3D Vision

Marc Pollefeys
Daniel Barath

Spring 2024

ETH

Schedule

Feb 19	Introduction
Feb 26	Geometry, Camera Model, Calibration
Mar 4	**Guest lecture + Features, Tracking / Matching**
Mar 11	**Project Proposals by Students**
Mar 18	3DV conference
Mar 25	Structure from Motion (SfM) + papers
Apr 1	Easter break
Apr 8	Dense Correspondence (stereo / optical flow) + papers
Apr 15	Bundle Adjustment & SLAM + papers
Apr 22	**Student Midterm Presentations**
Apr 29	Multi-View Stereo & Volumetric Modeling + papers
May 6	3D Modeling with Depth Sensors + papers
May 13	Guest lecture + papers
May 20	Holiday

3D Vision – Class 3

Features & Correspondences

feature extraction, image descriptors,
feature matching, feature tracking

Chapters 4, 8 in Szeliski's Book
[Shi & Tomasi, Good Features to Track, CVPR 1994]

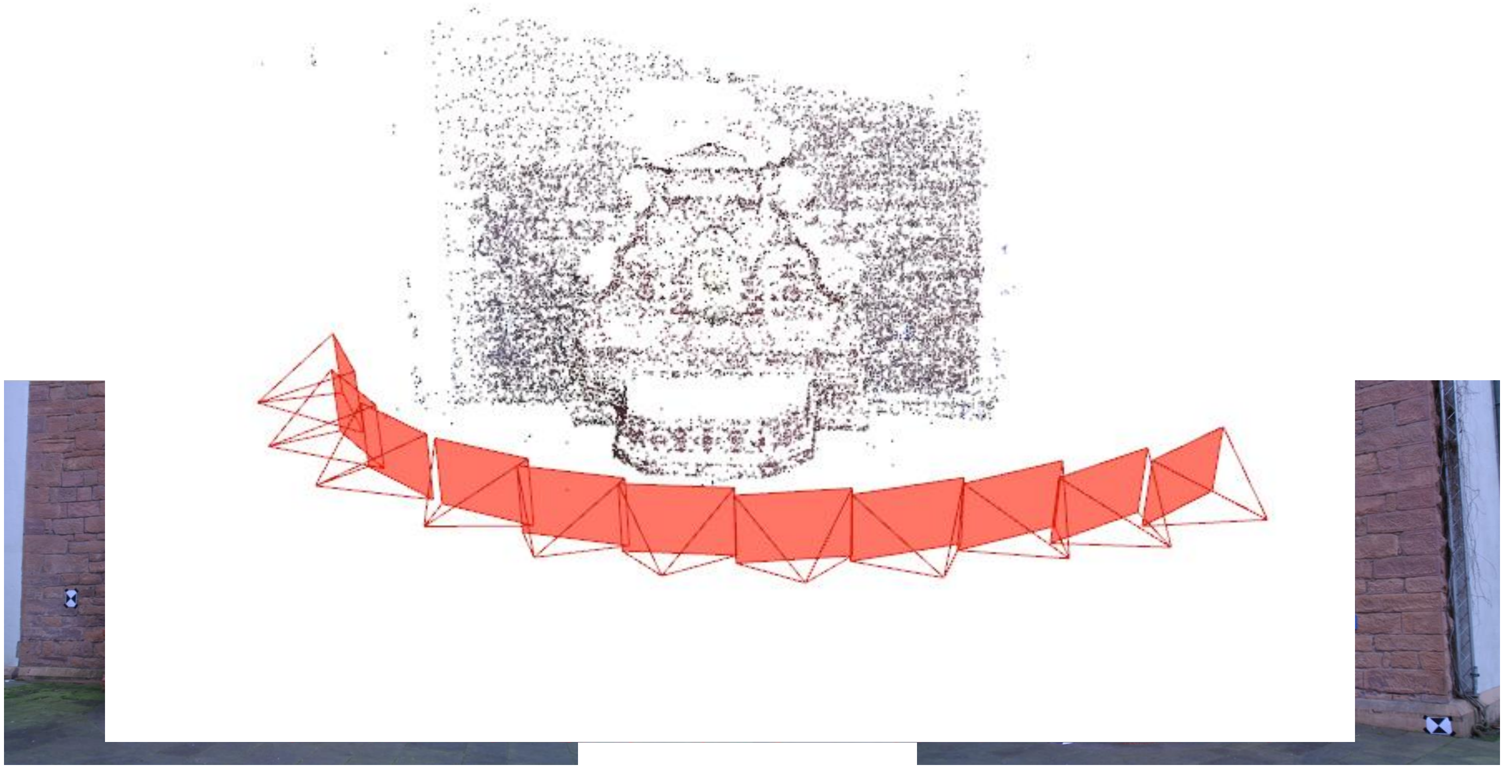
Overview

- **Local Features**

- Invariant Feature Detectors
- Invariant Descriptors & Matching

- Feature Tracking

Importance of Features



Features are key component of many 3D Vision algorithms

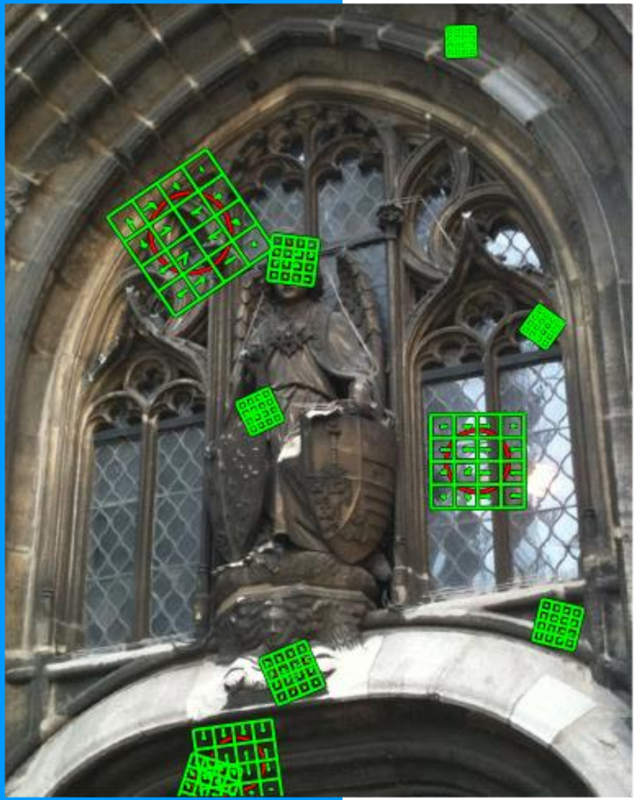
Importance of Features

Rome dataset

74,394 images

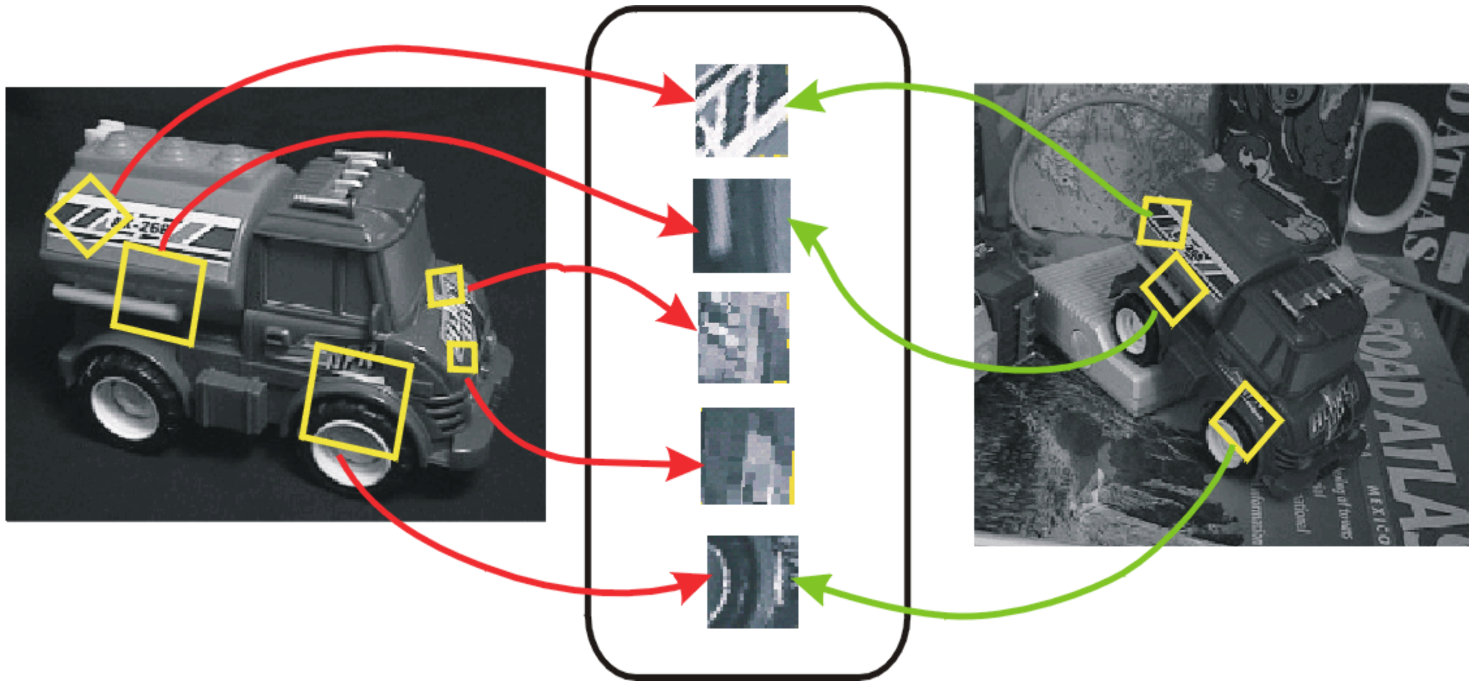
Schönberger & Frahm, Structure-From-Motion Revisited, CVPR 2016

