Tracking surfaces / objects across video frames

Readings:

Paper, First dozen pages:
Baker&Matthews, IJCV

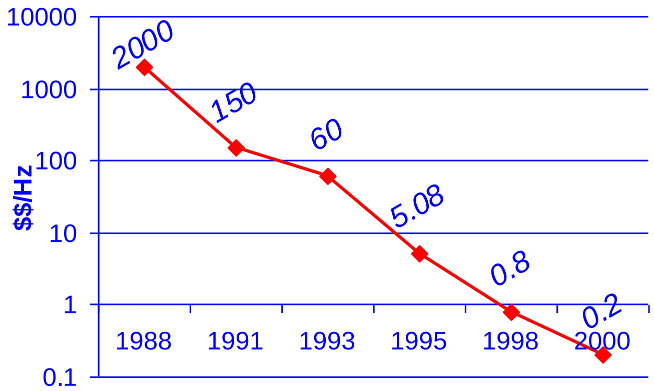
“Lukas-Kanade 20 years on”

Szeliski 8.1.3, 4.1

Ma et al Ch 4.

Forsythe and Ponce Ch 17.

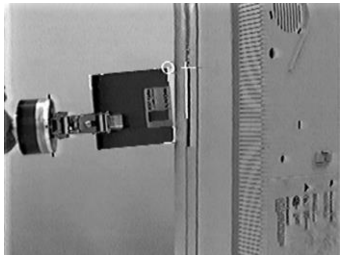
Why Visual Tracking?



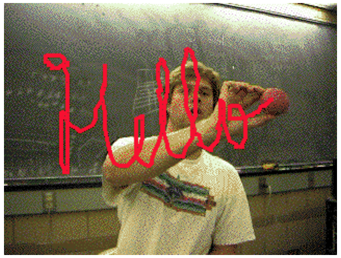
Computers are fast enough!

Related technology is cheap!

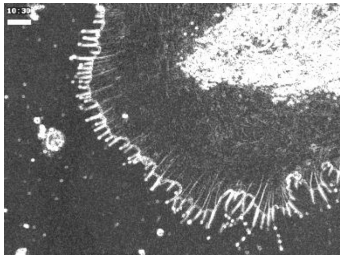
Applications abound



Motion Control



HCI



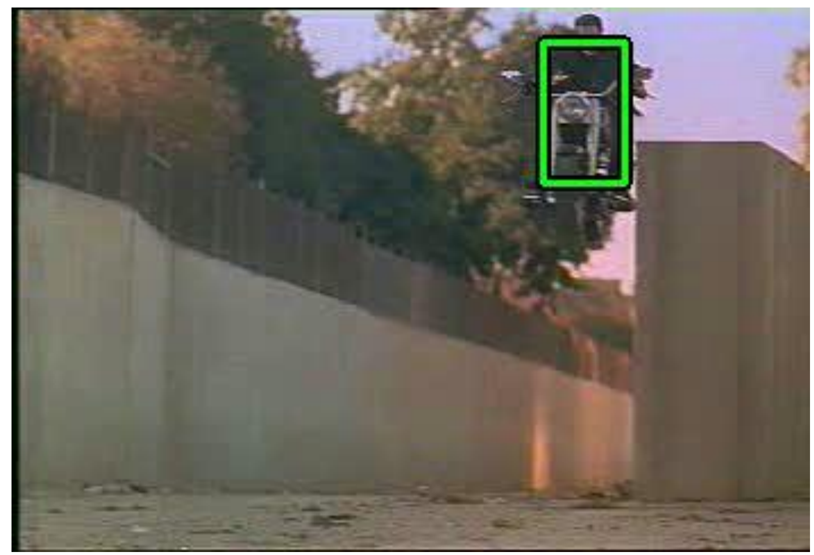
Measurement

Applications: Watching a moving target

- Camera + computer can determine how things move in an scene over time.

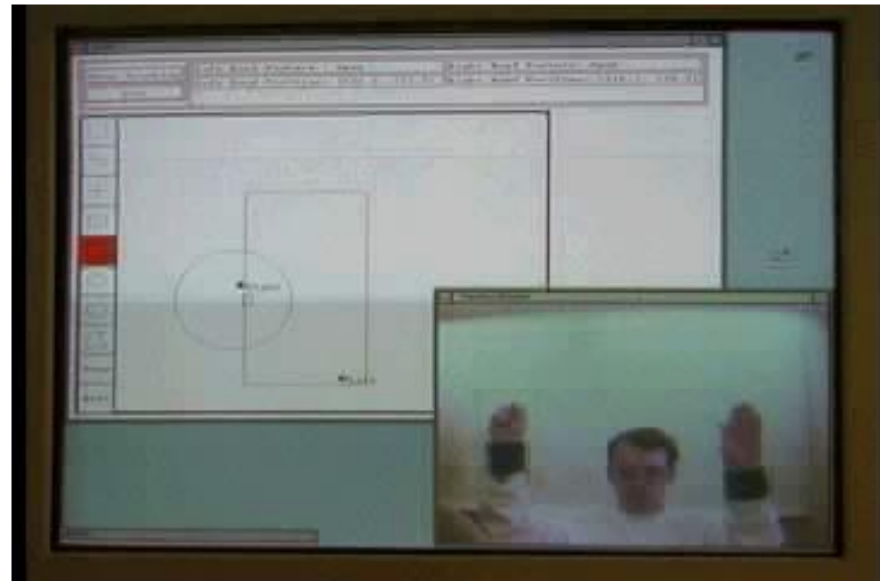
Uses:

- Security: e.g. monitoring people moving in a subway station or store
- Measurement: Speed, alert on colliding trajectories etc.



Applications
Human-Computer Interfaces

- Camera + computer tracks motions of human user and interprets this in an on-line interaction.
- Can interact with menus, buttons and e.g. drawing programs using hand movements as mouse movements, and gestures as clicking
- Furthermore, can interpret physical interactions



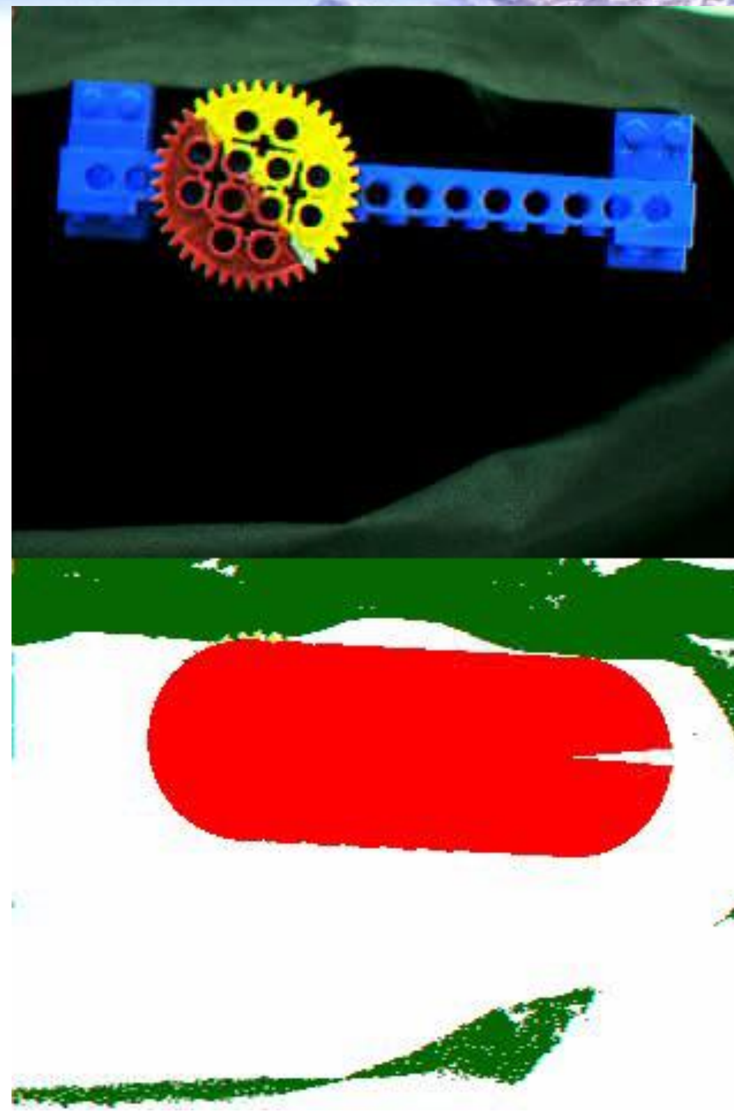
Applications
Interpreting tasks and functinality

Use tracking to interpret:

1.Physics of a mechanism

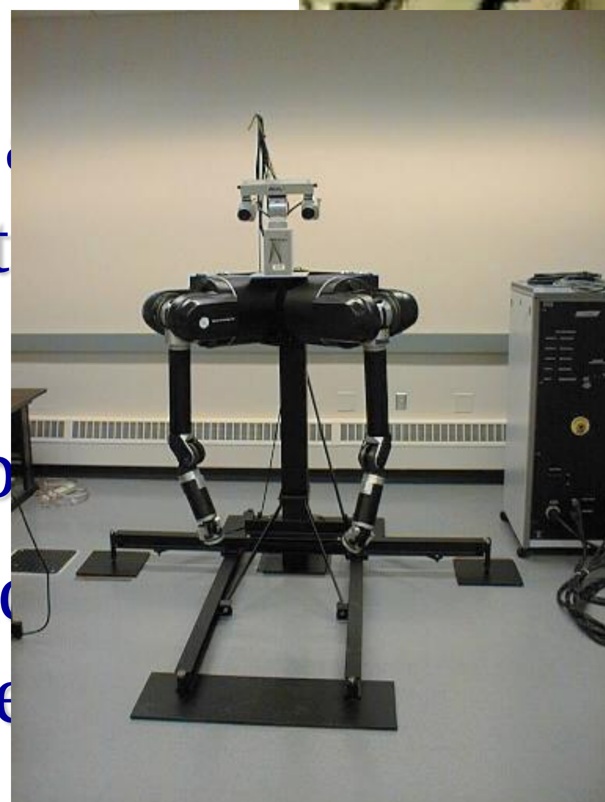
2.How a car is repaired.

3. How a human cooks in a kitchen



Applications
Human-Machine Interfaces

- Camera + computer tracks motions of human user, interprets this
machine/robot
out task.
- Remote manip
- Service roboti
the handicappe
elderly



Applications
Human-Human Interfaces

- Camera + computer tracks motions and expressions of human user, interprets, codes and transmits to render at remote location
- Ultra low bandwidth video communication
- Handheld video cell phone

BANDWIDTH and PROCESSIN REQUIREMENTS: TV camera

<u>Binary</u>
1 bit * 640x480 * 30 = 9.2 Mbits/second

<u>Grey</u>
1 byte * 640x480 * 30 = 9.2 Mbytes/second

<u>Color</u>
3 bytes * 640x480 * 30 = 27.6 Mbytes/second (actually about 37 mbytes/sec)

Typical operation: 8x8 convolution on grey image
64 multiplies + adds ➔ 600 Mflops

Consider 800x600, 1200x1600 (2MP)…

Today’s PC’s are getting to the point they
can process images at frame rate
For real time: process small window in image

Characteristics of Video Processing

abs(Image 1 - Image 2) = ?

Note: Almost all pixels change!

