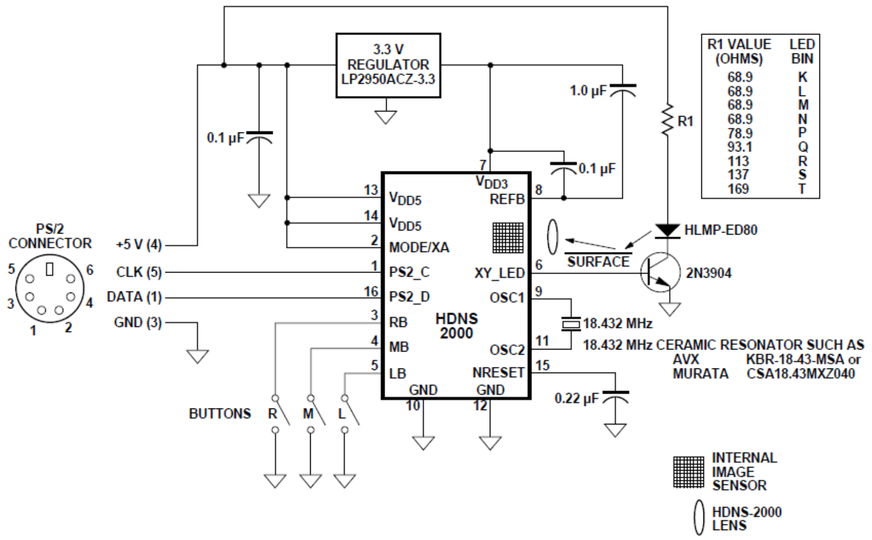
$$\Sigma_u = \sigma^2 A^{-1}$$

Optical Computer Mouse (since 1999)

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







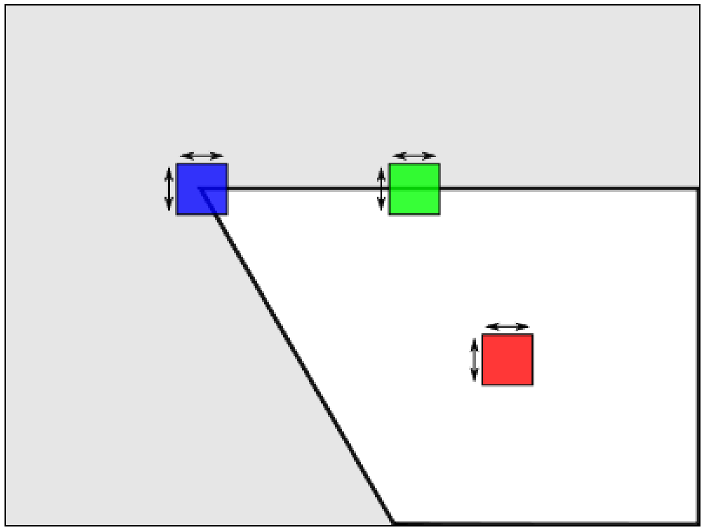
R1 VALUE (OHMS)	LED BIN
68.9	K
68.9	L
68.9	M
68.9	N
78.9	P
93.1	Q
113	R
137	S
169	T

Image Patches

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

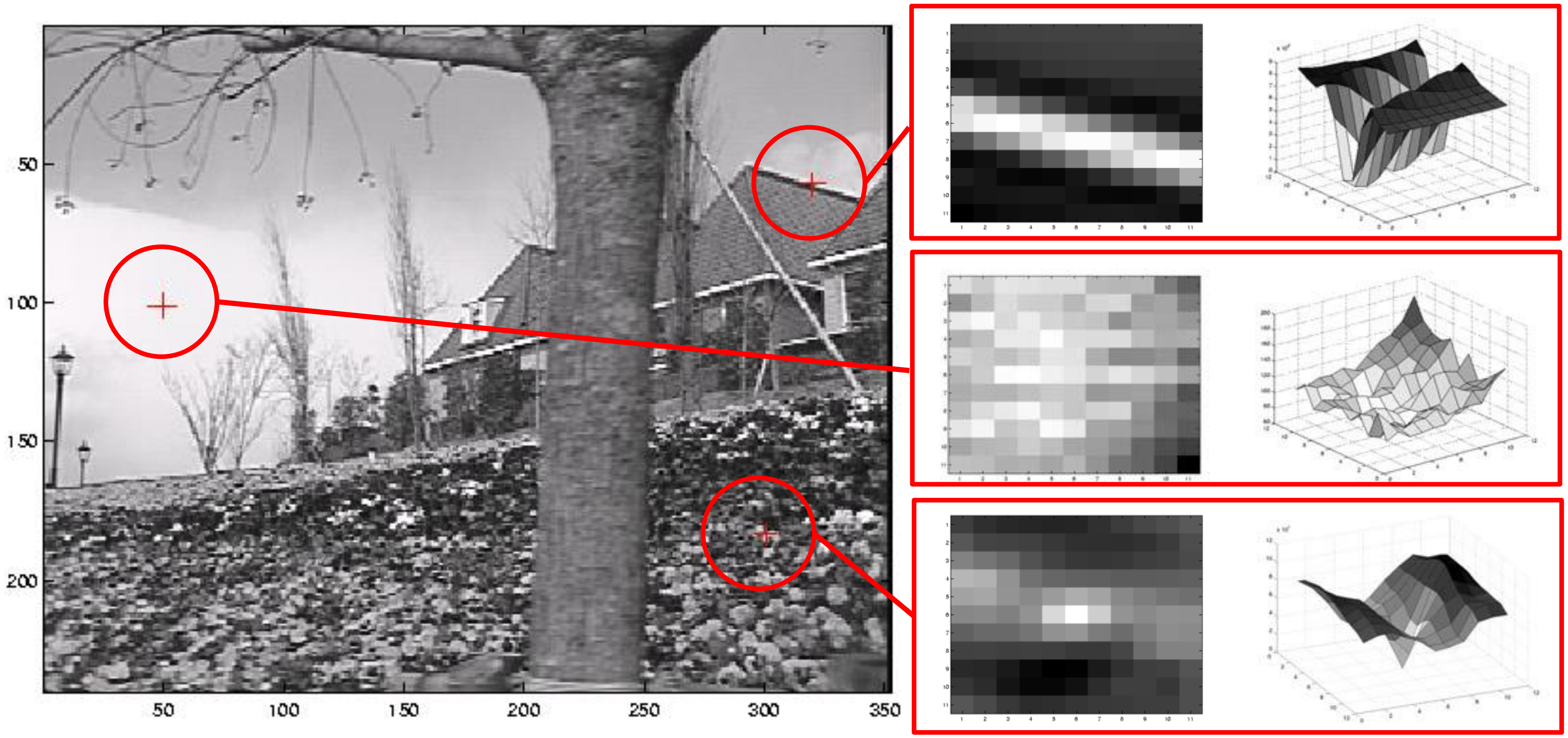
Image Patches

- 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



Corner Detection

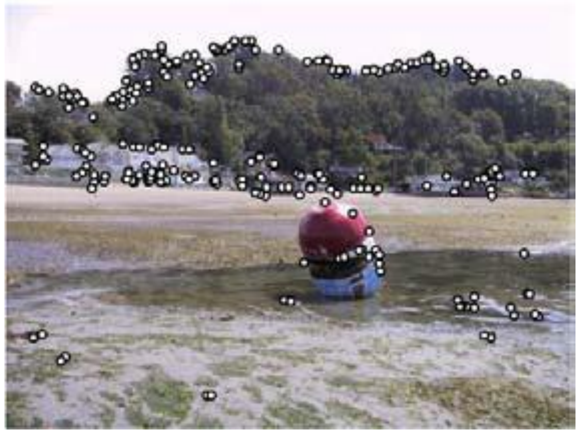
$$A = \begin{pmatrix} \sum f_x^2 & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^2 \end{pmatrix}$$

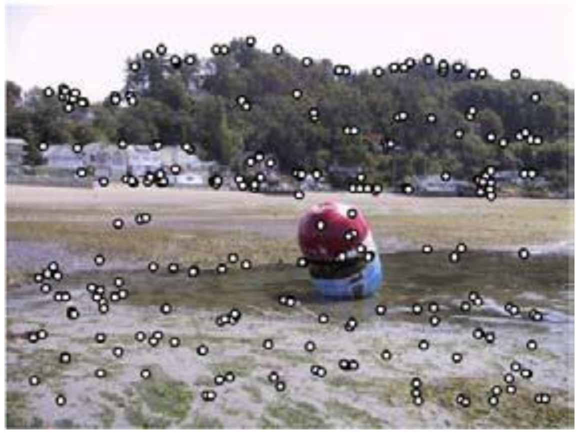
- **Idea:** Inspect eigenvalues λ_1, λ_2 of Hessian A
  - λ_1, λ_2 small → no point of interest
  - λ_1 large, λ_2 small → edge
  - λ_1, λ_2 large → corner
- Harris detector (does not need eigenvalues)

  $$\lambda_1 \lambda_2 > \kappa (\lambda_1 + \lambda_2)^2 \Leftrightarrow \det(A) > \kappa \operatorname{trace}^2(A)$$
- Shi-Tomasi (or Kanade-Lucas) $\min(\lambda_1, \lambda_2) > \kappa$