Feature Detectors & Descriptors



- **Detector**: Find salient structures
  - Corners, blob-like structures, ...
  - **Keypoints** should be **repeatable**
- **Descriptor**: Compact representation of image region around keypoint
  - Describes patch around keypoints
  - Establish matches between images by comparing descriptors

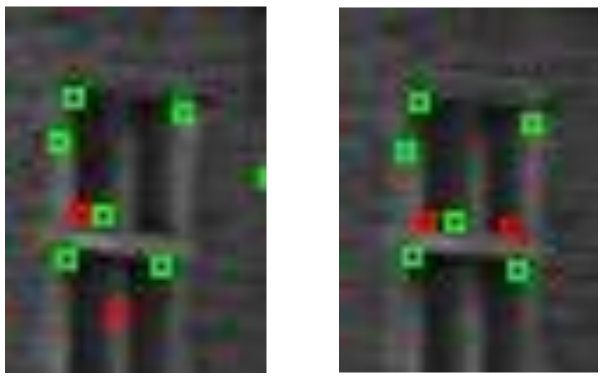
Feature Detectors & Descriptors



(Lowe, Distinctive Image Features From Scale-Invariant Keypoints, IJCV'04)

Feature Matching vs. Tracking

Matching



- Extract features independently
- Match by comparing descriptors

Tracking

- Extract features in first image
- Find same feature in next view

Wide Baseline Matching

- Requirement to cope with larger variations between images
  - Translation, rotation, scaling
  - Perspective foreshortening

  geometric transformations

  - Non-diffuse reflections
  - Illumination

  photometric transformations

Good Detectors & Descriptors?

- What are the properties of good detectors and descriptors?
  - **Invariances** against transformations
- How to design such detectors and descriptors?
- This lecture:
  - Feature detectors & their invariances
  - Feature descriptors, invariances, & matching
  - Feature tracking

Overview

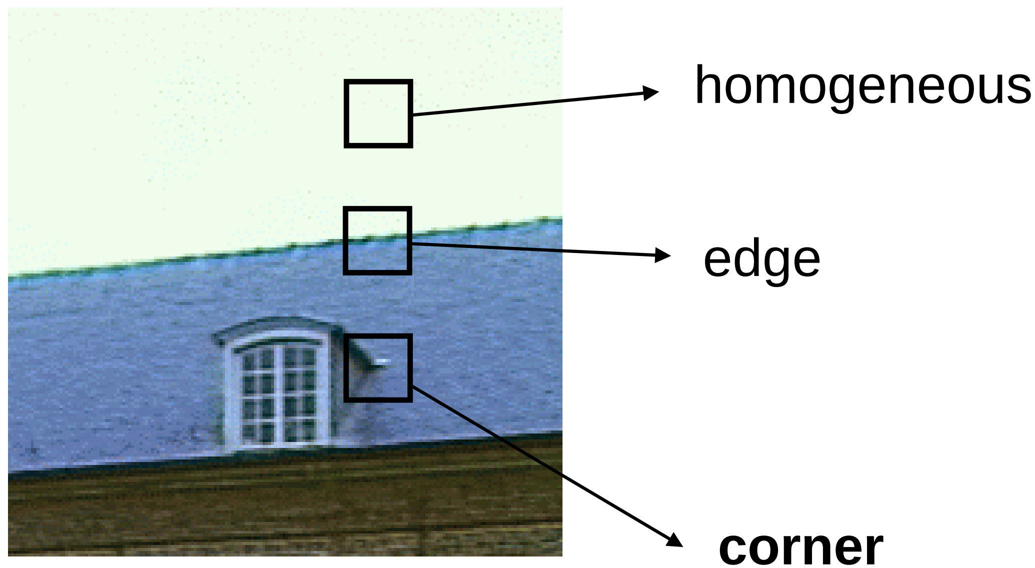
- Local Features Intro
- **Invariant Feature Detectors**
- Invariant Descriptors & Matching
- Feature Tracking

Good Feature Detectors?