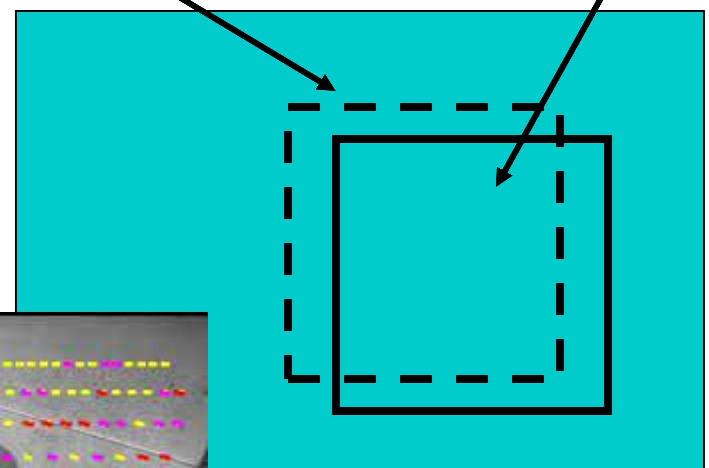
Constancy: The physical scene is the same

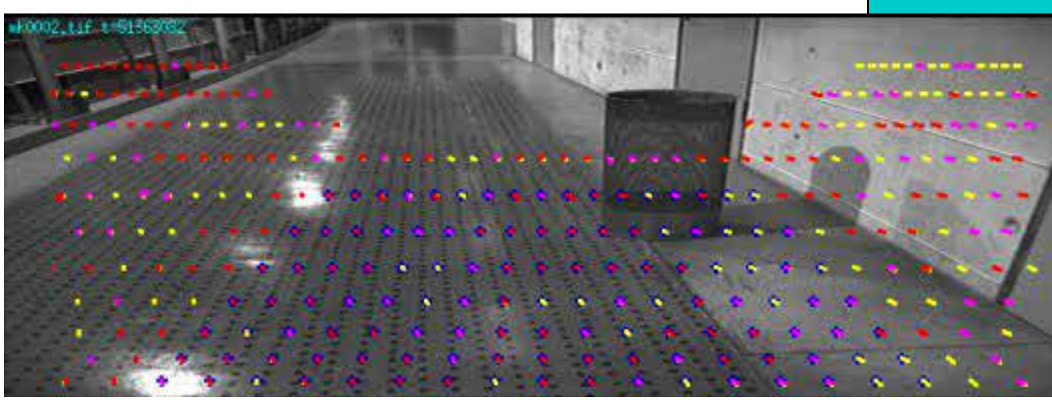
How do we make use of this?

Fundamental types of video processing "Visual motion detecton"

- Relating two adjacent frames: (small differences):

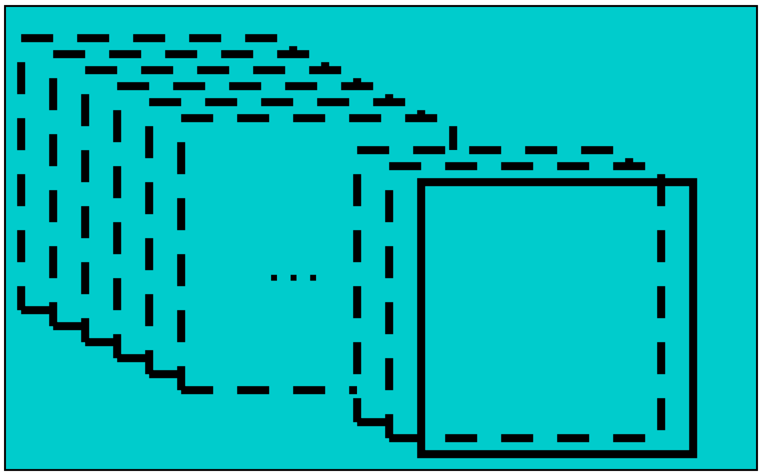
$$\mathrm{Im}(x+\delta x,\ \ y+\delta y,\ \ t+\delta t)\ \ =\ \ \mathrm{Im}(x,\ \ y,\ \ t)$$

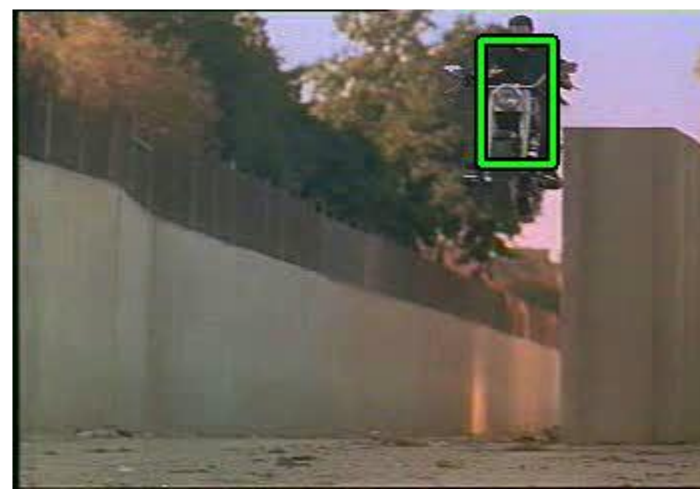




Fundamental types of video processing “Visual Tracking” / Stabilization

- Globally relating all frames: (large differences):





Types of tracking

- **Point tracking**

  Extract the point (pixel location) of a particular image feature in each image.

- **Segmentation based tracking**

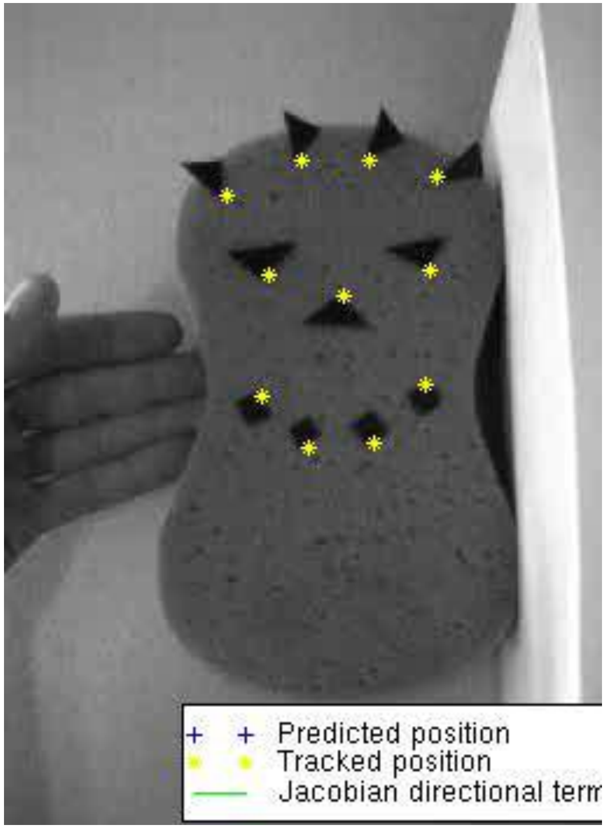
  Define an identifying region property (e.g. color, texture statistics). Repeatedly extract this region in each image.

- **Stabilization based tracking**

  Formulate image geometry and appearance equations and use these acquire image transform parameters for each image.

Point tracking

- Simplest technique. Already commercialized in motion capture systems.
- Features commonly LED’s (visible or IR), special markers (reflective, patterns)
- Detection: e.g. pick brightest pixel(s), cross correlate image to find best match for known pattern.



Commercial point trackers: "Motion capture"

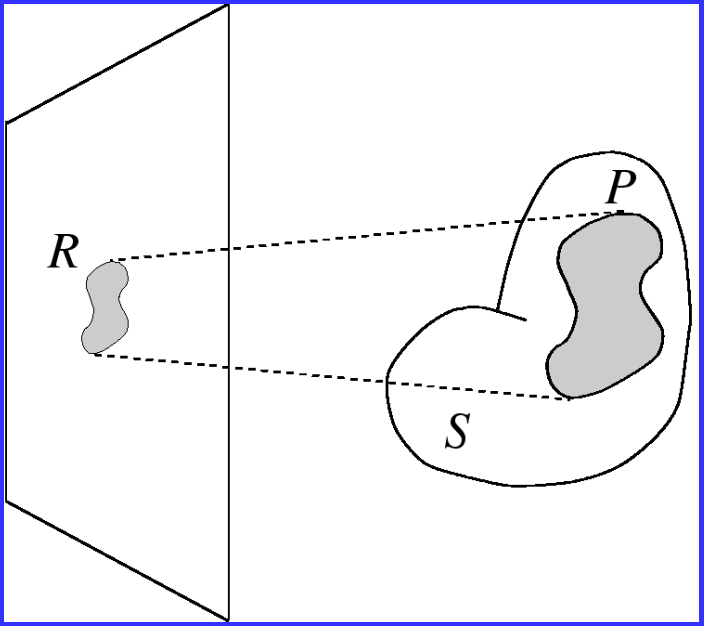
Region Tracking (Segmentation-Based)

- Select a "cue:" $\gamma(\mathbf{I}(x, y))$
  - foreground enhancement
  - background subtraction
- Segment
  - threshold
  - "clean up"
- Compute region geometry
  - centroid (first moment)
  - orientation (second moment)
  - scale (second moment)

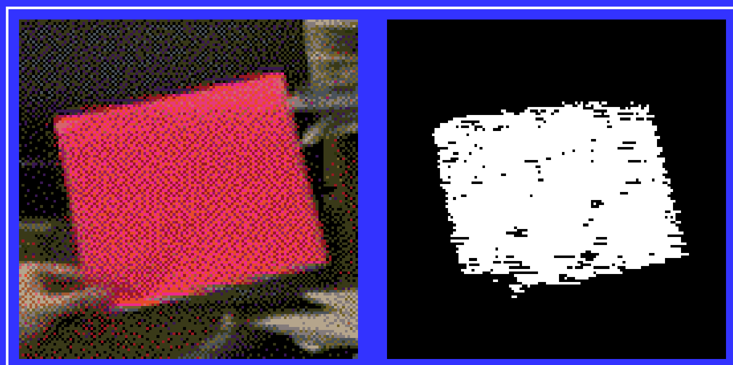
$$u = (x, y)'$$
$$m_i = \sum_i S(u) u^i$$
$$c = m_1 / m_0$$
$$\Lambda = m_2 / m_0 - m_1^2$$

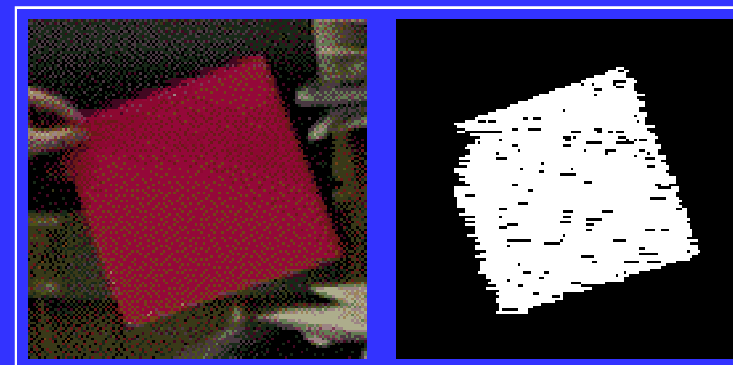
Regions

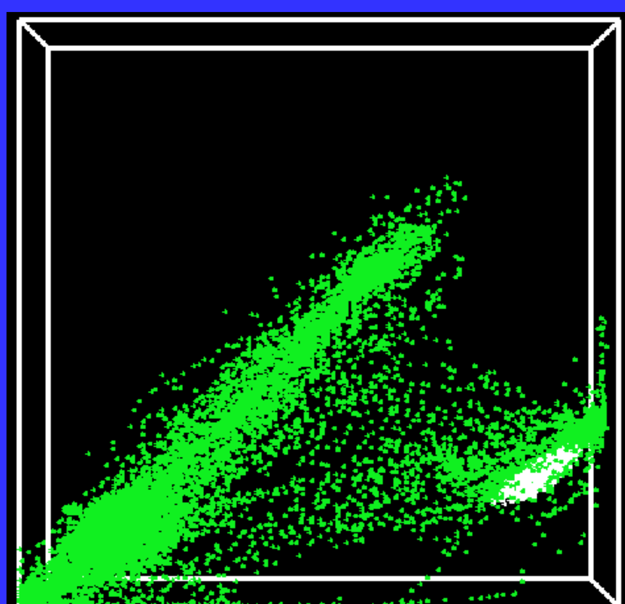
- $c_P = P \to \Re^3$ (color space): intrinsic coloration
  - Homogeneous region: $c_P(P)$ is roughly constant
  - Textured region: $c_P(P)$ has significant intensity gradients horizontally and vertically
  - Contour: (local) rapid change in contrast