Corner Detection

1. For all pixels, computer corner strength
2. Non-maximal suppression
   (E.g., sort by strength, strong corner suppresses weaker corners in circle of radius *r*)


strongest responses


non-maximal suppression

Other Detectors

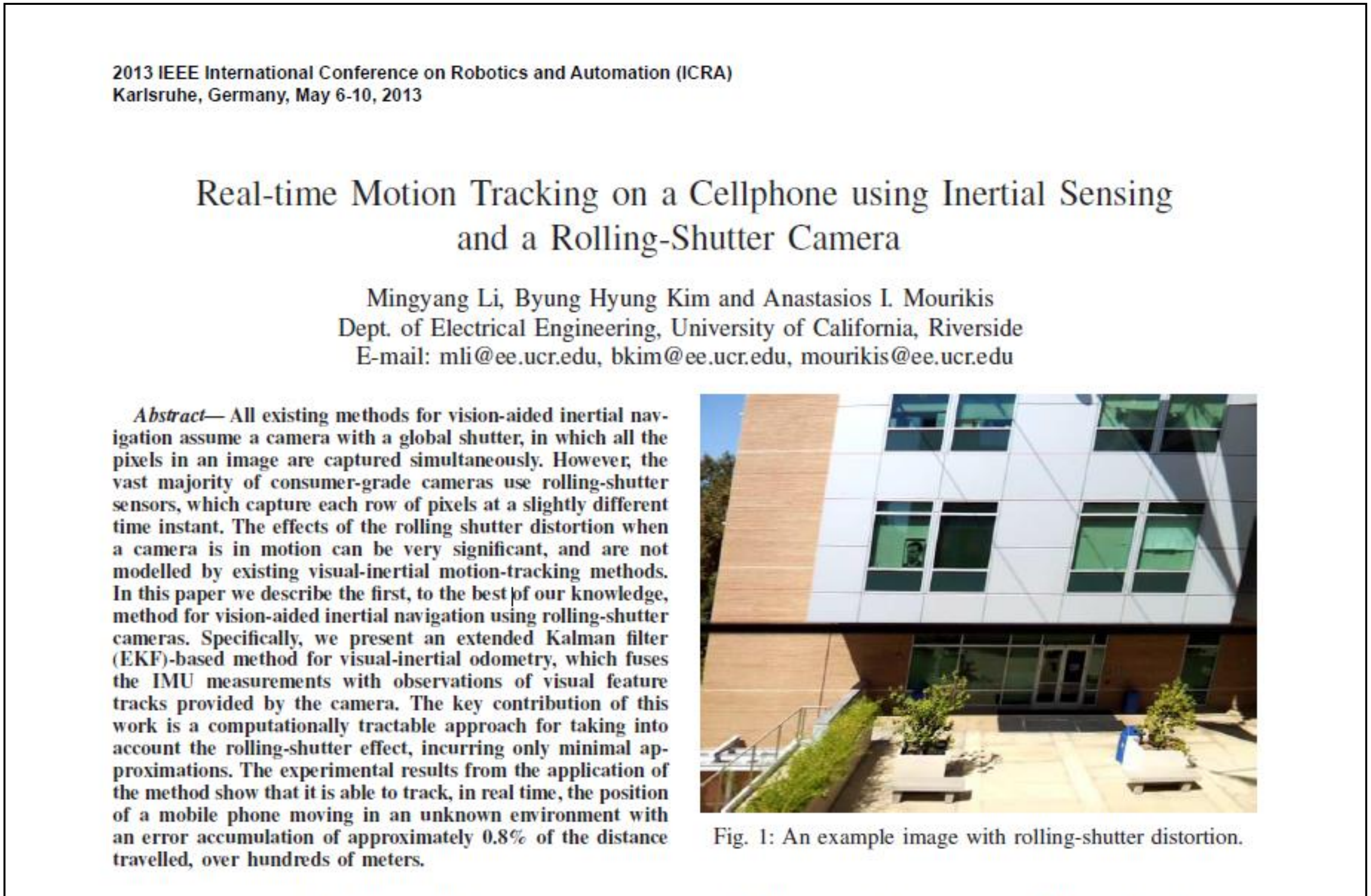
- Förstner detector (localize corner with sub-pixel accuracy)
- FAST corners (learn decision tree, minimize number of tests → super fast)
- Difference of Gaussians / DoG (scale-invariant detector)
- …

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

Example

Interesting Papers from ICRA 2013



2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Real-time Motion Tracking on a Cellphone using Inertial Sensing and a Rolling-Shutter Camera

Mingyang Li, Byung Hyung Kim and Anastasios I. Mourikis
Dept. of Electrical Engineering, University of California, Riverside
E-mail: mli@ee.ucr.edu, bkim@ee.ucr.edu, mourikis@ee.ucr.edu

Abstract— All existing methods for vision-aided inertial navigation assume a camera with a global shutter, in which all the pixels in an image are captured simultaneously. However, the vast majority of consumer-grade cameras use rolling-shutter sensors, which capture each row of pixels at a slightly different time instant. The effects of the rolling shutter distortion when a camera is in motion can be very significant, and are not modelled by existing visual-inertial motion-tracking methods. In this paper we describe the first, to the best of our knowledge, method for vision-aided inertial navigation using rolling-shutter cameras. Specifically, we present an extended Kalman filter (EKF)-based method for visual-inertial odometry, which fuses the IMU measurements with observations of visual feature tracks provided by the camera. The key contribution of this work is a computationally tractable approach for taking into account the rolling-shutter effect, incurring only minimal approximations. The experimental results from the application of the method show that it is able to track, in real time, the position of a mobile phone moving in an unknown environment with an error accumulation of approximately 0.8% of the distance travelled, over hundreds of meters.

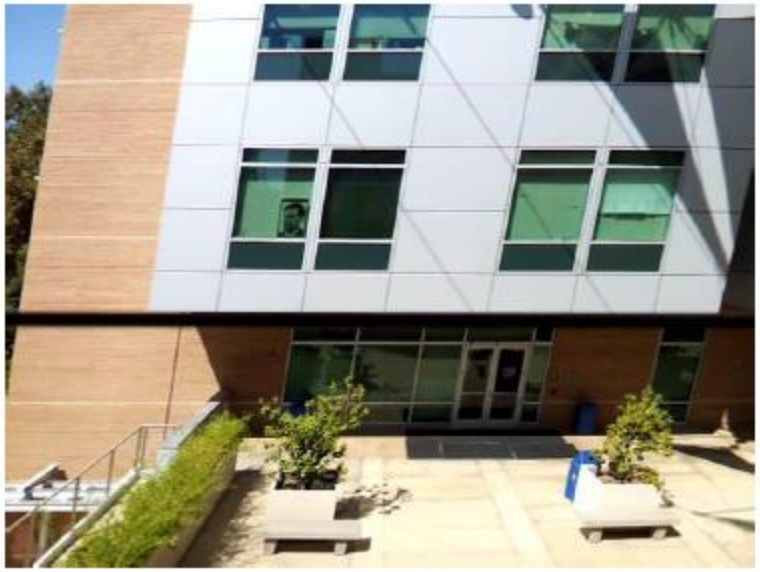


Fig. 1: An example image with rolling-shutter distortion.

Smartphone as a Sensor Platform

[Li et al., ICRA ‘13]

- Quadcore
- Multiple cameras
- Accelerometer
- Gyroscopes
- GPS
- Wireless
- Relatively cheap

Challenges
[Li et al., ICRA '13]

- Rolling shutter camera
- Poor synchronization between camera and IMU

Kalman Filter
[Li et al., ICRA '13]

- Kalman state contains
  - IMU pose
  - last m camera poses

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{I_k}^T & \boldsymbol{\pi}_{C_{k-m}}^T & \cdots & \boldsymbol{\pi}_{C_{k-1}}^T \end{bmatrix}^T$$

IMU state   camera poses of m last states

- We need to define
  - Motion model
  - Observation model

Kalman Filter: Motion Model
[Li et al., ICRA '13]

- Motion model (IMU)
  - Angular velocity

$$\underset{\substack{\uparrow\\ \text{measure-}\\ \text{ment}}}{\omega_m} = \underset{\substack{\uparrow\\ \text{true}\\ \text{velocity}}}{{}^{I}\omega} + \underset{\substack{\uparrow\\ \text{bias}}}{\mathbf{b_g}} + \underset{\substack{\uparrow\\ \text{noise}}}{\mathbf{n_r}}$$