- Desirable properties?
  - Precise (sub-pixel perfect) **localization**
  - **Repeatable** detections under
    - Rotation
    - Translation
    - Illumination
    - Perspective distortions
    - …
  - Detect **distinctive** / **salient** structures

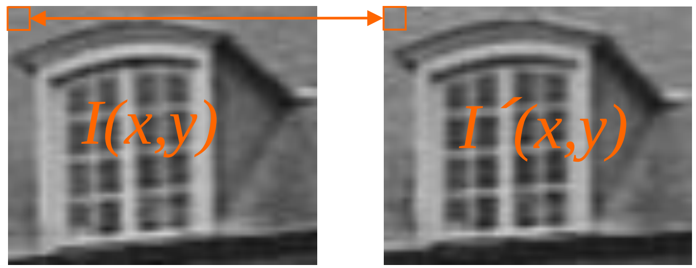
Feature Point Extraction

- Find “distinct” keypoints (local image patches)
- As different as possible from neighbors



Comparing Image Regions

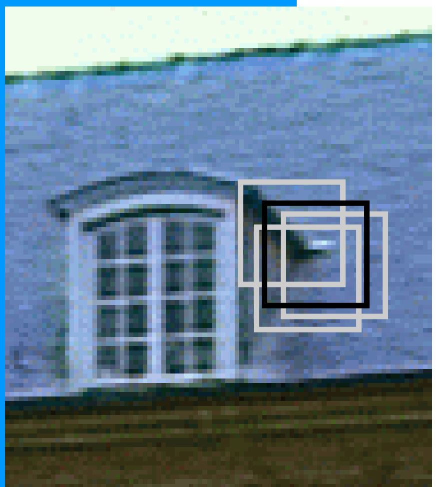
- Compare intensities pixel-by-pixel



- Dissimilarity measure: **Sum of Squared Differences / Distances** (**SSD**)

$$SSD = \sum_{x} \sum_{y} \left[I'(x,y) - I(x,y) \right]^2$$

Finding Stable Features



- Measure uniqueness of candidate
- Approximate SSD for small displacement Δ

$$SSD = \sum_i w(\mathbf{x}_i)\left[I(\mathbf{x}_i + \Delta) - I(\mathbf{x}_i)\right]^2$$

$$\approx \sum_i w(\mathbf{x}_i)\left[I(\mathbf{x}_i) + \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix}\Delta - I(\mathbf{x}_i)\right]^2$$

$$= \sum_i w(\mathbf{x}_i)\Delta^T \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}\Delta = \Delta^T \mathbf{M}\Delta \qquad I_x = \frac{\partial I}{\partial x}$$