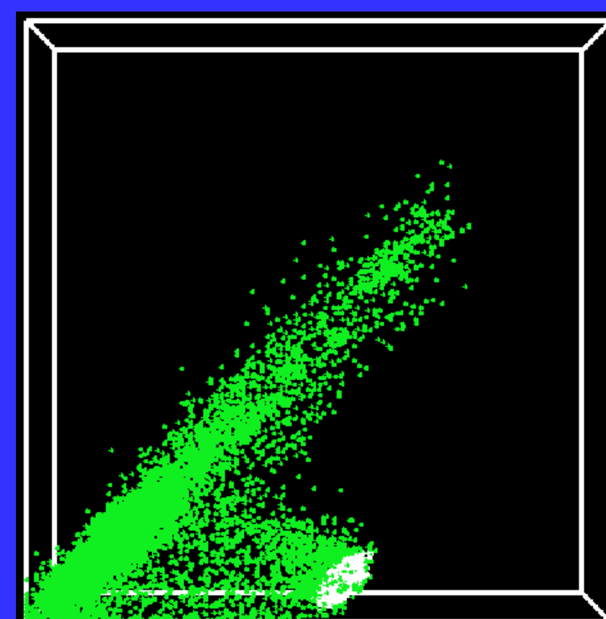


Homogeneous Color Region: Photometry







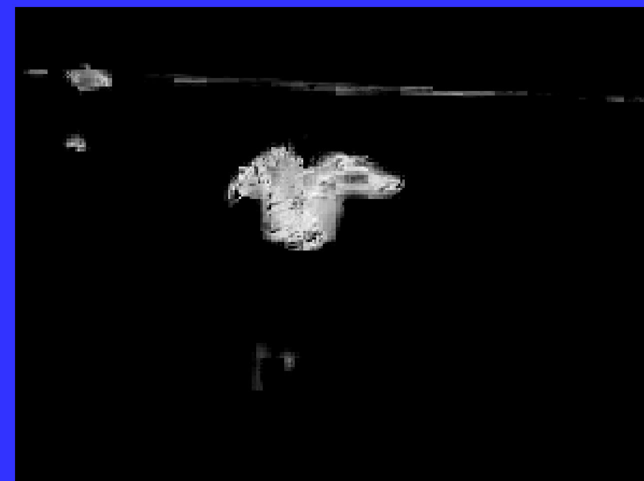


Color Representation

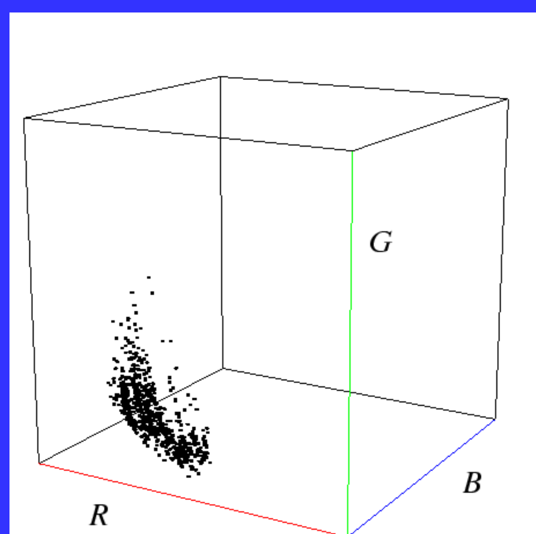
- $c_R = R \rightarrow \Re^3$ is irradiance of region *R*
- Color representation
  - DRM [Klinker et al., 1990]: if P is Lambertian, has *matte* line and *highlight* line
  - User selects *matte* pixels in *R*
  - PCA fits ellipsoid $(\mathbf{S}, \mathbf{R}^T, \mathbf{T})$ to matte cluster
  - Color similarity $\gamma(\mathbf{I}(x,y))$ is defined by Mahalanobis distance

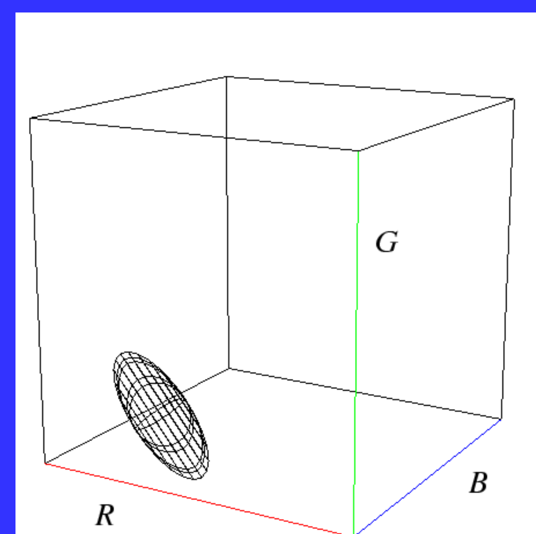
$$|\mathbf{S}^{-1}\mathbf{R}^T(\mathbf{I}(x,y)-\mathbf{T})| < 1 inside$$

Homogeneous Region: Photometry



Sample





PCA-fitted ellipsoid

Color Extension (Contd.)

Color Histograms (Swain et al.)

- H_i = number of pixels in class i
- Histograms are vectors of bin counts
- Given histograms H and G, compare by

$$H \cdot G = \frac{\sum H_i G_i}{\sqrt{\sum H_i^2}\sqrt{\sum G_i^2}}$$

  - dense, stable signature
  - relies on segmentation
  - relative color and feature locations lost
  - affine transformations preserve area ratios of planar objects

Intro to Registration Tracking

p_t

I_0 I_t

Variability model: $I_t = g(I_0, p_t)$

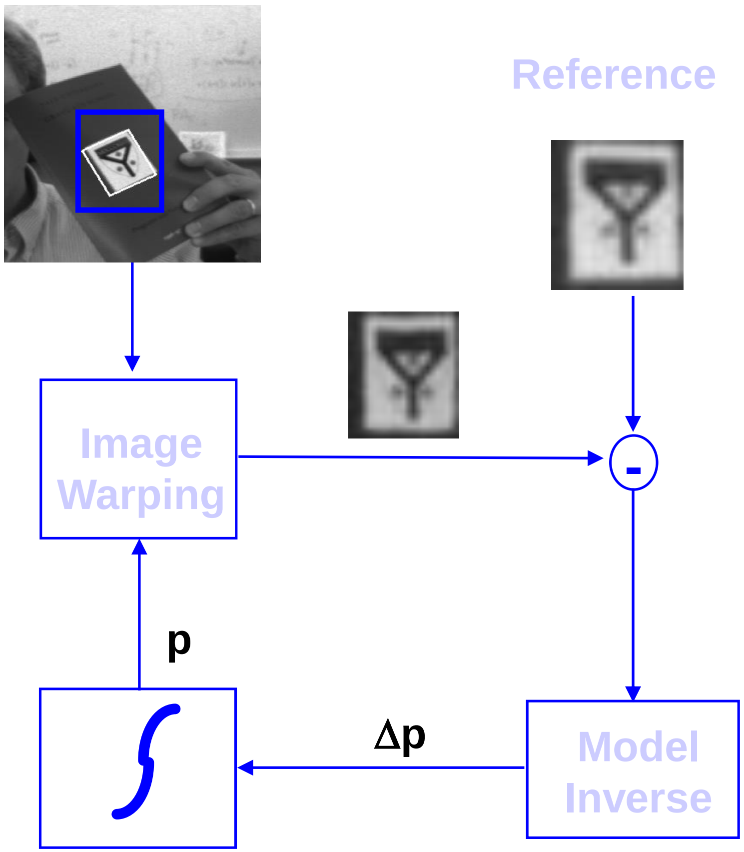
Incremental Estimation: From I_0, I_{t+1} and p_t compute Δp_{t+1}

$$\boxed{\| I_0 - g(I_{t+1}, p_{t+1}) \|^2 \implies \min}$$

Visual Tracking = Visual Stabilization

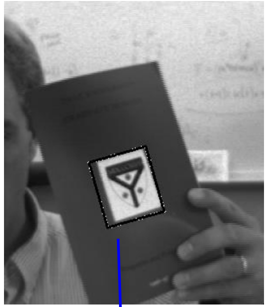
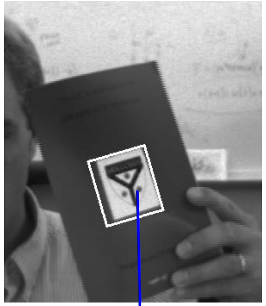
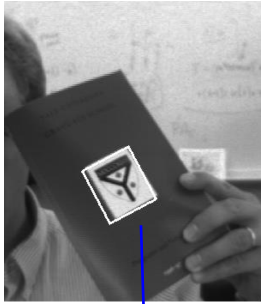
Tracking Cycle

- Prediction
  - Prior states predict new appearance
- Image warping
  - Generate a “normalized view”
- Model inverse
  - Compute error from nominal
- State integration
  - Apply correction to state



Stabilization tracking: Planar Case

Planar Object +
Linear camera => Affine motion model: $u'_i = A u_i + d$

Subtract these:  -  -  = nonsense

Warping

But these:  -  -  = small variation

$$I_t = g(p_t, I_0)$$

Registration based Tracking

$$\mathbf{p_t} = \underset{\mathbf{p}}{\operatorname{argmax}}\ f(\mathbf{I_0}(\mathbf{x}), \mathbf{I_t}(\mathbf{w}(\mathbf{x}, \mathbf{p})))$$

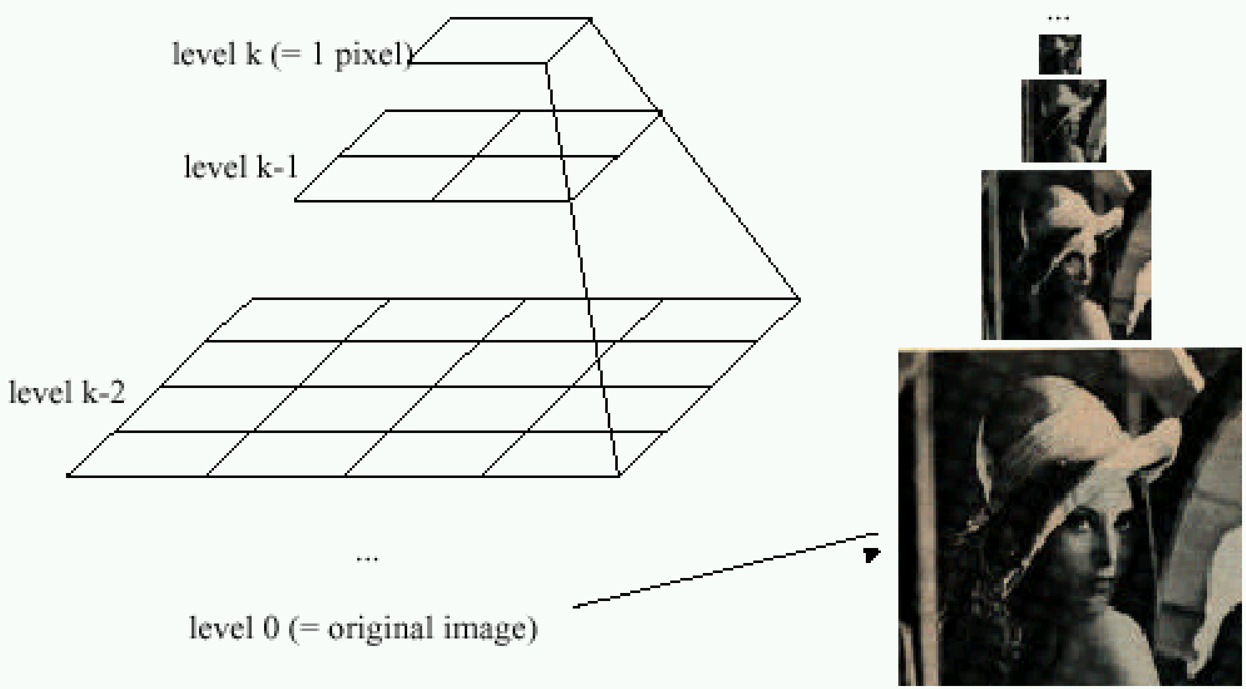
Motivation

- Learning/detection based trackers are not suitable for tasks requiring **fast** and **high precision** tracking
  - Visual Servoing
  - Virtual reality
  - SLAM