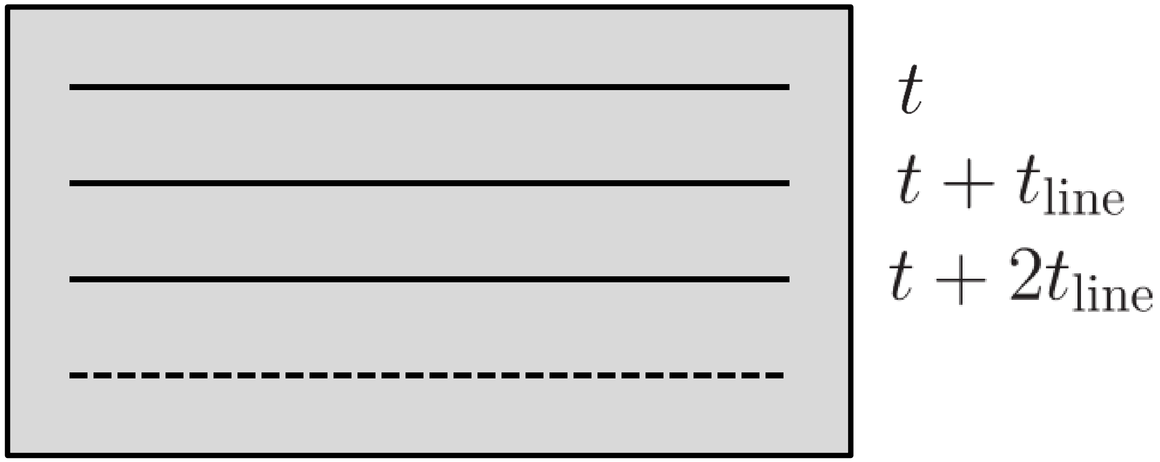
  - Linear acceleration

$$\underset{\substack{\uparrow\\ \text{measure-}\\ \text{ment}}}{\mathbf{a}_m} = \underset{\substack{\uparrow\\ \text{orient-}\\ \text{ation}}}{{}^{I}_{G}\mathbf{R}} \left(\underset{\substack{\uparrow\\ \text{true}\\ \text{acc.}}}{{}^{G}\mathbf{a}} - \underset{\substack{\uparrow\\ \text{gravity}}}{{}^{G}\mathbf{g}} \right) + \underset{\substack{\uparrow\\ \text{bias}}}{\mathbf{b_a}} + \underset{\substack{\uparrow\\ \text{noise}}}{\mathbf{n_a}}$$

Rolling Shutter

[Li et al., ICRA '13]

- Read-out time of the whole image $t_{\text{image}} = n t_{\text{line}}$
- Observation time of first row t
- Observation time of i-th row $t + i t_{\text{line}}$



Kalman Filter: Observation Model

[Li et al., ICRA ‘13]

- Observations from KLT tracker
  - 2D observations of 3D points (3D positions estimated by triangulation, covered next week)
- Observation function

$$\mathbf{z}_j = \mathbf{h}({}^{C_j}\mathbf{p}_f) + \mathbf{n}_j$$

- Observation function with rolling shutter comp.

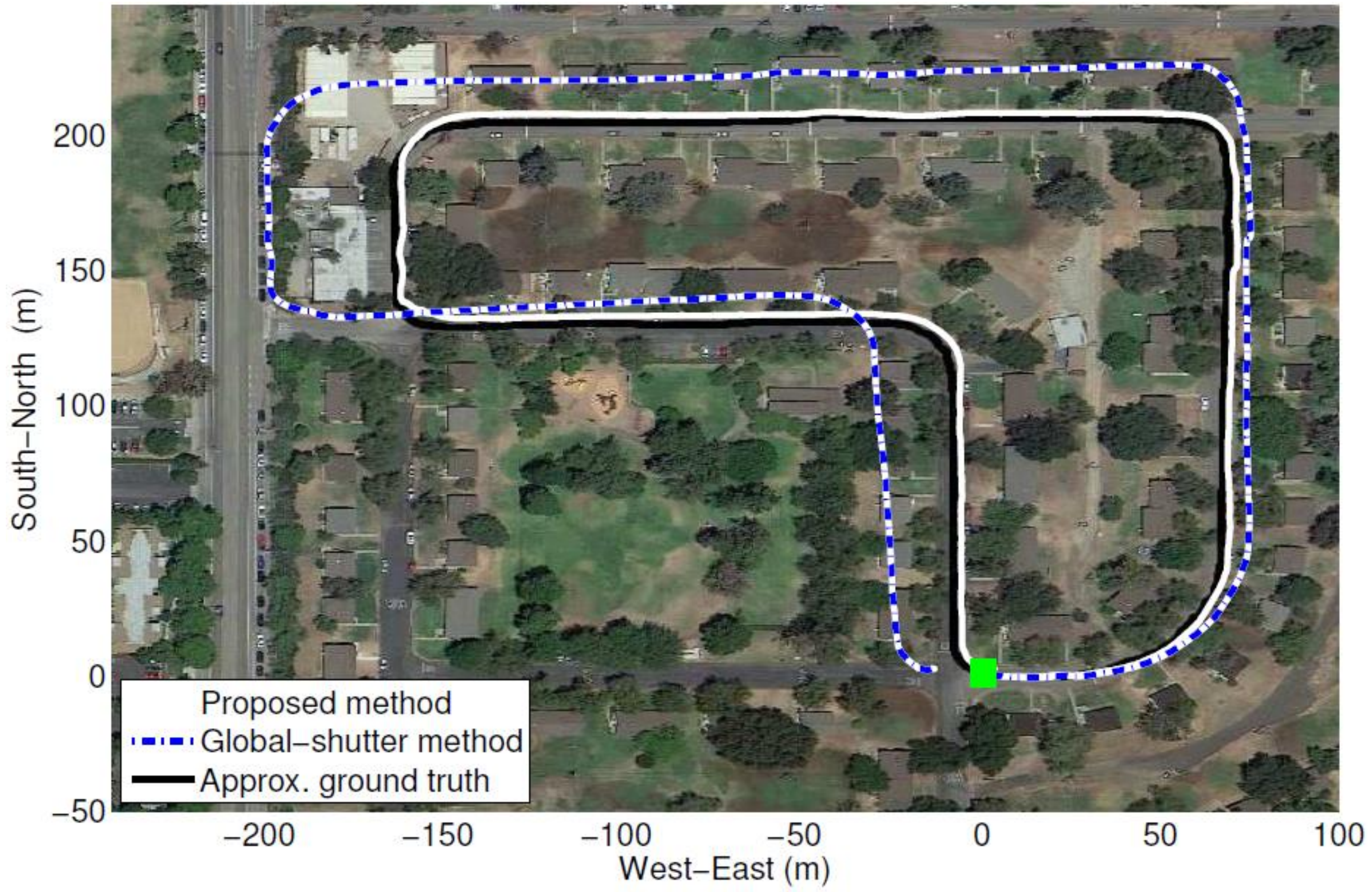
$$\mathbf{z}_j^{(n)} = \mathbf{h}({}^{C}\mathbf{p}_f(t_j + nt_d)) + \mathbf{n}_j^{(n)}$$

Results

[Li et al., ICRA '13]

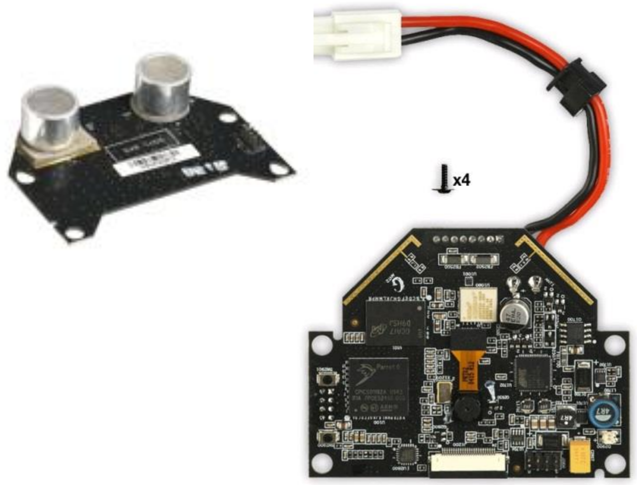
- Galaxy S2 (live demo at ICRA with S3)
- IMU updates at 90 Hz (downsampled)
  - Gyroscope at 106 Hz
  - Accelerometer at 93 Hz
- Images at 15 Hz
  - Read-out time 32msec
- 610m and 900m trajectory
- Drift less than 0.8%, absolute scale from IMU

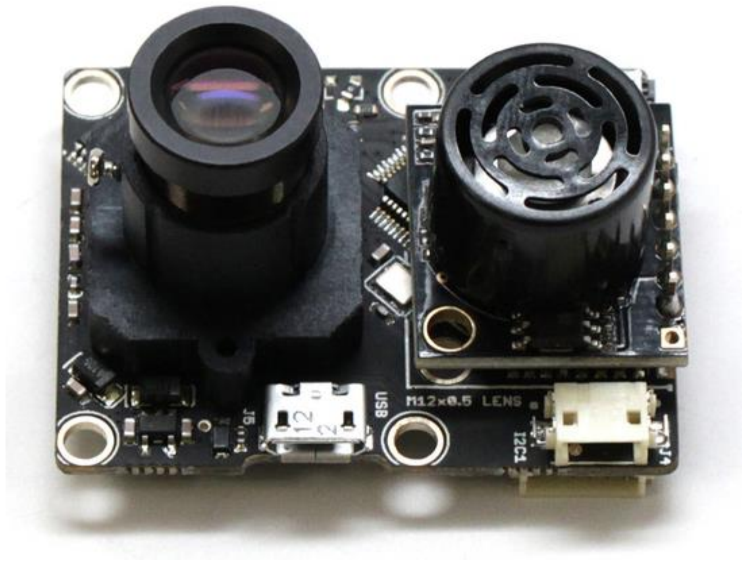
Results
[Li et al., ICRA '13]



Commercial Solutions

- Pix4flow from ETH/3D robotics
  1 camera, IMU, ultrasound, 150 EUR
- Parrot Mainboard + Navigation board
  1 camera, IMU, ultrasound, 210 USD