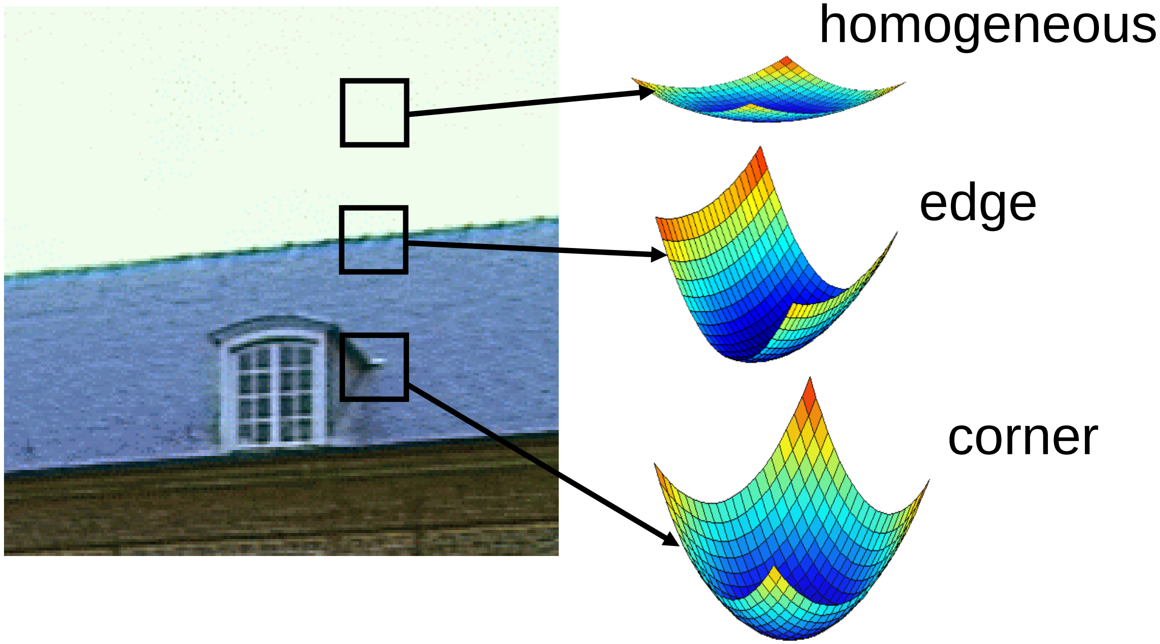
- possible weights

$$w(x,y) = (\mathrm{abs}(x) < \frac{w}{2})(\mathrm{abs}(y) < \frac{w}{2})$$

$$w(x,y) = e^{-\frac{x^2+y^2}{\sigma^2}}$$

Finding Stable Features

$$SSD \approx \Delta^\top \mathbf{M} \Delta$$



Suitable feature positions should maximize

$$\min \Delta^\top \mathbf{M} \Delta \text{ for } \|\Delta\| = 1$$

i.e. maximize ***smallest eigenvalue*** *of M*

ETH

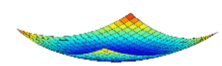
Harris Corner Detector

- Use small local window: $w(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$ $\sigma = 0.7$
- Directly computing eigenvalues λ_1, λ_2 of M is computationally expensive
- Alternative measure for "**cornerness**":

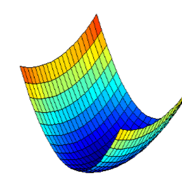
$$R = \det \mathrm{M} - k\,(\mathrm{trace}\mathrm{M})^2$$

$$= \lambda_1 \cdot \lambda_2 - k\,(\lambda_1 + \lambda_2)^2$$

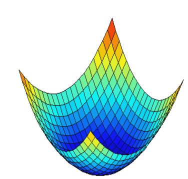
- Homogeneous: λ_1, λ_2 small $\Rightarrow R$ small



- Edge: $\lambda_1 \gg \lambda_2 \approx 0 \Rightarrow R = \lambda_1 \cdot 0 - k\lambda_1^2 < 0$