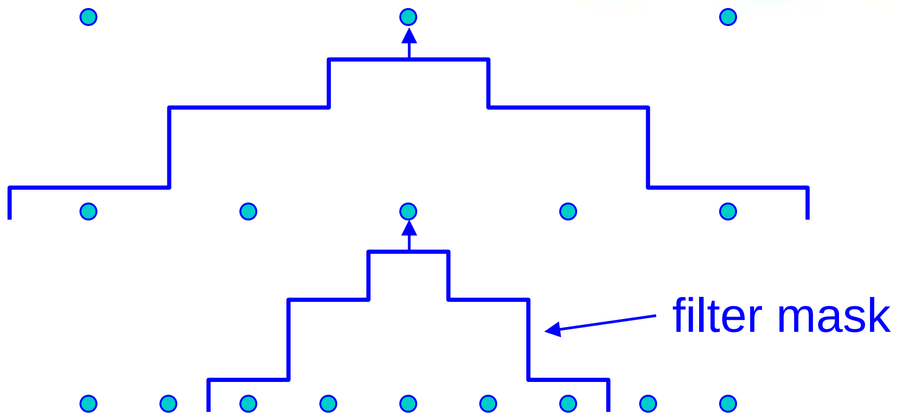
Image Pyramids

Image Pyramids

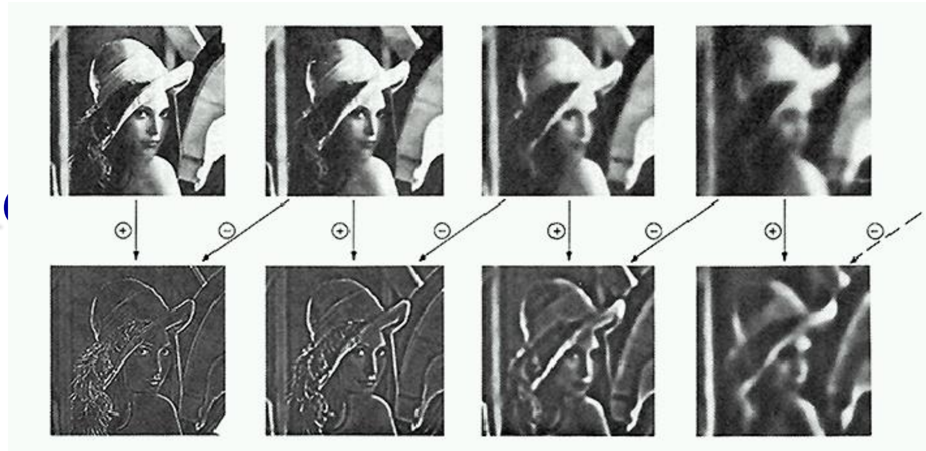
Idea: Represent NxN image as a "pyramid" of
$1\text{x}1$, $2\text{x}2$, $4\text{x}4$,…, $2^k\text{x}2^k$ images (assuming $N=2^k$)



Pyramid Creation



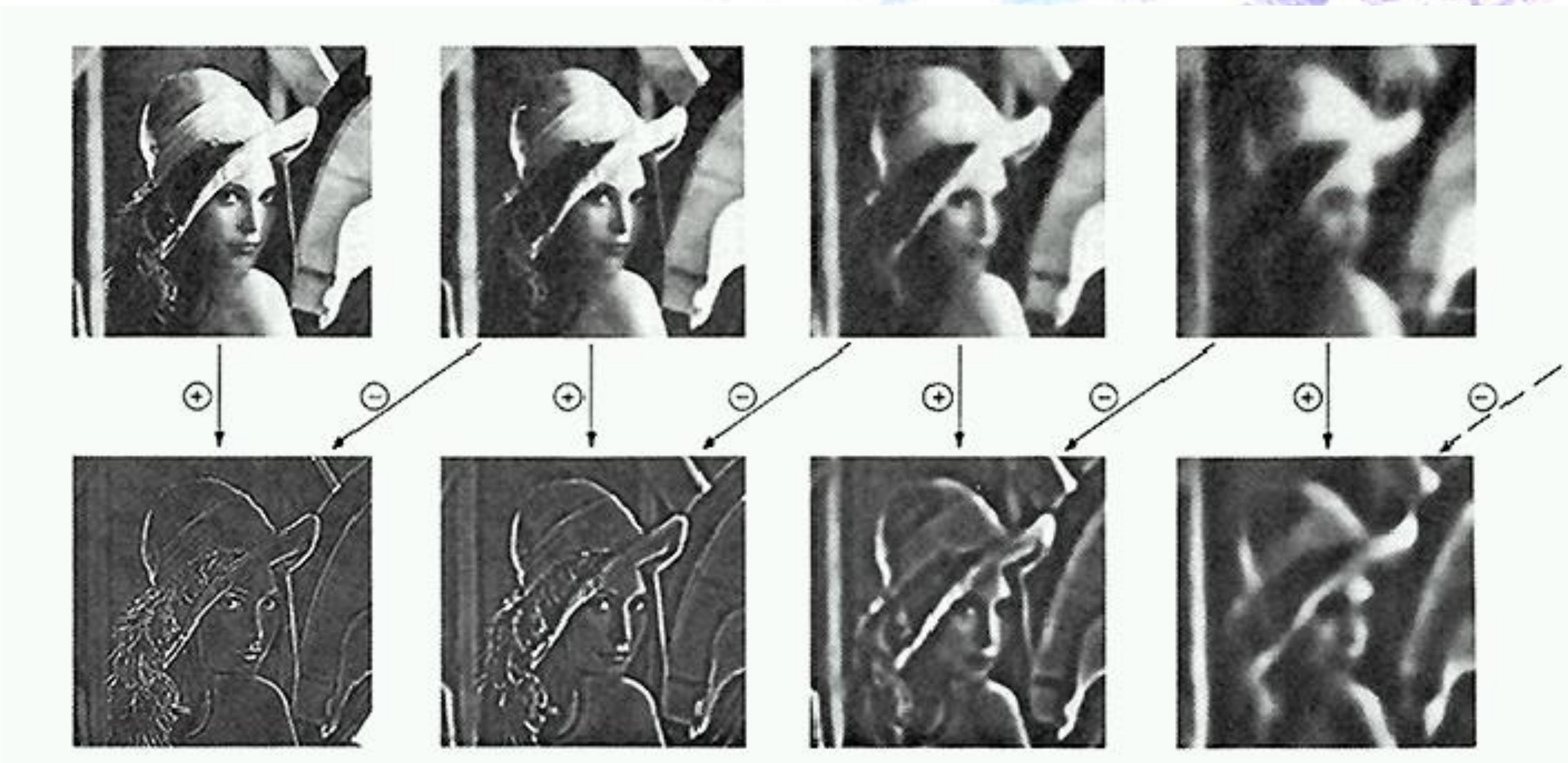
"Gaussian" Pyramid



- "Laplacian" Pyramid
  - Created from Gaussian pyramid by subtraction $L_l = G_l - \text{expand}(G_{l+1})$

Octaves in the Spatial Domain

Lowpass Images



- Bandpass Images

Pyramids

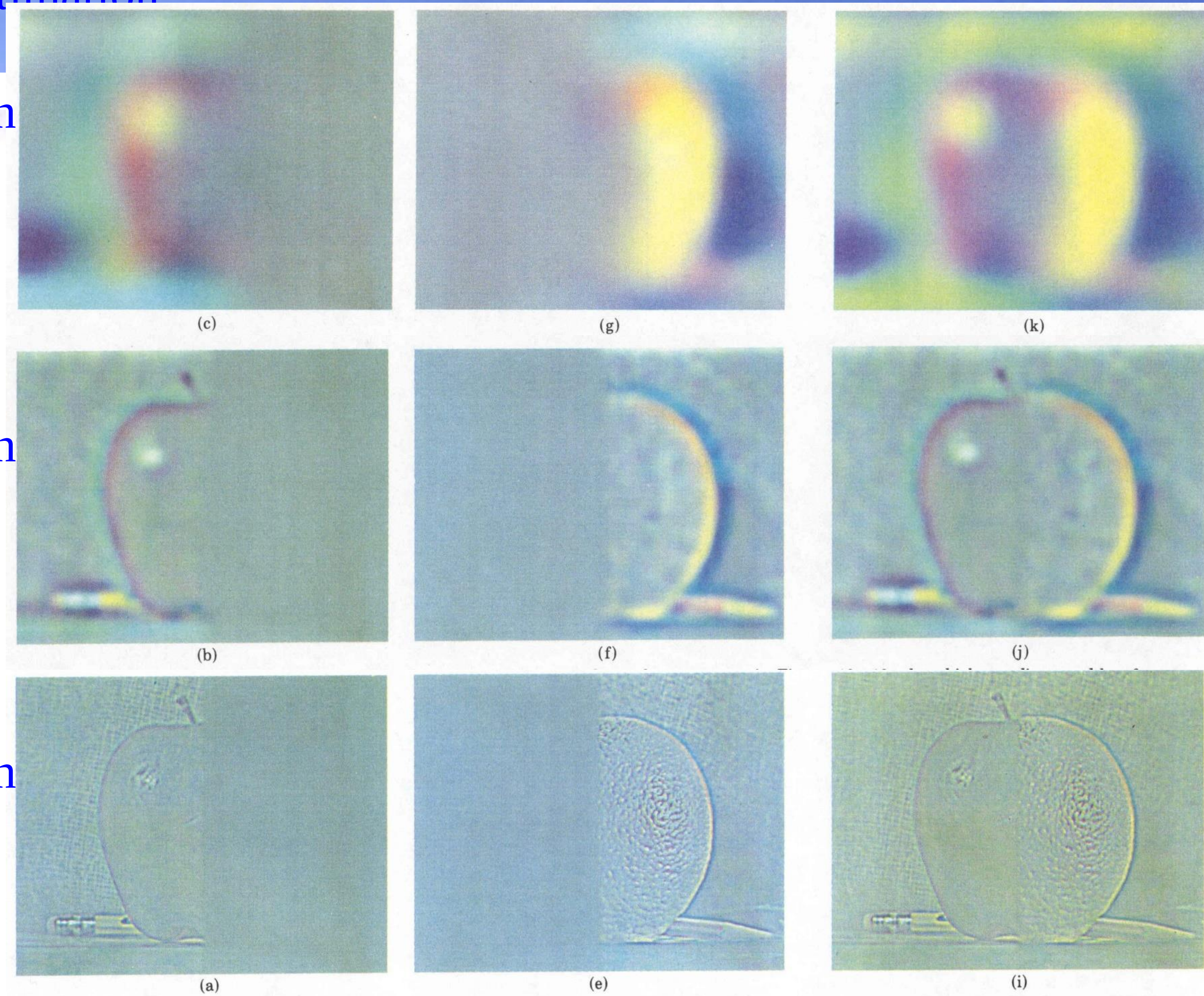
- Advantages of pyramids
  - Faster than Fourier transform
  - Avoids “ringing” artifacts
- Many applications
  - small images faster to process
  - good for multiresolution processing
  - compression
  - progressive transmission
- Known as “mip-maps” in graphics community
- Precursor to wavelets
  - Wavelets also have these advantages