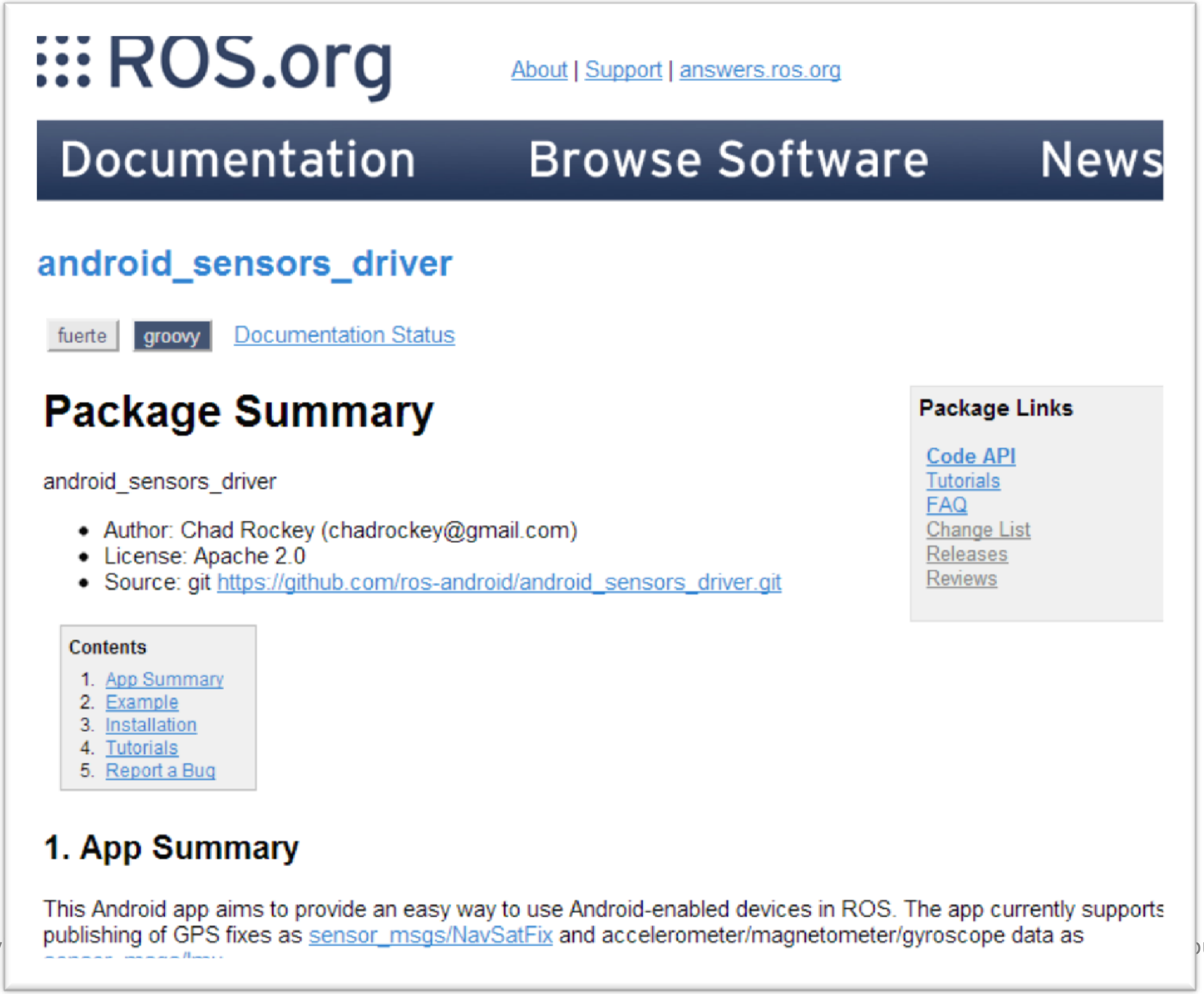




Talks at ROSCon 2013

- ROS Conference (in conjunction with ICRA)
- Talk by Chad Rockey (Willow Garage) on Android sensors driver
- Android app (Google Play store or github)
- Provides:
  - `/android/imu`
  - `/android/fix (GPS)`
  - `/camera/camera_info`
  - `/camera/image/compressed`

Android Sensors Driver



ROS.org

About | Support | answers.ros.org

Documentation Browse Software News

android_sensors_driver

fuerte groovy Documentation Status

Package Summary

android_sensors_driver

- Author: Chad Rockey (chadrockey@gmail.com)
- License: Apache 2.0
- Source: git https://github.com/ros-android/android_sensors_driver.git

Package Links

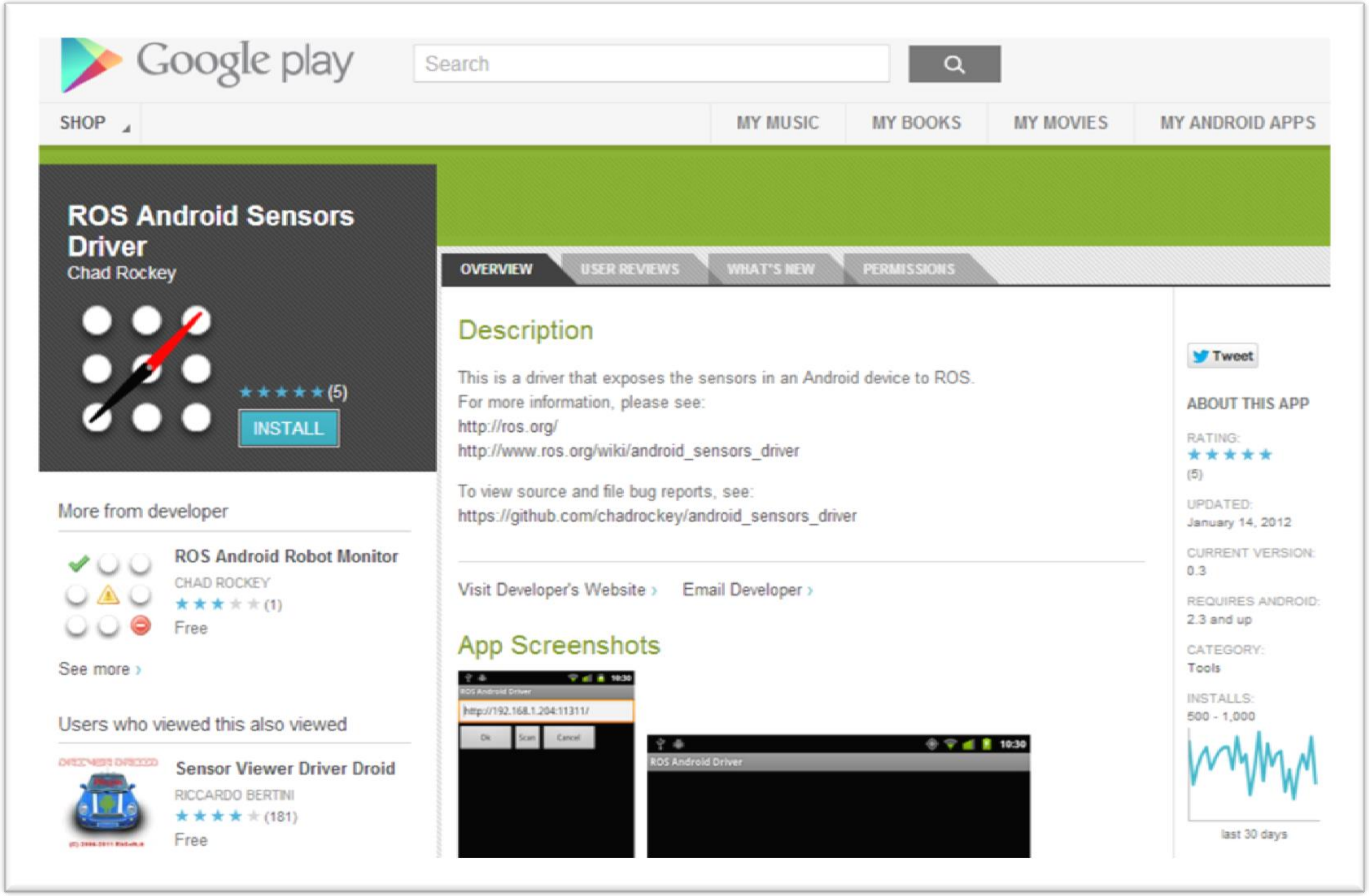
Code API
Tutorials
FAQ
Change List
Releases
Reviews

Contents

1. App Summary

This Android app aims to provide an easy way to use Android-enabled devices in ROS. The app currently supports publishing of GPS fixes as sensor_msgs/NavSatFix and accelerometer/magnetometer/gyroscope data as

Android Sensors Driver



ROS Android Sensors Driver
Chad Rockey
★★★★★ (5)

Description

This is a driver that exposes the sensors in an Android device to ROS.
For more information, please see:
http://ros.org/
http://www.ros.org/wiki/android_sensors_driver

To view source and file bug reports, see:
https://github.com/chadrockey/android_sensors_driver