- Corner: λ_1, λ_2 large $\Rightarrow R$ large



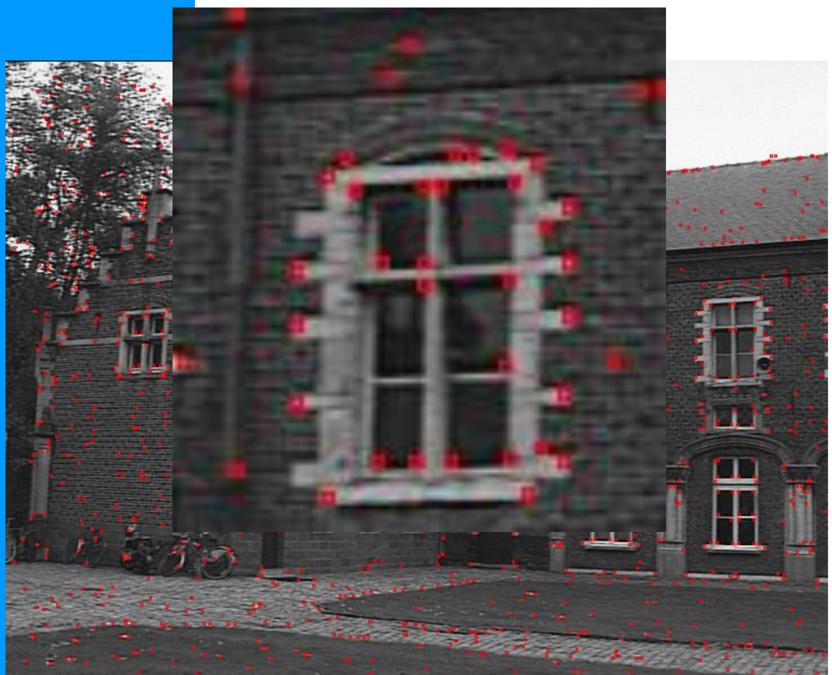
Harris Corner Detector

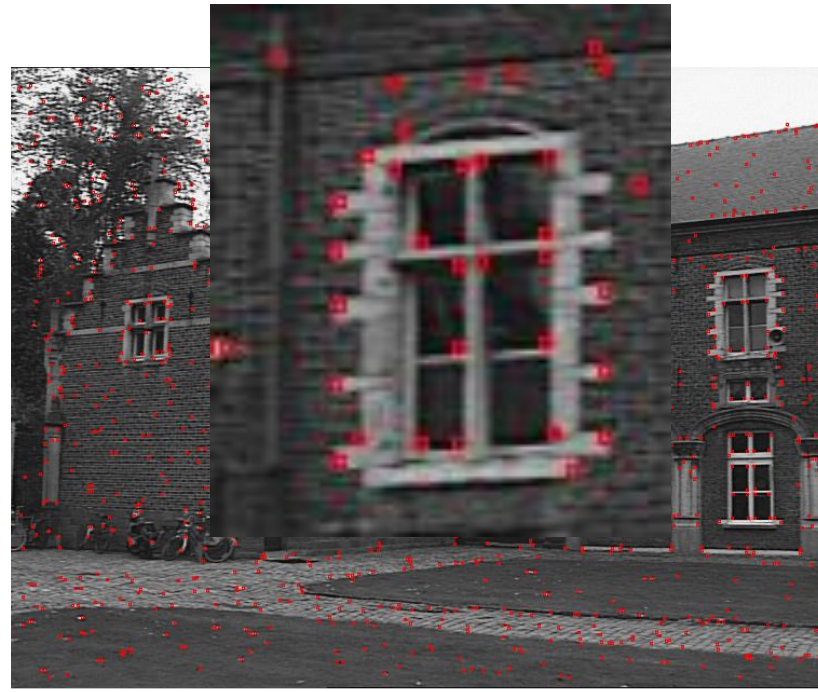
- Alternative measure for “cornerness”

$$R = \det \mathrm{M} - k\,(\mathrm{trace}\mathrm{M})^2$$

- Select **local maxima** as keypoints
- Subpixel accuracy through second order surface fitting (parabola in 1D)

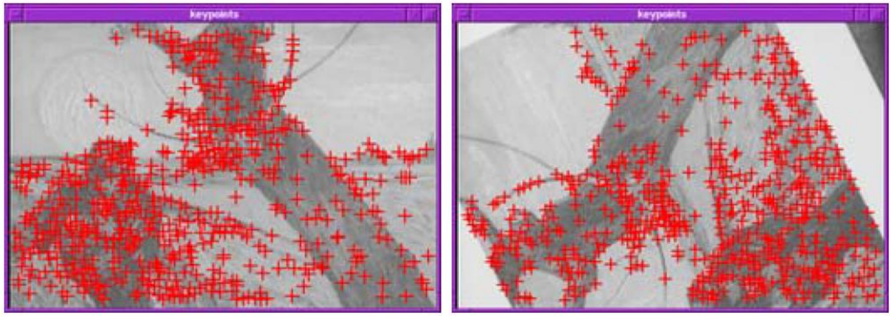
Harris Corner Detector





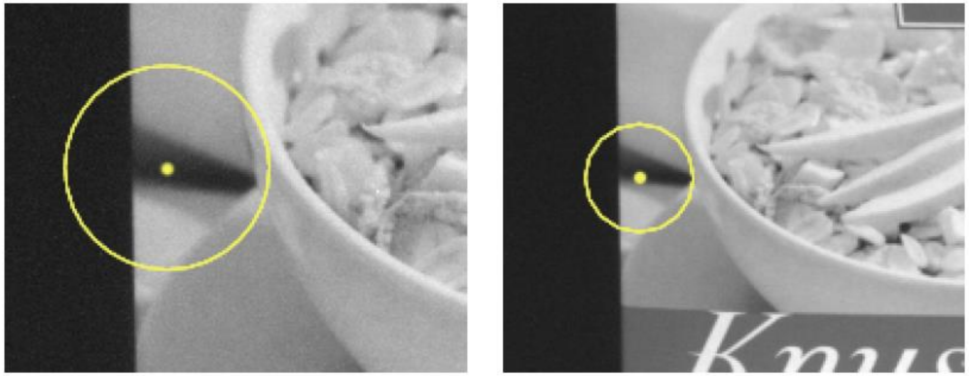
- Keypoint detection: Select strongest features over whole image or over each tile (e.g. 1000 per image or 2 per tile)
- **Invariances** against geometric transformations
  - Shift / translation?