Motion estimation

Laplacian level 4

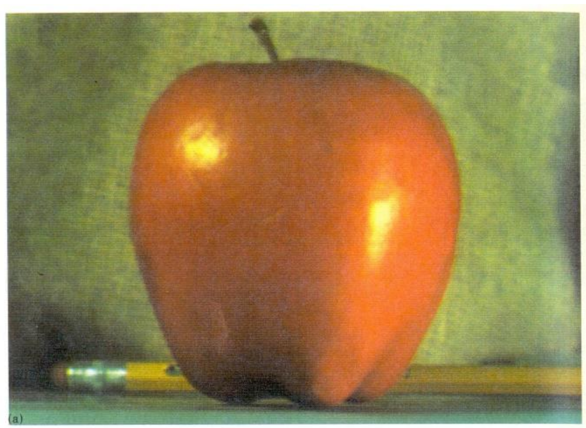
Laplacian level 2

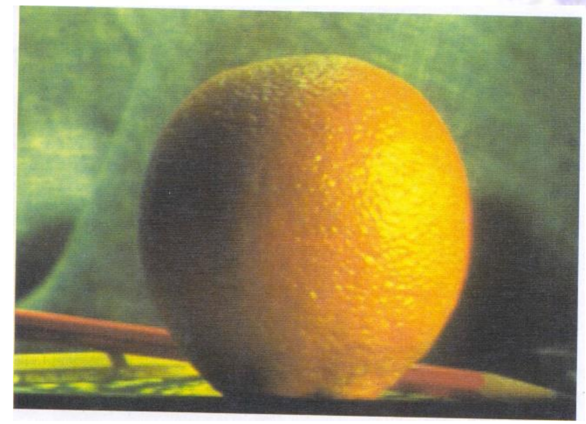
Laplacian level 0

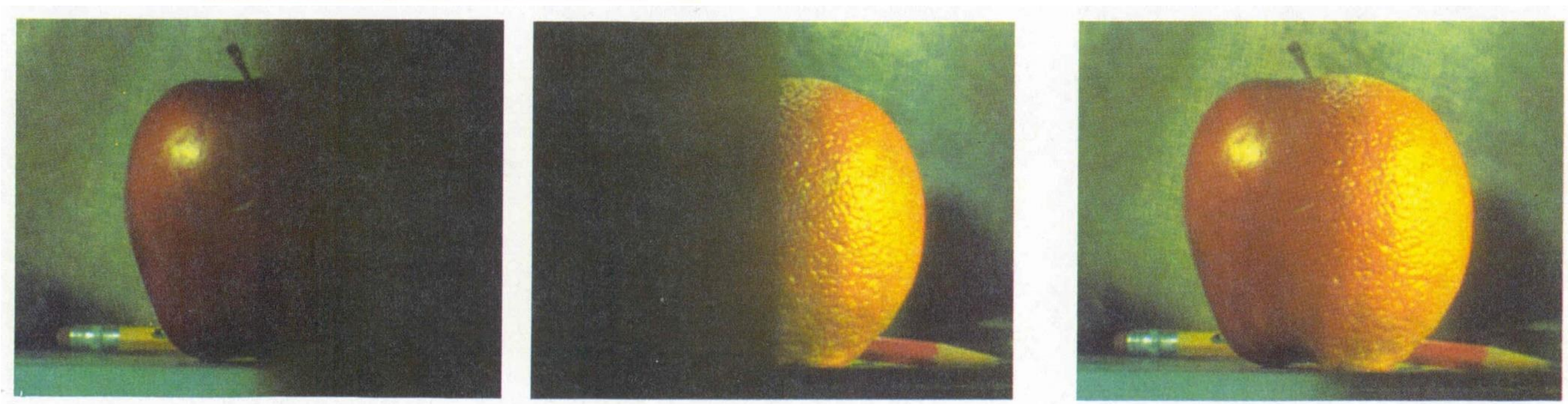


left pyramid   right pyramid   blended pyramid

Pyramid Blending







(d) (h) (l)

Image Warping

Parametric (global) warping

- Examples of parametric warps:

translation

rotation

aspect

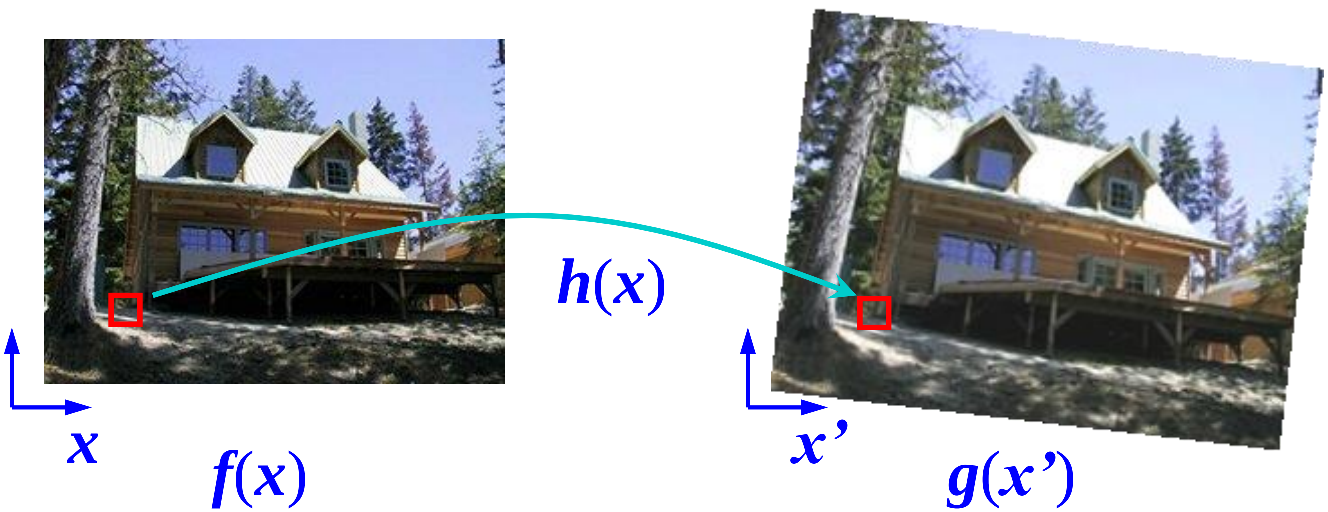
affine

perspective

cylindrical

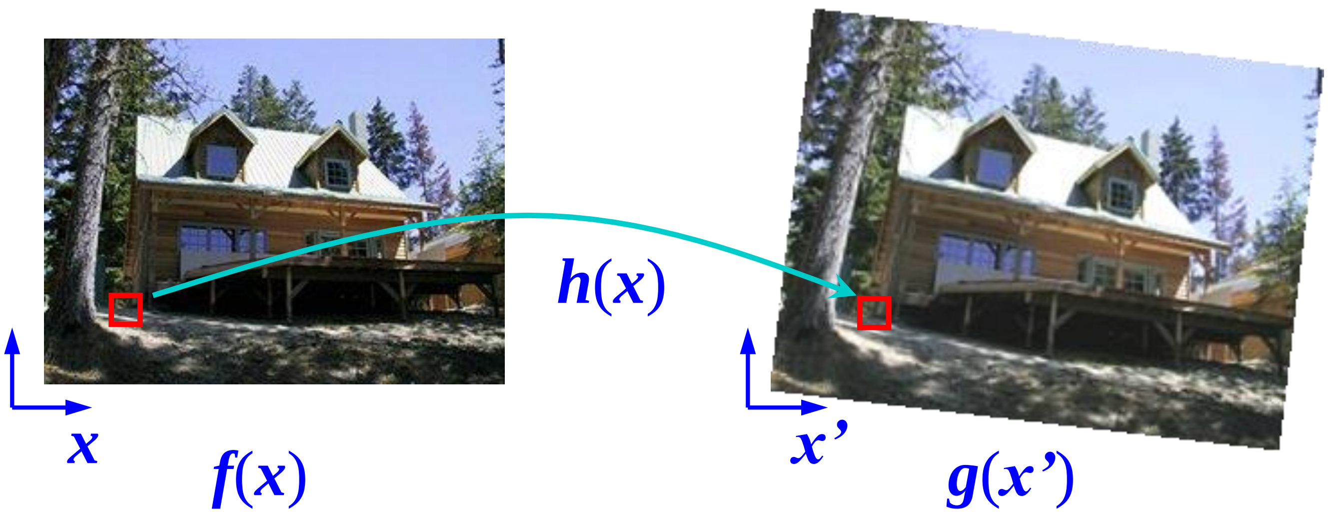
Image Warping

- Given a coordinate transform $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ and a source image $\boldsymbol{f}(\boldsymbol{x})$, how do we compute a transformed image $\boldsymbol{g}(\boldsymbol{x'}) = \boldsymbol{f}(\boldsymbol{h}(\boldsymbol{x}))$?



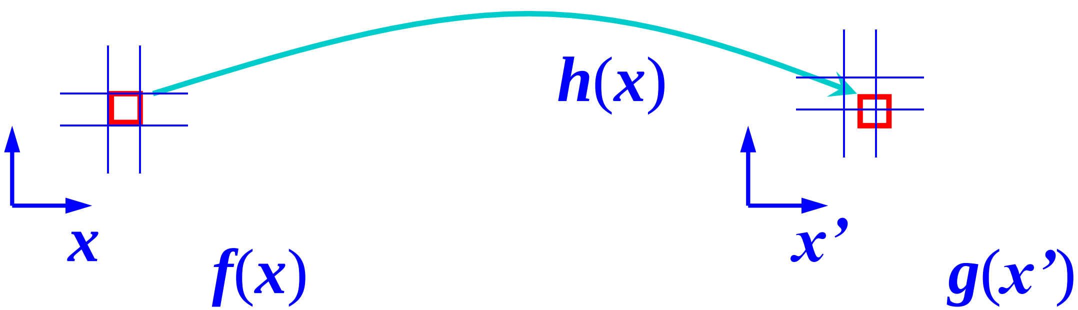
Forward Warping

- Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
  - What if pixel lands “between” two pixels?



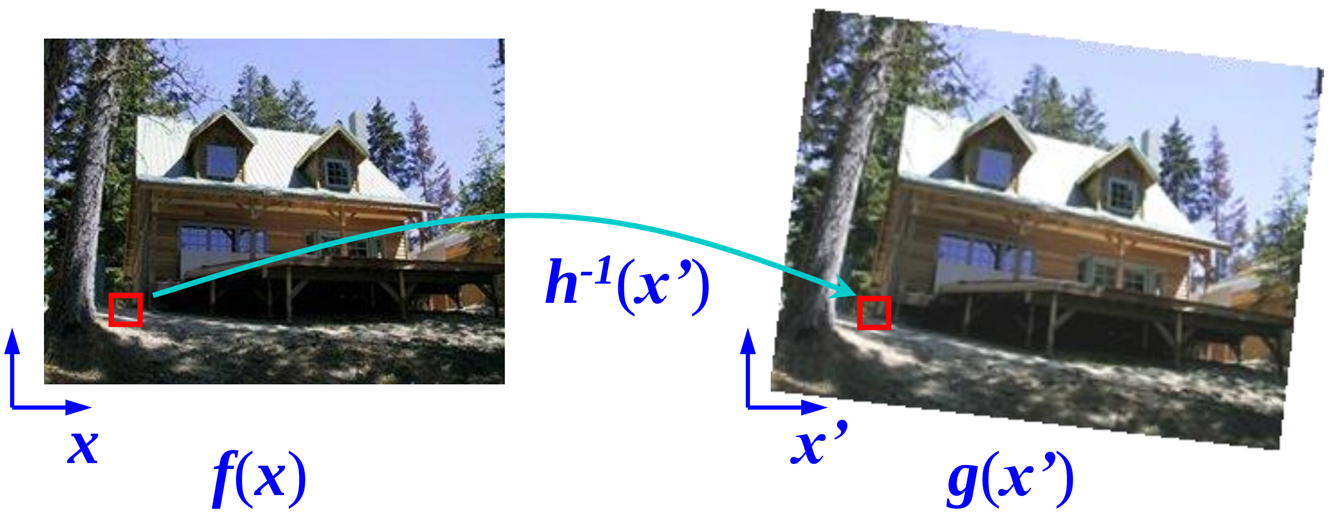
Forward Warping

- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x'})$
  - What if pixel lands “between” two pixels?
  - Answer: add “contribution” to several pixels, normalize later (*splatting*)



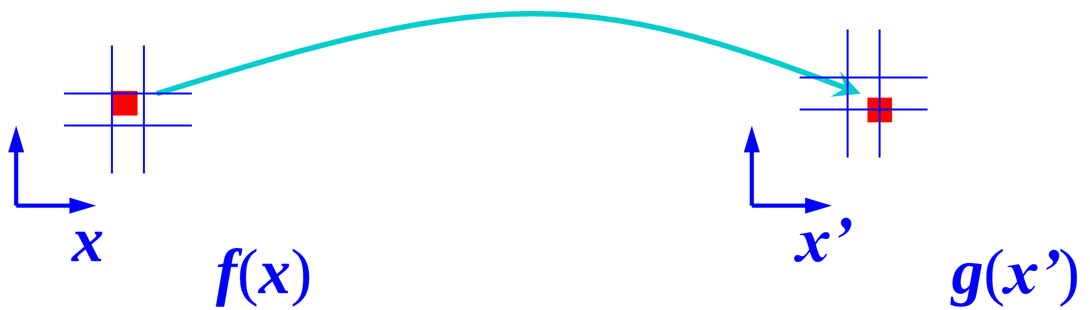
Inverse Warping

- Get each pixel $\boldsymbol{g}(\boldsymbol{x}')$ from its corresponding location $\boldsymbol{x} = \boldsymbol{h}^{-1}(\boldsymbol{x}')$ in $\boldsymbol{f}(\boldsymbol{x})$
  - What if pixel comes from “between” two pixels?