Visit Developer's Website › Email Developer ›

App Screenshots

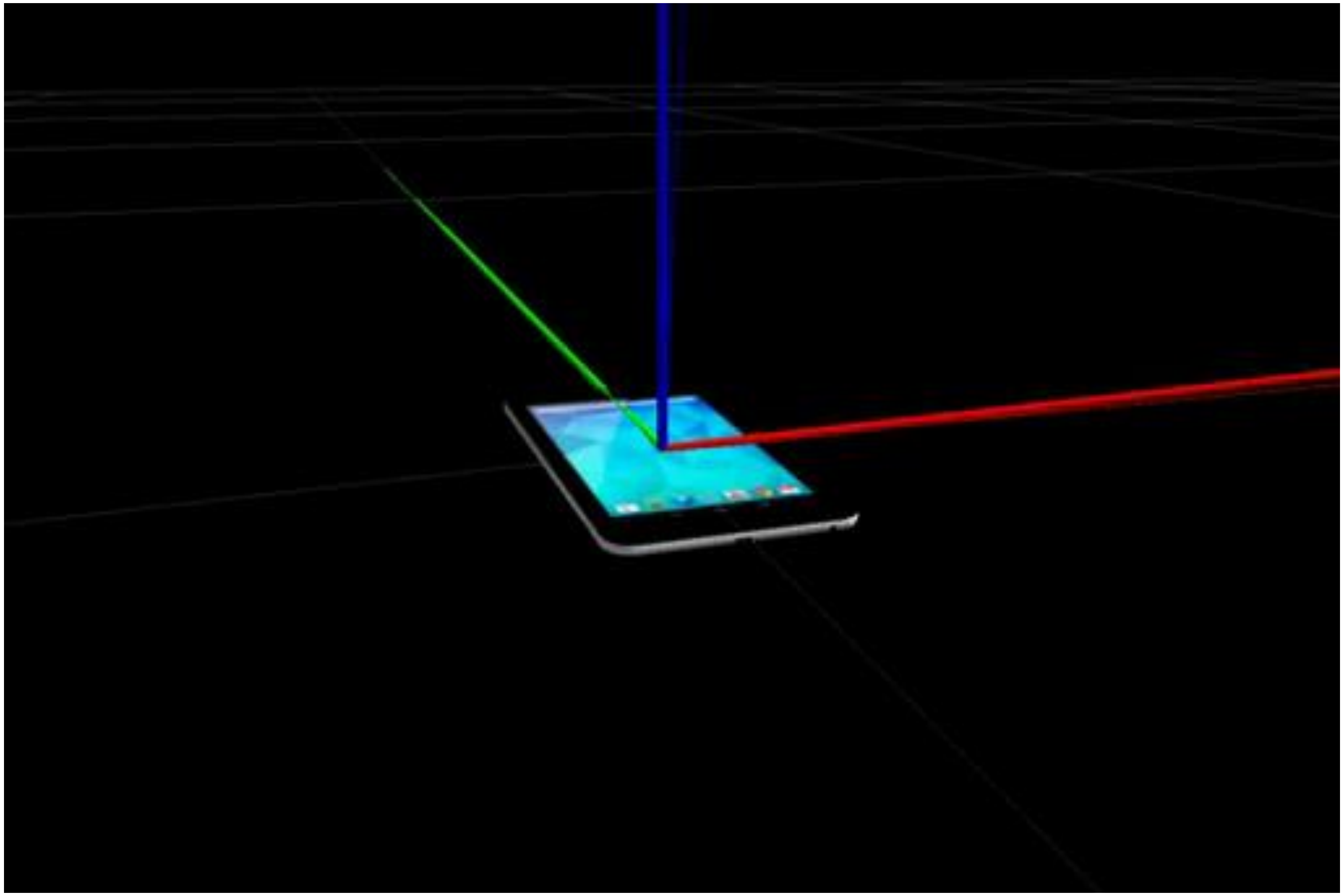
http://192.168.1.204:11311/

ABOUT THIS APP
RATING: ★★★★★ (5)
UPDATED: January 14, 2012
CURRENT VERSION: 0.3
REQUIRES ANDROID: 2.3 and up
CATEGORY: Tools
INSTALLS: 500 - 1,000
last 30 days

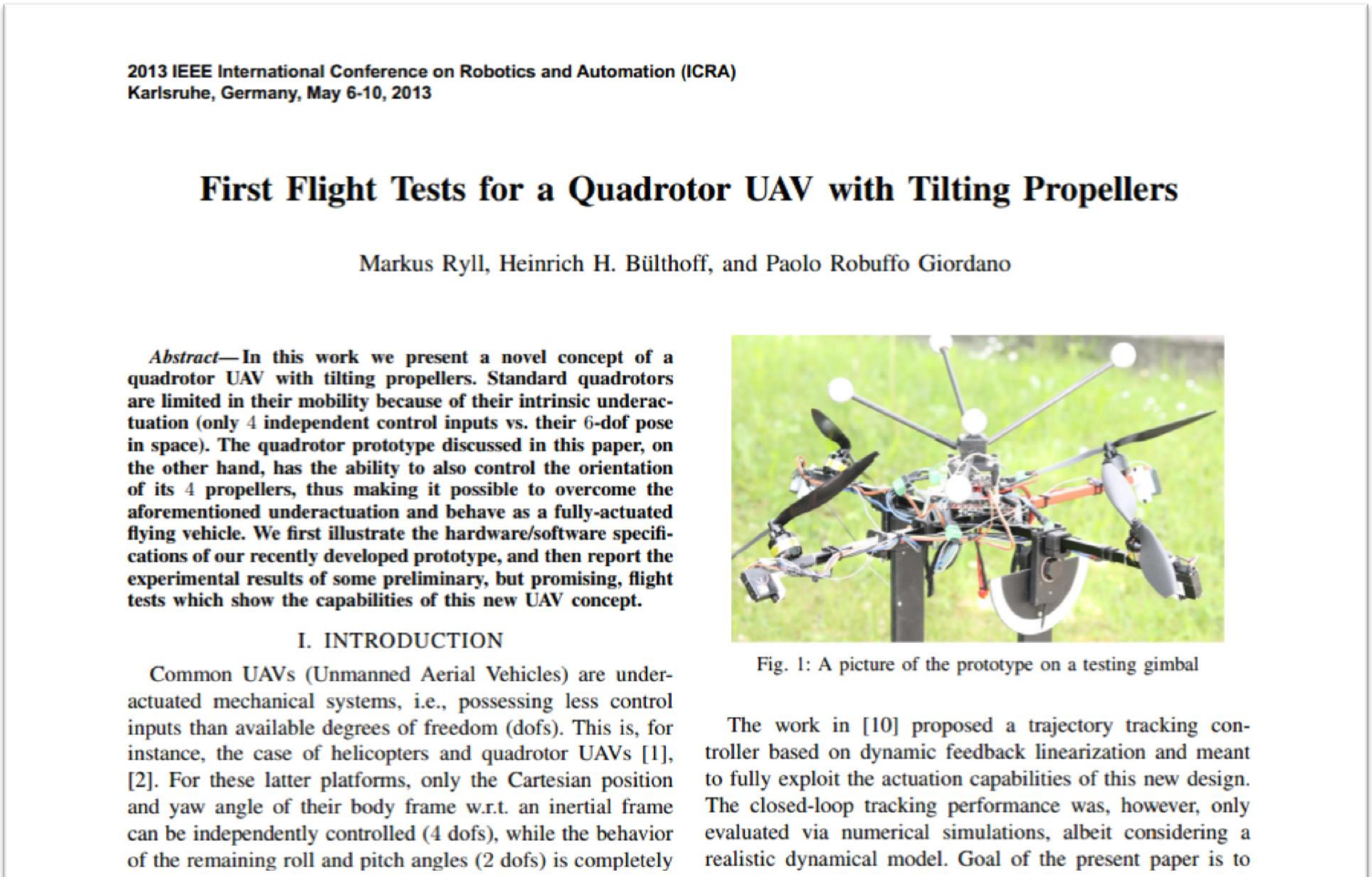
More from developer
ROS Android Robot Monitor
CHAD ROCKEY
★★★ (1)
Free
See more ›

Users who viewed this also viewed
Sensor Viewer Driver Droid
RICCARDO BERTINI
★★★★ (181)
Free

Android Sensors Driver



Cool ICRA Papers



2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

First Flight Tests for a Quadrotor UAV with Tilting Propellers

Markus Ryll, Heinrich H. Bülthoff, and Paolo Robuffo Giordano

Abstract—**In this work we present a novel concept of a quadrotor UAV with tilting propellers. Standard quadrotors are limited in their mobility because of their intrinsic underactuation (only 4 independent control inputs vs. their 6-dof pose in space). The quadrotor prototype discussed in this paper, on the other hand, has the ability to also control the orientation of its 4 propellers, thus making it possible to overcome the aforementioned underactuation and behave as a fully-actuated flying vehicle. We first illustrate the hardware/software specifications of our recently developed prototype, and then report the experimental results of some preliminary, but promising, flight tests which show the capabilities of this new UAV concept.**

I. INTRODUCTION

Common UAVs (Unmanned Aerial Vehicles) are underactuated mechanical systems, i.e., possessing less control inputs than available degrees of freedom (dofs). This is, for instance, the case of helicopters and quadrotor UAVs [1], [2]. For these latter platforms, only the Cartesian position and yaw angle of their body frame w.r.t. an inertial frame can be independently controlled (4 dofs), while the behavior of the remaining roll and pitch angles (2 dofs) is completely