Geometric Invariances



Rotation

Harris: Yes



Scale

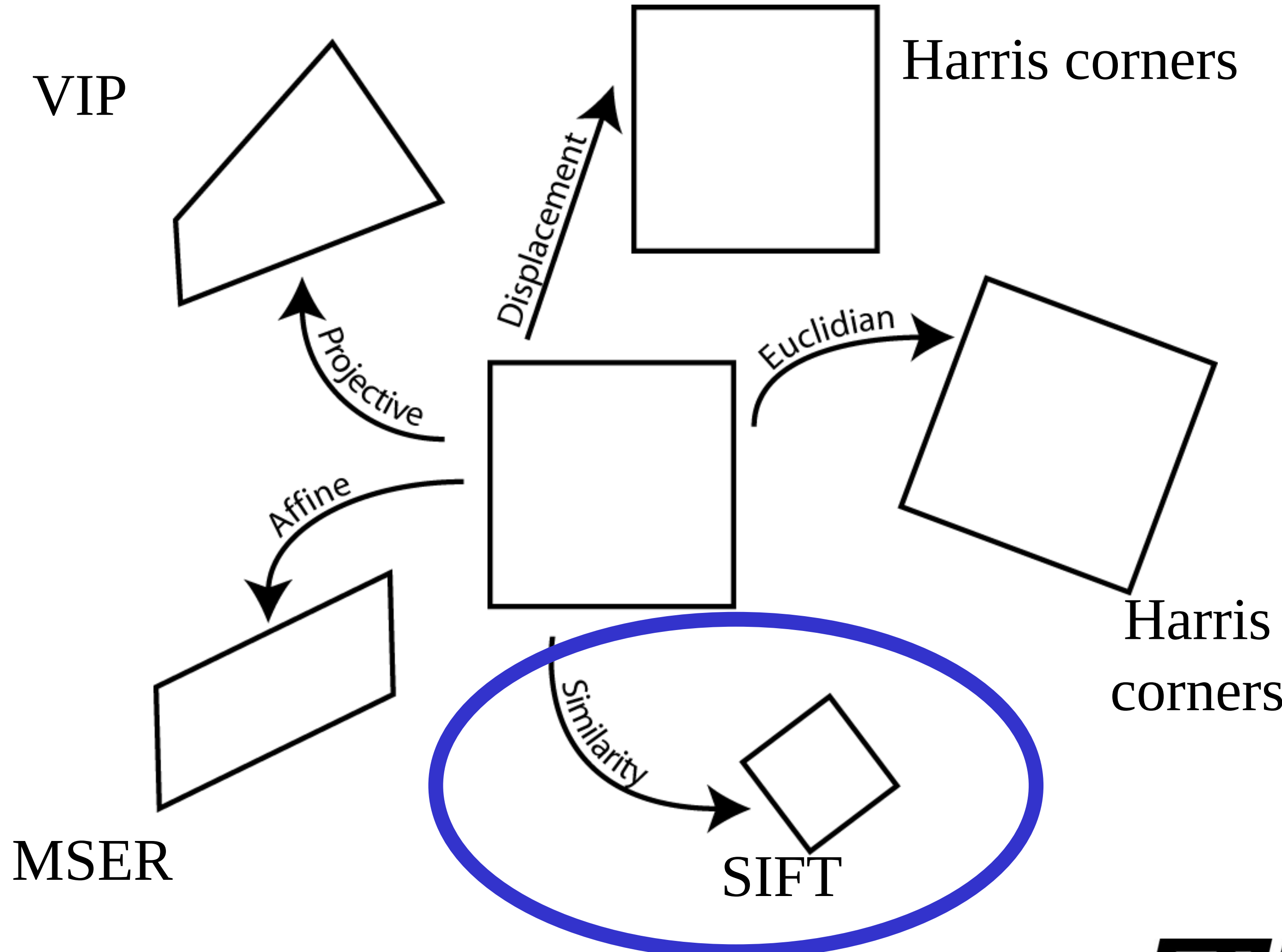
Harris: No

Affine
(approximately invariant
w.r.t. perspective/viewpoint)