Inverse Warping

- Get each pixel $g(x')$ from its corresponding location $x = h^{-1}(x')$ in $f(x)$
  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* (*prefiltered*) source image

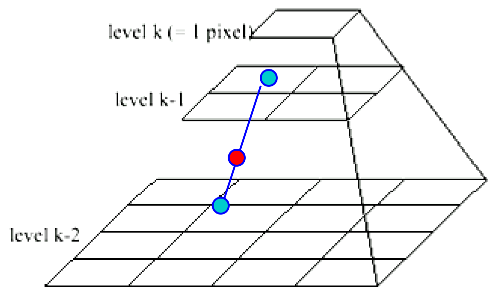


Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic (interpolating)
  - sinc / FIR
- Needed to prevent “jaggies” and “texture crawl” (see demo)

Prefiltering

- Essential for *downsampling* (*decimation*) to prevent *aliasing*
- MIP-mapping [Williams'83]:
  1. build pyramid (but what decimation filter?)
     - block averaging
     - Burt & Adelson (5-tap binomial)
     - 7-tap wavelet-based filter (better)
  2. *trilinear* interpolation
     - bilinear within each 2 adjacent levels
     - linear blend *between* levels (determined by pixel size)

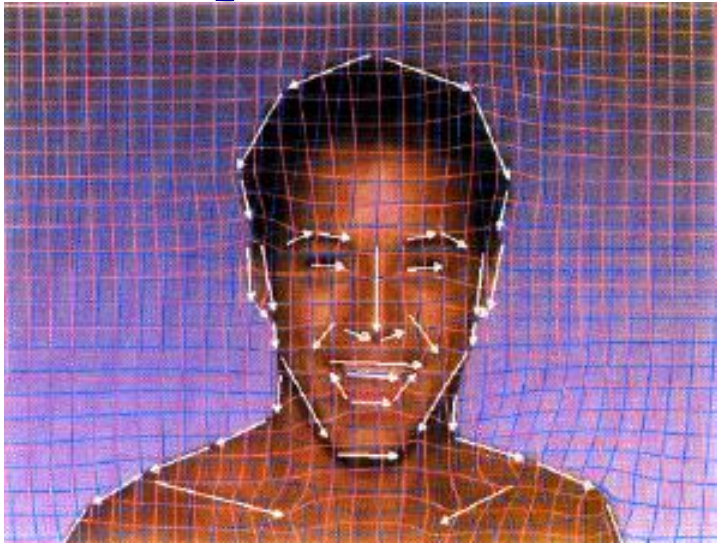


Prefiltering

- Essential for *downsampling* (*decimation*) to prevent *aliasing*
- Other possibilities:
  - summed area tables
  - elliptically weighted Gaussians (EWA) [Heckbert'86]

Image Warping – non-parametric

- Specify more detailed warp function



- Examples:
  - splines
  - triangles
  - optical flow (per-pixel motion)

Image Warping – non-parametric

- Move control points to specify spline warp

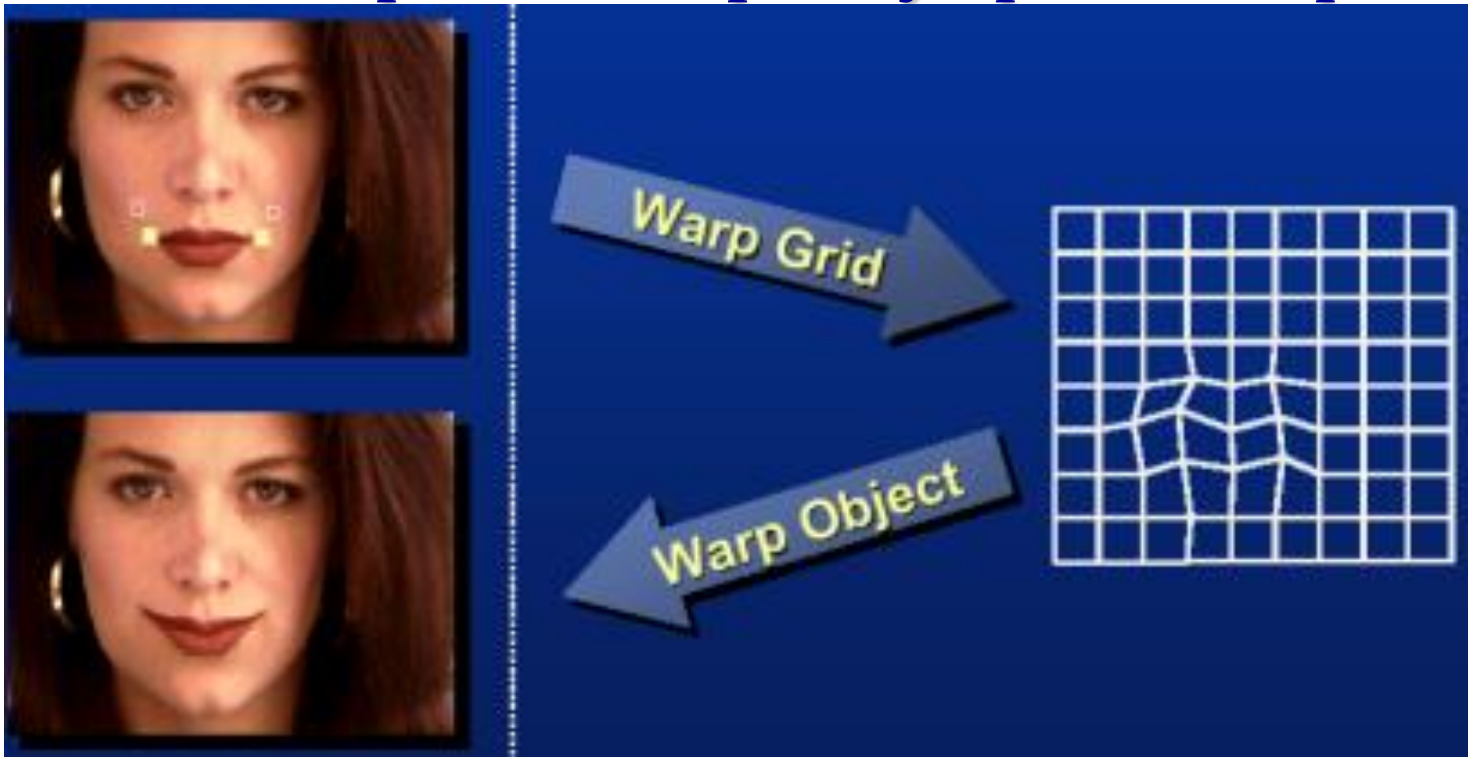
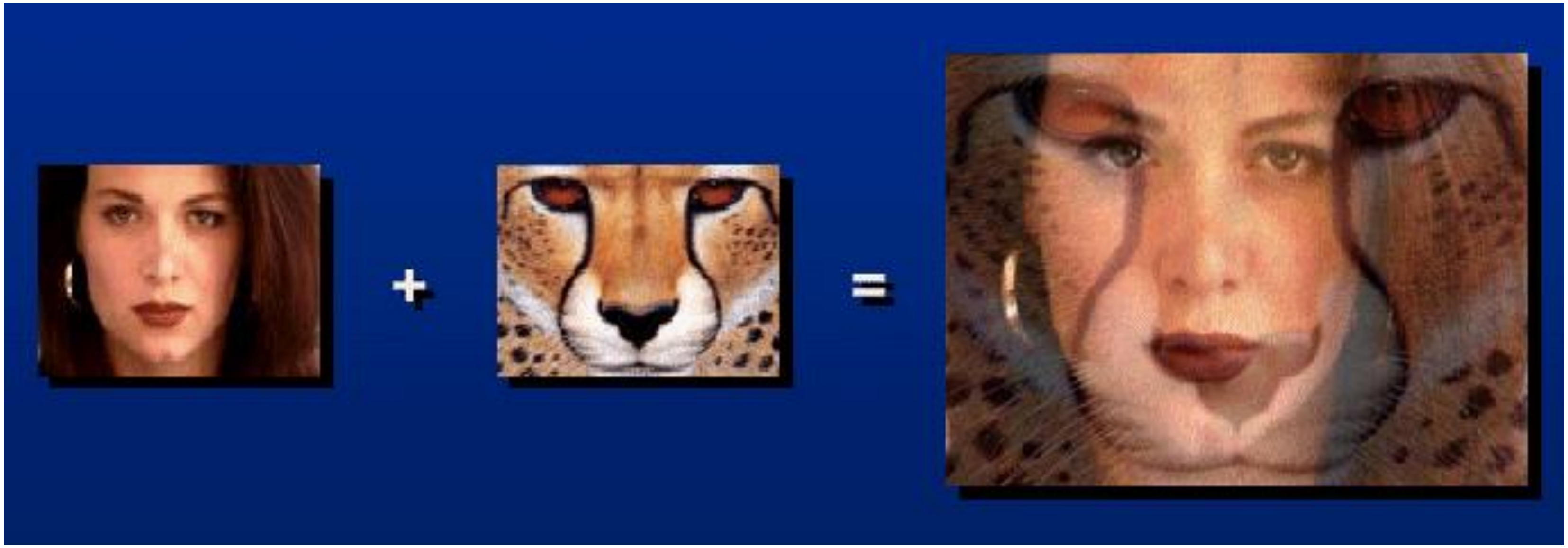


Image Morphing

Image Morphing

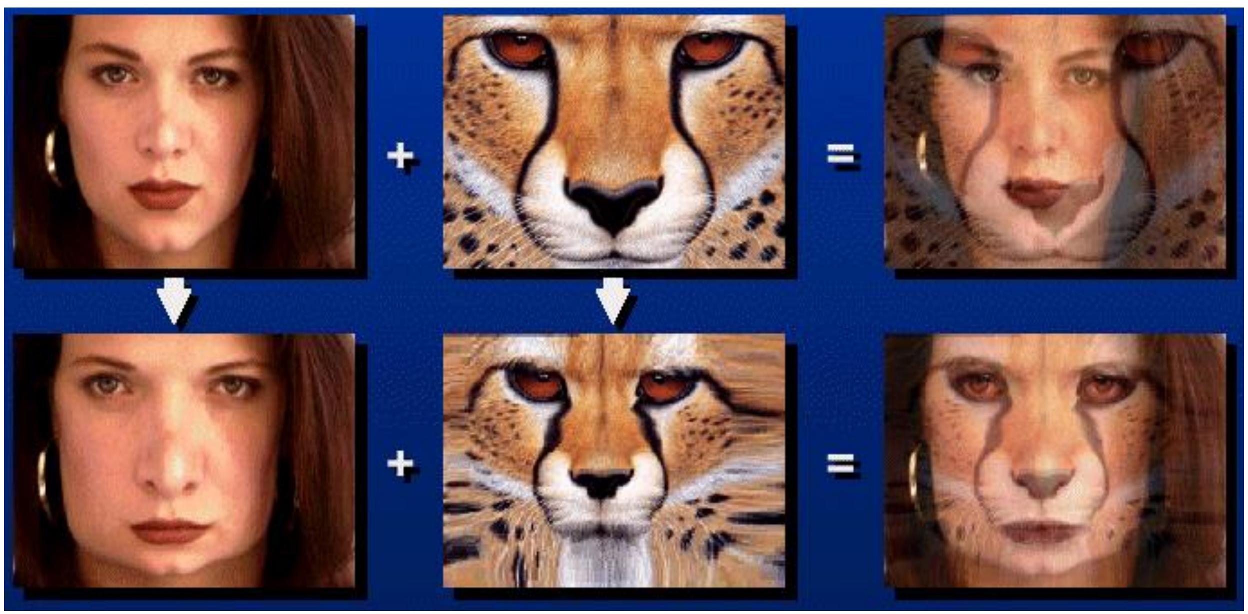
- How can we *in-between* two images?
  1. Cross-dissolve