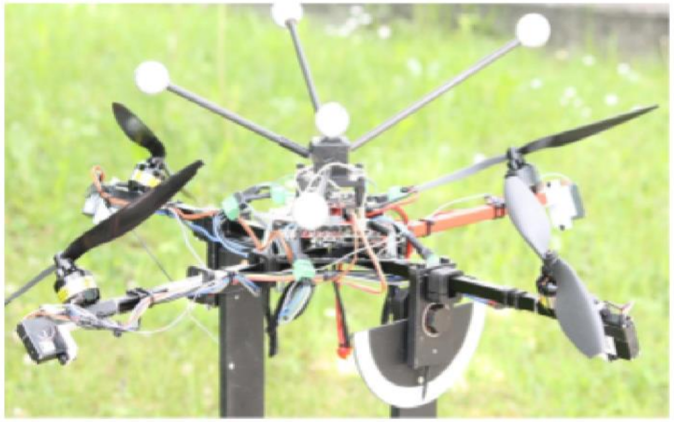
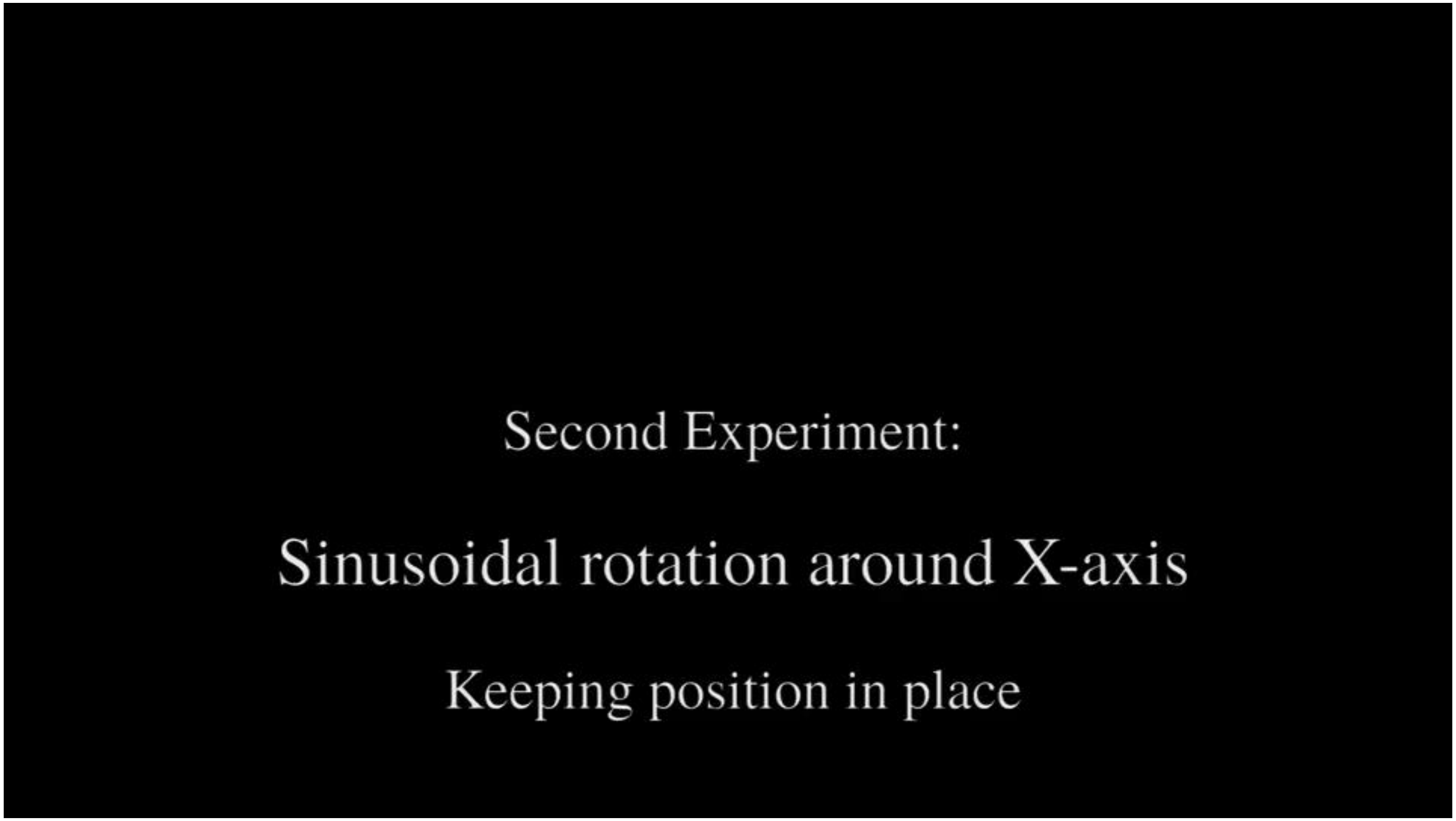


Fig. 1: A picture of the prototype on a testing gimbal

The work in [10] proposed a trajectory tracking controller based on dynamic feedback linearization and meant to fully exploit the actuation capabilities of this new design. The closed-loop tracking performance was, however, only evaluated via numerical simulations, albeit considering a realistic dynamical model. Goal of the present paper is to

Tilting Propellors
[Ryll et al., ICRA '13]

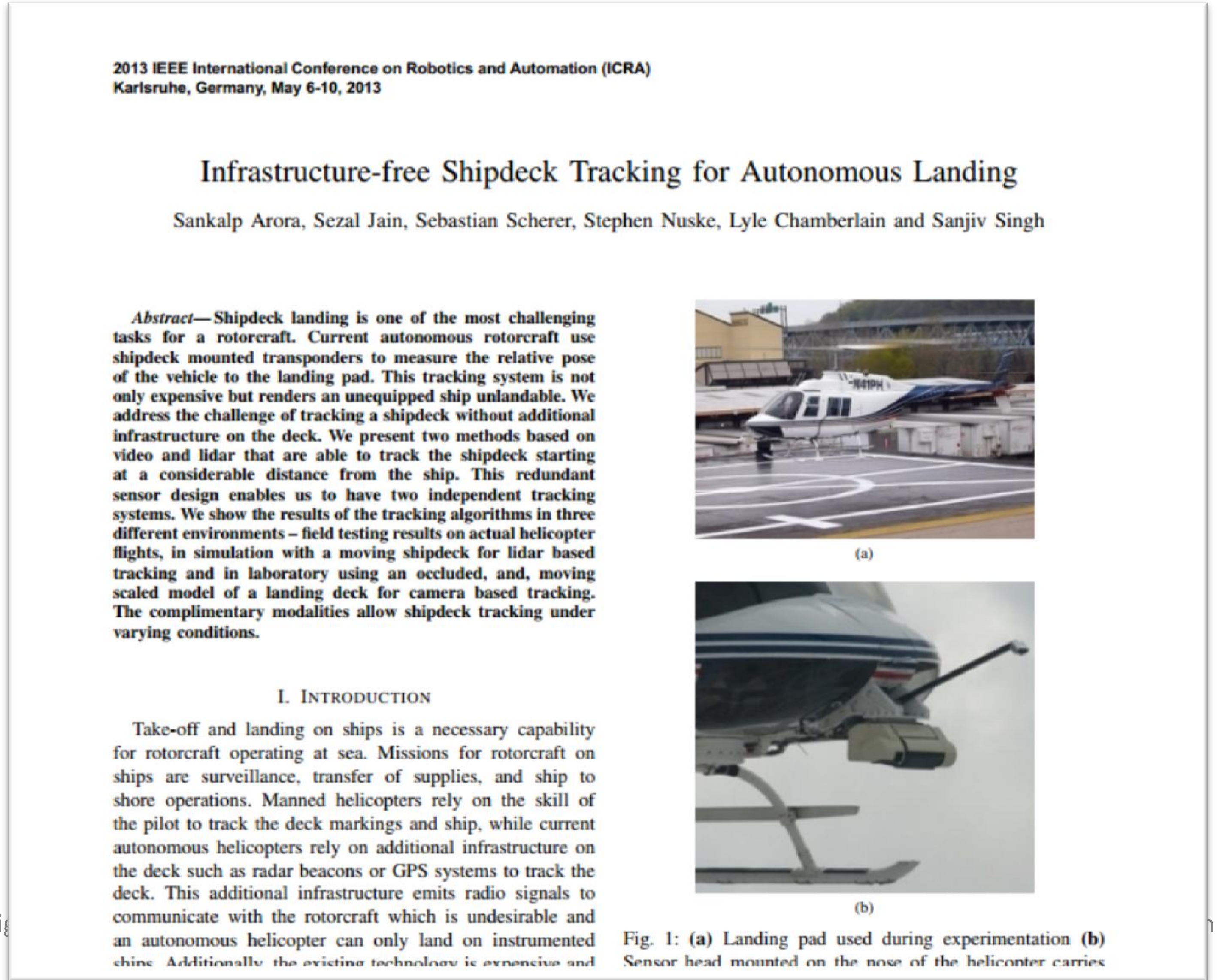

Second Experiment:

Sinusoidal rotation around X-axis

Keeping position in place

Shipdeck Tracking

[Arora et al., ICRA ‘13]



2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Infrastructure-free Shipdeck Tracking for Autonomous Landing

Sankalp Arora, Sezal Jain, Sebastian Scherer, Stephen Nuske, Lyle Chamberlain and Sanjiv Singh

Abstract—**Shipdeck landing is one of the most challenging tasks for a rotorcraft. Current autonomous rotorcraft use shipdeck mounted transponders to measure the relative pose of the vehicle to the landing pad. This tracking system is not only expensive but renders an unequipped ship unlandable. We address the challenge of tracking a shipdeck without additional infrastructure on the deck. We present two methods based on video and lidar that are able to track the shipdeck starting at a considerable distance from the ship. This redundant sensor design enables us to have two independent tracking systems. We show the results of the tracking algorithms in three different environments – field testing results on actual helicopter flights, in simulation with a moving shipdeck for lidar based tracking and in laboratory using an occluded, and, moving scaled model of a landing deck for camera based tracking. The complimentary modalities allow shipdeck tracking under varying conditions.**

I. Introduction

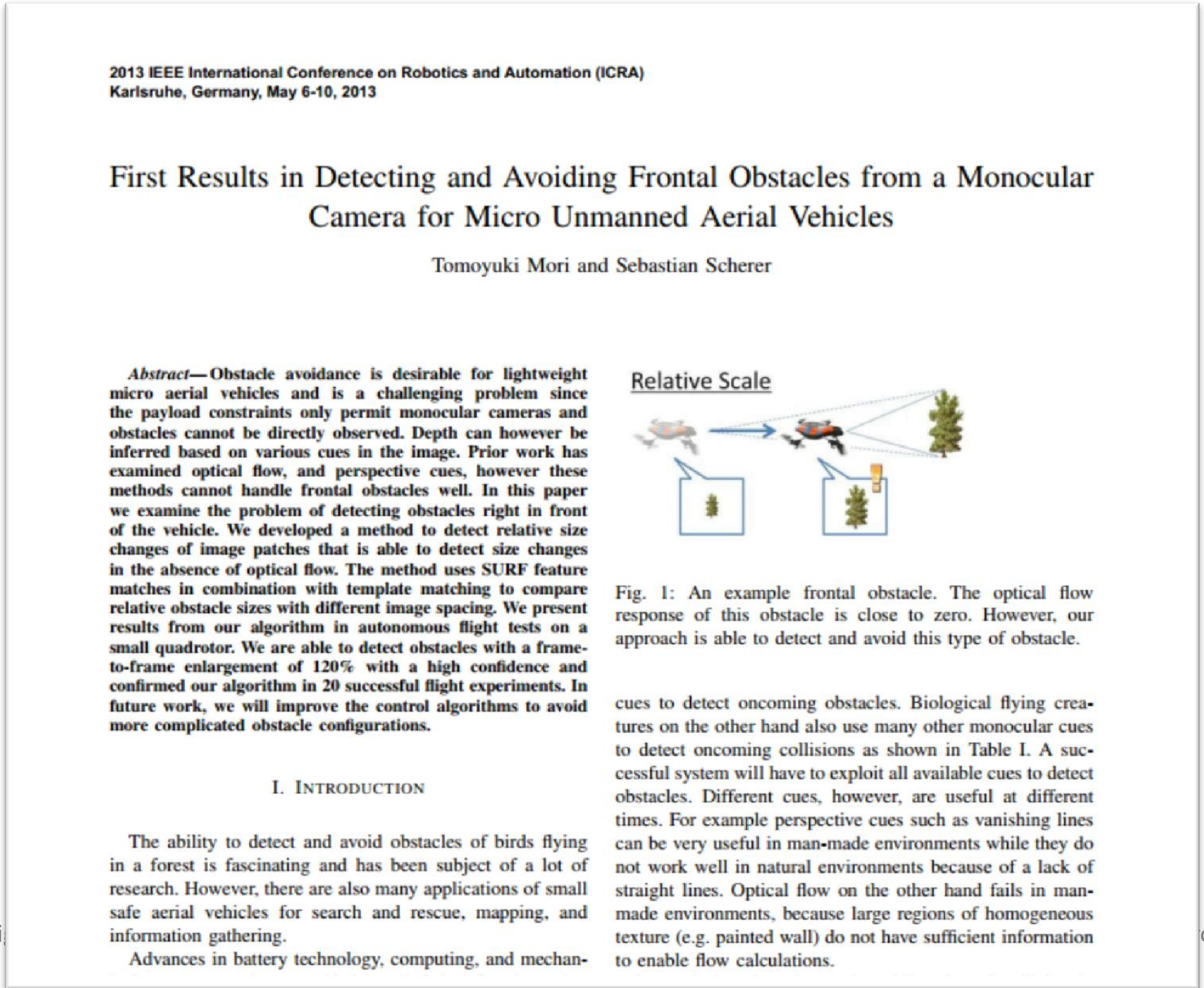
Take-off and landing on ships is a necessary capability for rotorcraft operating at sea. Missions for rotorcraft on ships are surveillance, transfer of supplies, and ship to shore operations. Manned helicopters rely on the skill of the pilot to track the deck markings and ship, while current autonomous helicopters rely on additional infrastructure on the deck such as radar beacons or GPS systems to track the deck. This additional infrastructure emits radio signals to communicate with the rotorcraft which is undesirable and an autonomous helicopter can only land on instrumented ships. Additionally, the existing technology is expensive and

Fig. 1: **(a)** Landing pad used during experimentation **(b)** Sensor head mounted on the nose of the helicopter carries

Shipdeck Tracking

[Arora et al., ICRA '13]

Cool ICRA Papers



2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

First Results in Detecting and Avoiding Frontal Obstacles from a Monocular Camera for Micro Unmanned Aerial Vehicles

Tomoyuki Mori and Sebastian Scherer

Abstract—**Obstacle avoidance is desirable for lightweight micro aerial vehicles and is a challenging problem since the payload constraints only permit monocular cameras and obstacles cannot be directly observed. Depth can however be inferred based on various cues in the image. Prior work has examined optical flow, and perspective cues, however these methods cannot handle frontal obstacles well. In this paper we examine the problem of detecting obstacles right in front of the vehicle. We developed a method to detect relative size changes of image patches that is able to detect size changes in the absence of optical flow. The method uses SURF feature matches in combination with template matching to compare relative obstacle sizes with different image spacing. We present results from our algorithm in autonomous flight tests on a small quadrotor. We are able to detect obstacles with a frame-to-frame enlargement of 120% with a high confidence and confirmed our algorithm in 20 successful flight experiments. In future work, we will improve the control algorithms to avoid more complicated obstacle configurations.**

I. Introduction

The ability to detect and avoid obstacles of birds flying in a forest is fascinating and has been subject of a lot of research. However, there are also many applications of small safe aerial vehicles for search and rescue, mapping, and information gathering.

Advances in battery technology, computing, and mechan-

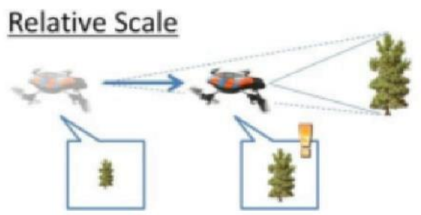
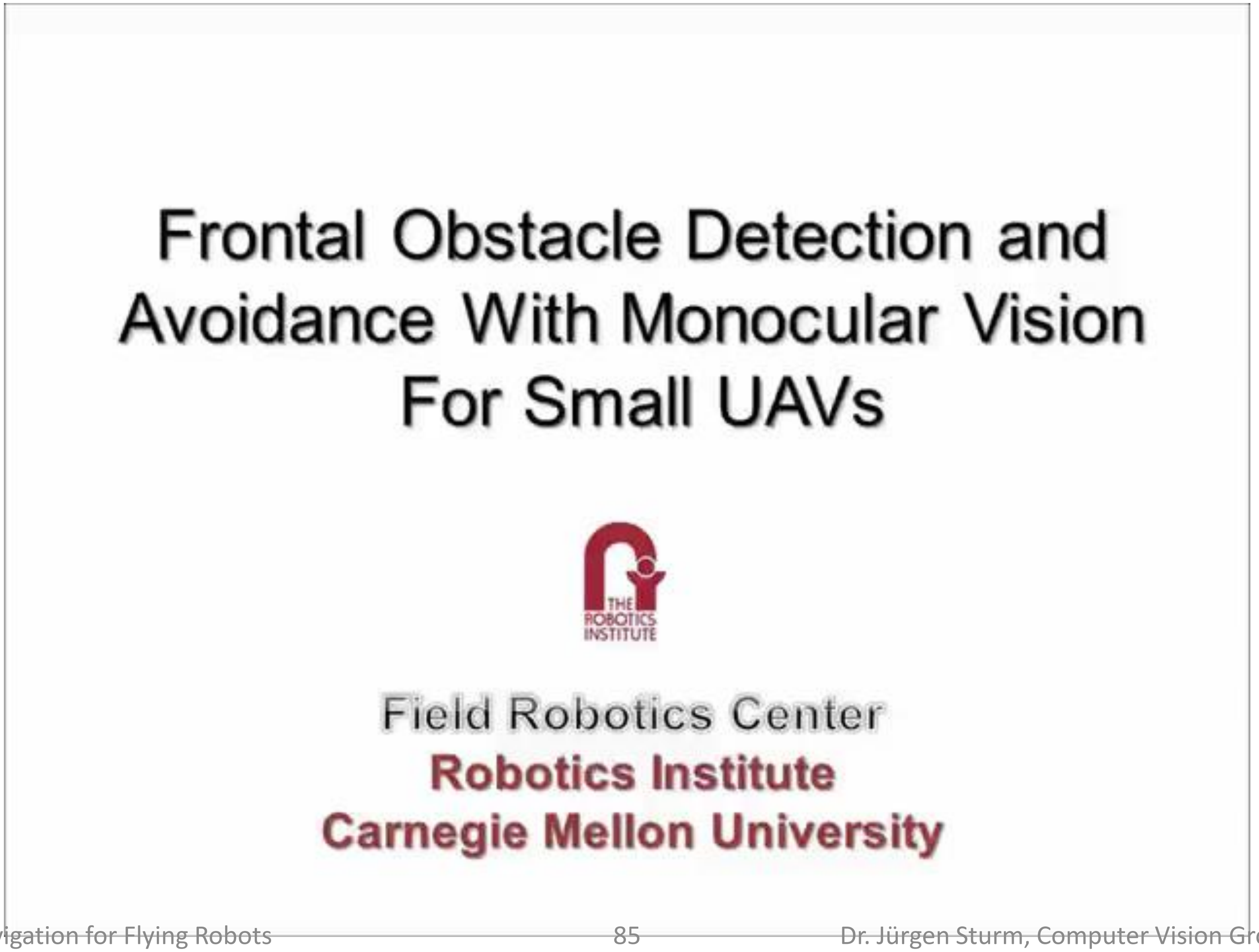