(all examples from [Gomes *et al.*’99])

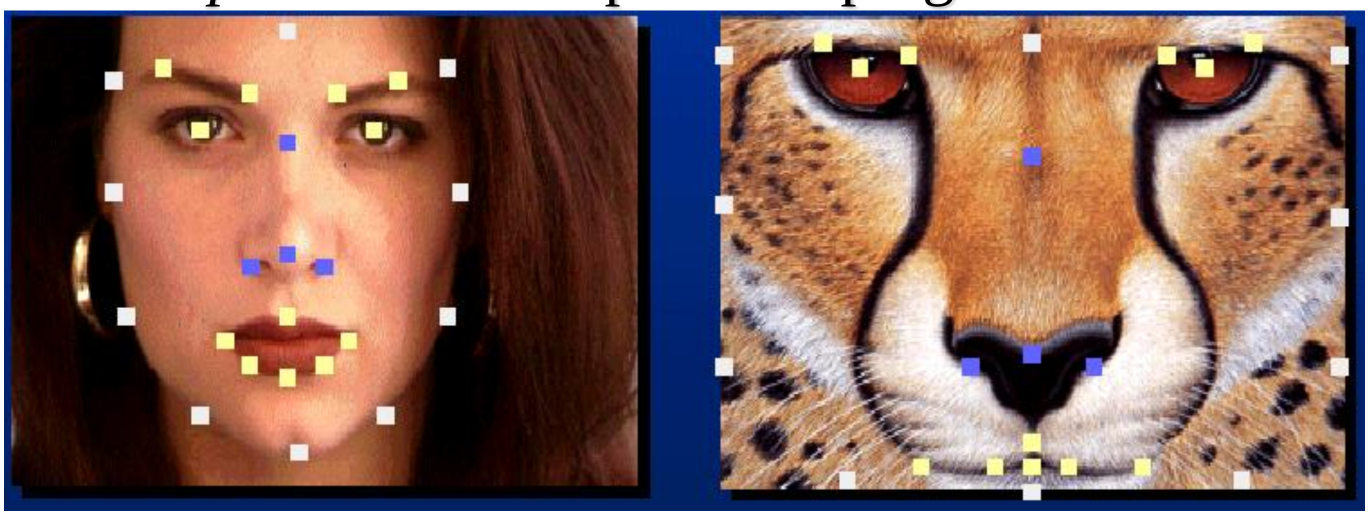
Image Morphing

- How can we *in-between* two images?
  2. Warp then cross-dissolve = *morph*



Warp specification

- How can we specify the warp?
  1. Specify corresponding *points*
     - *interpolate* to a complete warping function



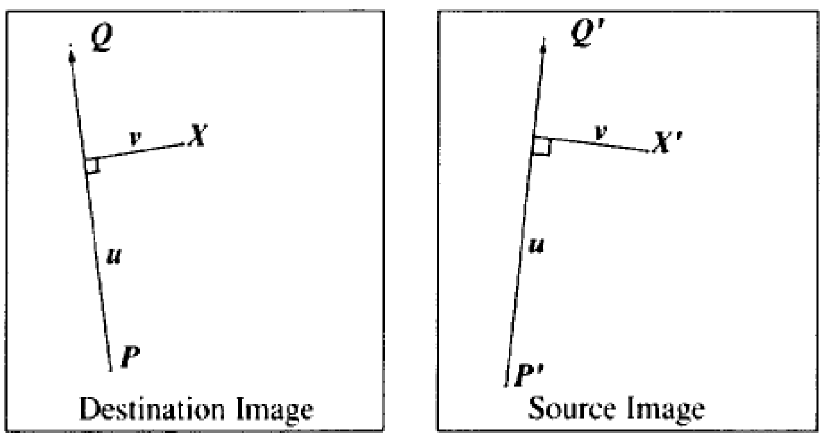
- Nielson, *Scattered Data Modeling*, IEEE CG&A'93]

Warp specification

- How can we specify the warp?
  2. Specify corresponding *vectors*
     - *interpolate* to a complete warping function

Warp specification

- How can we specify the warp?
  2. Specify corresponding *vectors*
     - *interpolate* [Beier & Neely, SIGGRAPH'92]

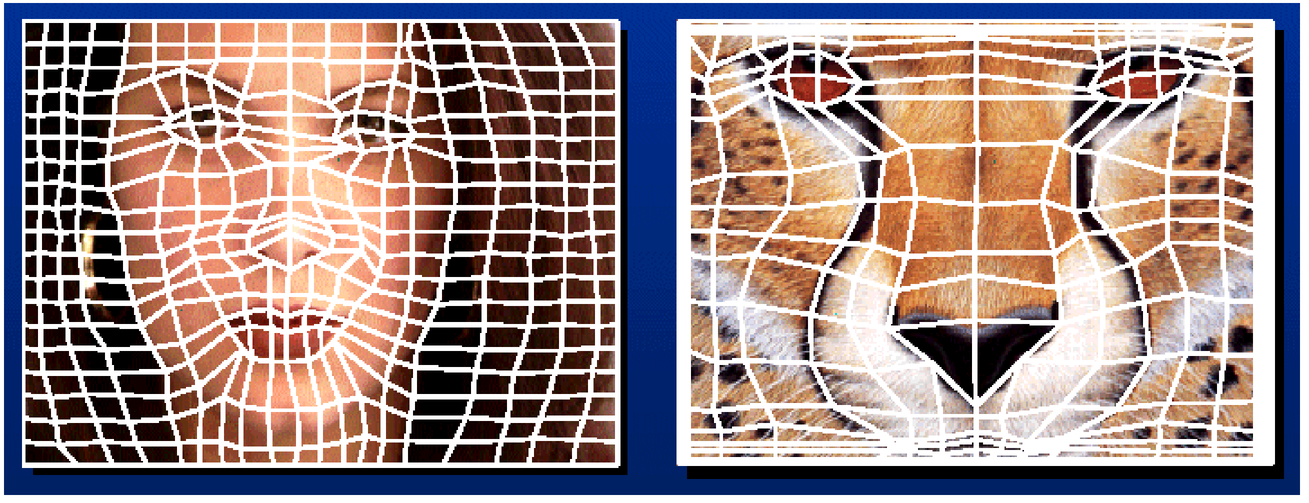



```
For each pixel X in the destination
    DSUM = (0,0)
    weightsum = 0
    For each line Pi Qi
        calculate u,v based on Pi Qi
        calculate X'i based on u,v and Pi'Qi'
        calculate displacement Di = Xi' - Xi for this line
        dist = shortest distance from X to Pi Qi
        weight = (length^p / (a + dist))^b
        DSUM += Di * weight
        weightsum += weight
    X' = X + DSUM / weightsum
    destinationImage(X) = sourceImage(X')
```

Warp specification

- How can we specify the warp?

3. Specify corresponding *spline control points*
    - *interpolate* to a complete warping function



Final Morph Result

Intro to Registration Tracking

p_t

I_0 $\qquad$ I_t

Variability model: $\quad I_t = g(I_0, p_t)$

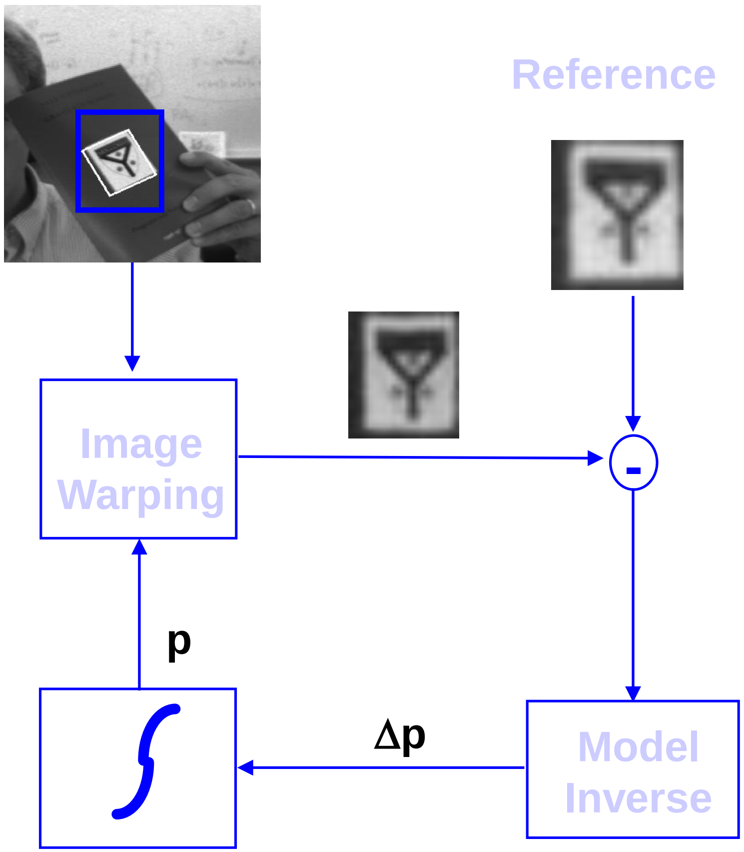
Incremental Estimation: $\quad$ From I_0, I_{t+1} and p_t compute Δp_{t+1}

$$\boxed{\| I_0 - g(I_{t+1}, p_{t+1}) \|^2 \Longrightarrow \min}$$

Visual Tracking = Visual Stabilization

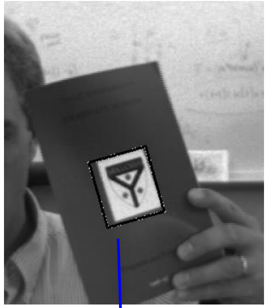
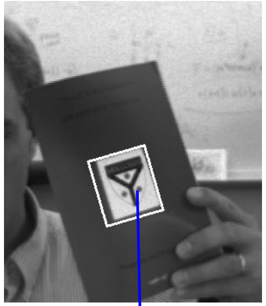
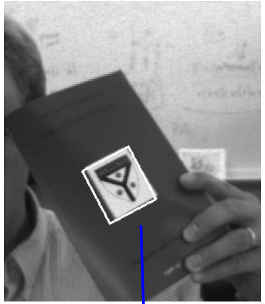
Tracking Cycle

- Prediction
  - Prior states predict new appearance
- Image warping
  - Generate a “normalized view”
- Model inverse
  - Compute error from nominal
- State integration
  - Apply correction to state



Stabilization tracking: Planar Case

Planar Object +
Linear camera => Affine motion model: $u'_i = A u_i + d$

Subtract these:  -  -  = nonsense

Warping

But these:  -  -  = small variation

$$I_t = g(p_t, I_0)$$

Registration based Tracking

$$\mathbf{p_t} = \underset{\mathbf{p}}{\operatorname{argmax}}\ f(\mathbf{I_0}(\mathbf{x}), \mathbf{I_t}(\mathbf{w}(\mathbf{x}, \mathbf{p})))$$

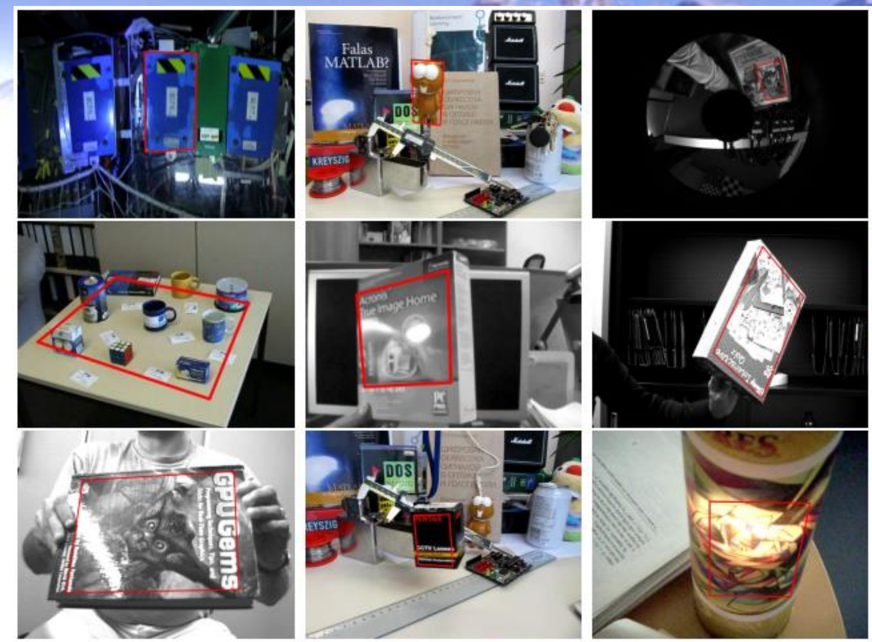
Motivation

- Learning/detection based trackers are not suitable for tasks requiring **fast** and **high precision** tracking
  - Visual Servoing
  - Virtual reality
  - SLAM

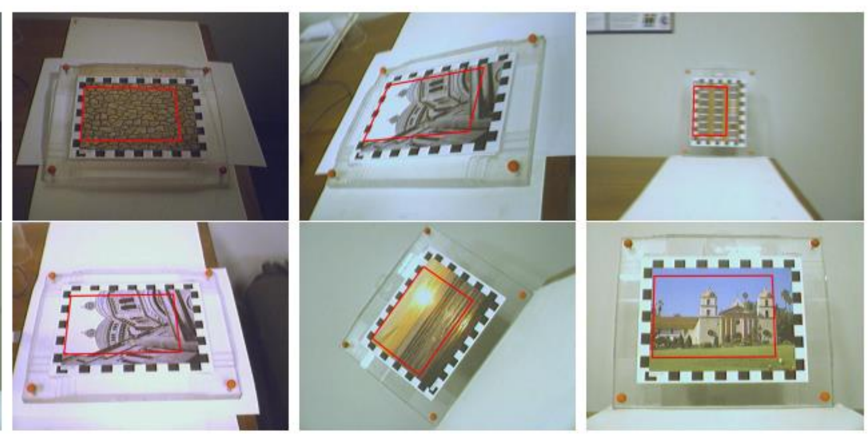
Evaluation Benchmarks

TMT

PAMI

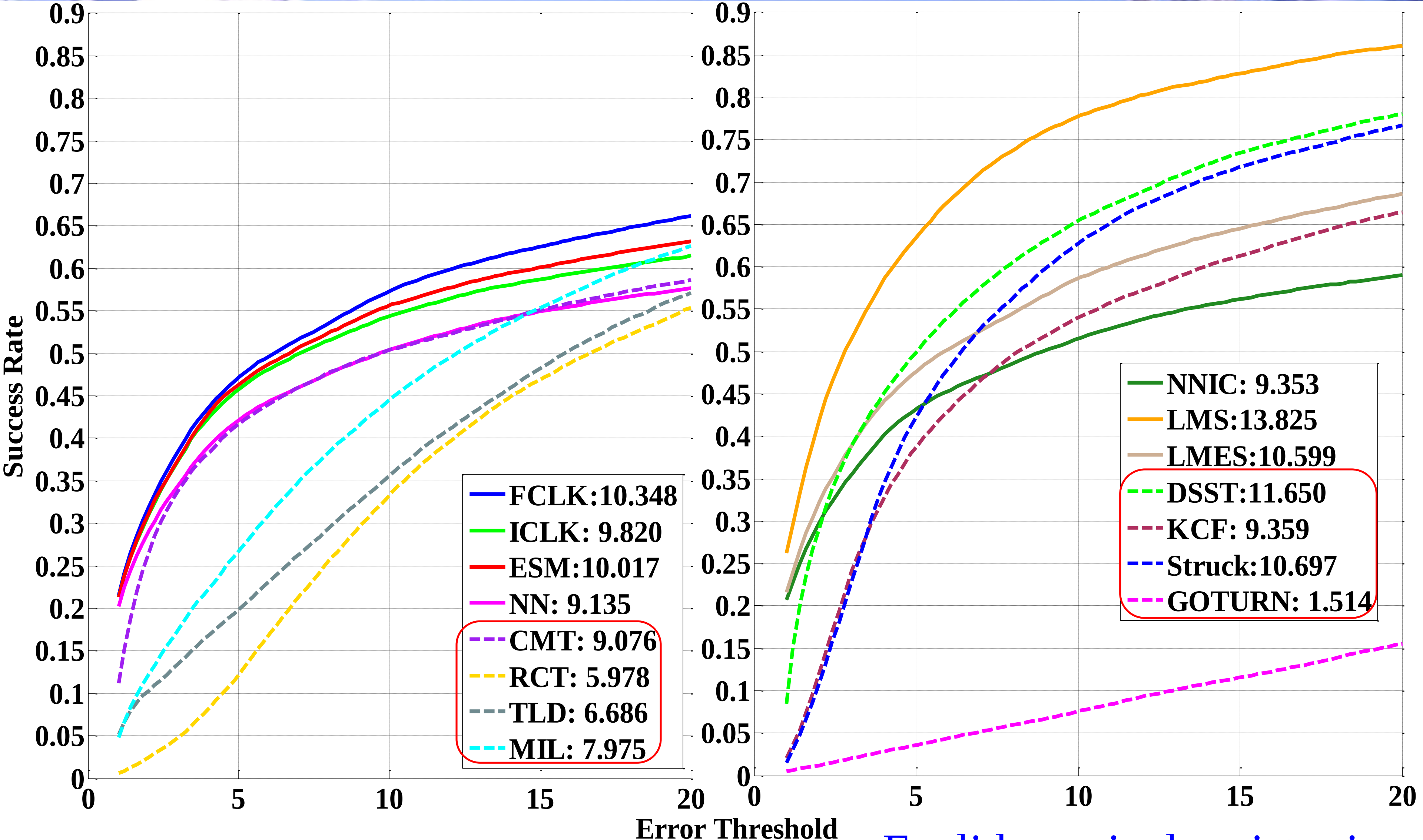


UCSB



LinTrack

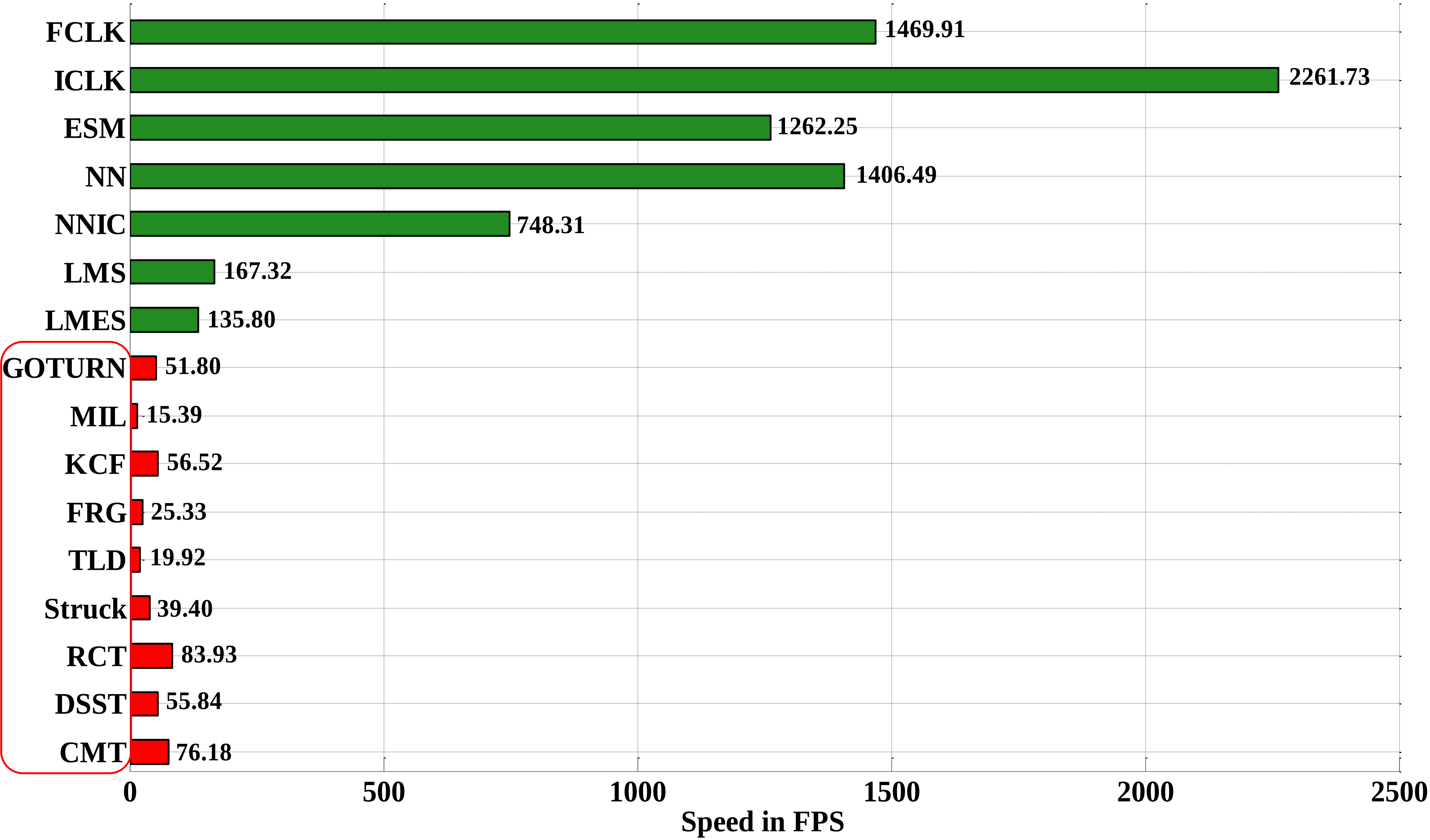
Results: Learning vs. 2DOF Registration Based Trackers (Accuracy)



Euclidean pixel registration e

RBT vs Learn vs GoTurn

Results: Learning vs. 2DOF Registration Based Trackers (Speed)



References

- S. Baker and I. Matthews, "*Equivalence and Efficiency of Image Alignment Algorithms*", CVPR 2001
- S. Benhimane and E. Malis, "*Real-time image-based tracking of planes using efficient second-order minimization* ", IROS 2004
- A. Dame and E. Marchand, "*Accurate Real-time Tracking Using Mutual Information*", ISMAR 2010
- T. Dick, C. Perez, A. Shademan and M. Jagersand, "*Realtime Registration-Based Tracking via Approximate Nearest Neighbor Search*", RSS 2013
- S. Gauglitz, T. Hollerer and M. Turk. "*Evaluation of interest point detectors and feature descriptors for visual tracking*", IJCV 2011
- J. Kwon, H. S. Lee, F. C. Park, and K. M. Lee, "*A Geometric Particle Filter for Template-Based Visual Tracking*", TPAMI 2014
- R. Richa, R. Sznitman, R. Taylor and G. Hager, "*Visual Tracking Using the Sum of Conditional Variance*", IROS 2011

References (cont'd)

- R. Richa, R. Sznitman and G. Hager, "*Robust Similarity Measures for Gradient-based Direct Visual Tracking*", CIRL Technical Report 2012
- R. Richa, M. Souza, G. Scandaroli, E. Comunello and A. Wangenheim, "*Direct visual tracking under extreme illumination variations using the sum of conditional variance*", ICIP 2014
- A. Roy, X. Zhang, N. Wolleb, C. P. Quintero and M. Jagersand, "*Tracking Benchmark and Evaluation for Manipulation Tasks* ", ICRA 2015
- L. Ruthotto, "*Mass-preserving registration of medical images*", Thesis 2010
- G. G. Scandaroli, M. Meilland and R. Richa, "*Improving NCC-Based Direct Visual Tracking*", ECCV 2012
- A. Singh and M. Jagersand, "*Modular Tracking Framework: A Unified Approach to Registration based Tracking*", arXiv:1602.09130, 2016