Fig. 1: An example frontal obstacle. The optical flow response of this obstacle is close to zero. However, our approach is able to detect and avoid this type of obstacle.

cues to detect oncoming obstacles. Biological flying creatures on the other hand also use many other monocular cues to detect oncoming collisions as shown in Table I. A successful system will have to exploit all available cues to detect obstacles. Different cues, however, are useful at different times. For example perspective cues such as vanishing lines can be very useful in man-made environments while they do not work well in natural environments because of a lack of straight lines. Optical flow on the other hand fails in man-made environments, because large regions of homogeneous texture (e.g. painted wall) do not have sufficient information to enable flow calculations.

Frontal Obstacle Avoidance

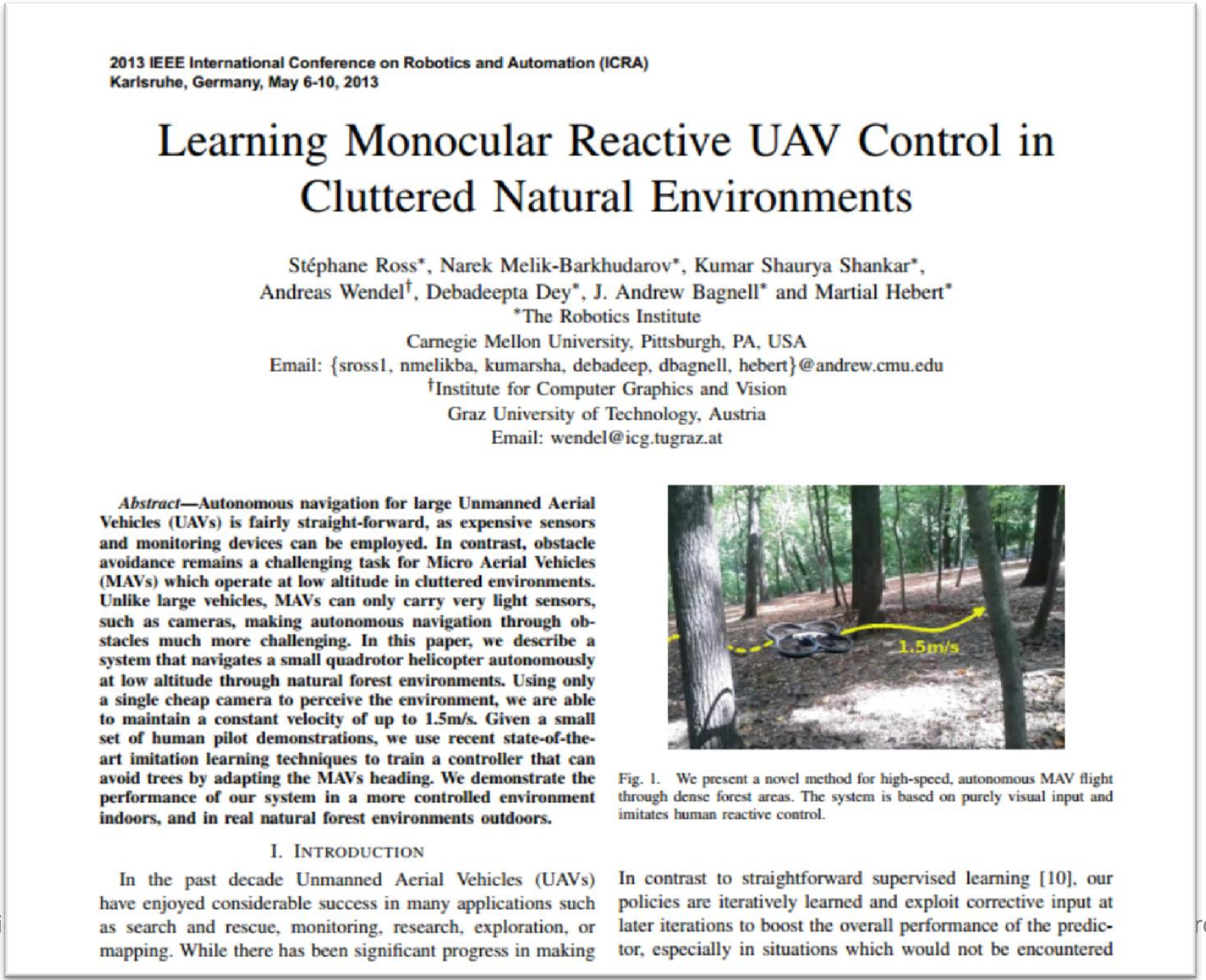
[Mori and Scherer, ICRA '13]



Frontal Obstacle Detection and Avoidance With Monocular Vision For Small UAVs

Field Robotics Center
Robotics Institute
Carnegie Mellon University

Cool ICRA Papers



2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Learning Monocular Reactive UAV Control in Cluttered Natural Environments

Stéphane Ross*, Narek Melik-Barkhudarov*, Kumar Shaurya Shankar*,
Andreas Wendel†, Debadeepta Dey*, J. Andrew Bagnell* and Martial Hebert*
*The Robotics Institute
Carnegie Mellon University, Pittsburgh, PA, USA
Email: {sross1, nmelikba, kumarsha, debadeep, dbagnell, hebert}@andrew.cmu.edu
†Institute for Computer Graphics and Vision
Graz University of Technology, Austria
Email: wendel@icg.tugraz.at

***Abstract*—Autonomous navigation for large Unmanned Aerial Vehicles (UAVs) is fairly straight-forward, as expensive sensors and monitoring devices can be employed. In contrast, obstacle avoidance remains a challenging task for Micro Aerial Vehicles (MAVs) which operate at low altitude in cluttered environments. Unlike large vehicles, MAVs can only carry very light sensors, such as cameras, making autonomous navigation through obstacles much more challenging. In this paper, we describe a system that navigates a small quadrotor helicopter autonomously at low altitude through natural forest environments. Using only a single cheap camera to perceive the environment, we are able to maintain a constant velocity of up to 1.5m/s. Given a small set of human pilot demonstrations, we use recent state-of-the-art imitation learning techniques to train a controller that can avoid trees by adapting the MAVs heading. We demonstrate the performance of our system in a more controlled environment indoors, and in real natural forest environments outdoors.**

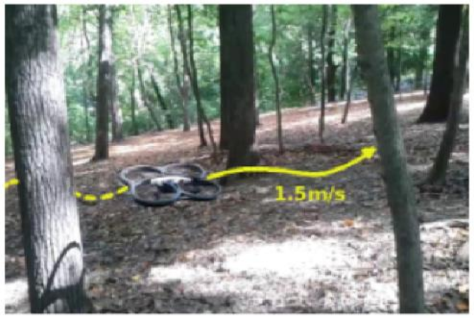


Fig. 1. We present a novel method for high-speed, autonomous MAV flight through dense forest areas. The system is based on purely visual input and imitates human reactive control.

I. Introduction

In the past decade Unmanned Aerial Vehicles (UAVs) have enjoyed considerable success in many applications such as search and rescue, monitoring, research, exploration, or mapping. While there has been significant progress in making

In contrast to straightforward supervised learning [10], our policies are iteratively learned and exploit corrective input at later iterations to boost the overall performance of the predictor, especially in situations which would not be encountered

Learning to Avoid Obstacles

[Ross et al., ICRA ‘13]

ICRA Proceedings

- All papers of a conference are (usually) collected into so-called proceedings
- Previously: One or more books
- Today: USB sticks or online

- Will put ICRA proceedings online (see website)
- Remember password (or ask by email)

Ideas for Your Mini-Project

- Person following (colored shirt or wearing a marker)
- Flying camera for taking group pictures (possibly using the OpenCV face detector)
- Fly through a hula hoop (brightly colored, white background)
- Navigate through a door (brightly colored)
- Navigate from one room to another (using ground markers)
- Avoid obstacles using optical flow
- Landing on a marked spot/moving target
- **Your own idea here – be creative!**
- ...

Lessons Learned Today

- How to estimate the translational motion from camera images
- Which image patches are easier to track than others
- How to estimate 2D motion from camera images
- Summary of ICRA and ROSCon papers/talks