Harris: No

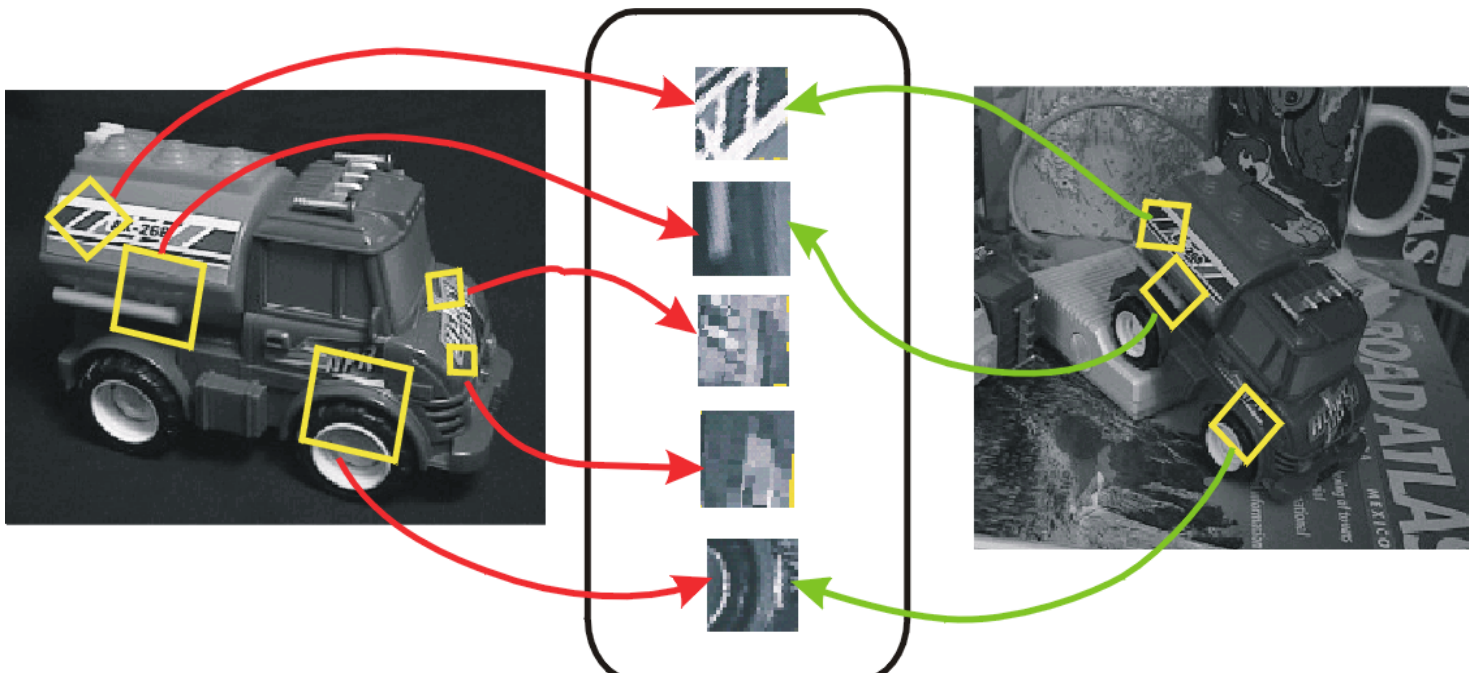
ETH

2D Transformations of a Patch



Scale-Invariant Feature Transform (SIFT)

- Detector + descriptor (later)
- Recover features with position, orientation and scale

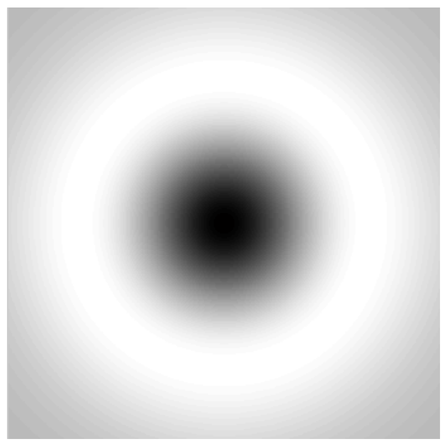


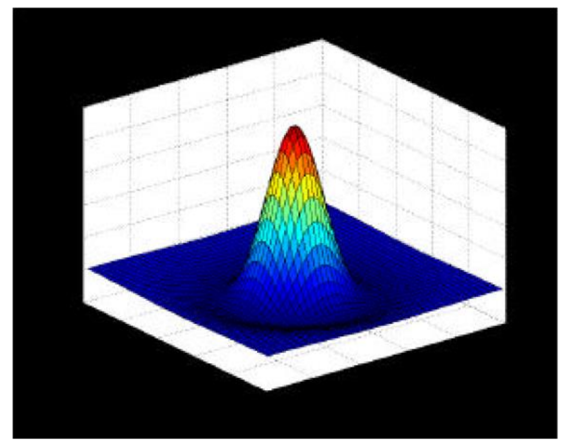
(Lowe, Distinctive Image Features From Scale-Invariant Keypoints, IJCV'04)

Position

- Look for strong responses of **Difference-of-Gaussian** filter (**DoG**)

$$\mathrm{DOG}(x,y) = \frac{1}{2\pi(k\sigma)^2} e^{-\frac{x^2+y^2}{(k\sigma)^2}} - \frac{1}{2\pi(\sigma)^2} e^{-\frac{x^2+y^2}{(\sigma)^2}} \qquad k = \sqrt[3]{2}$$

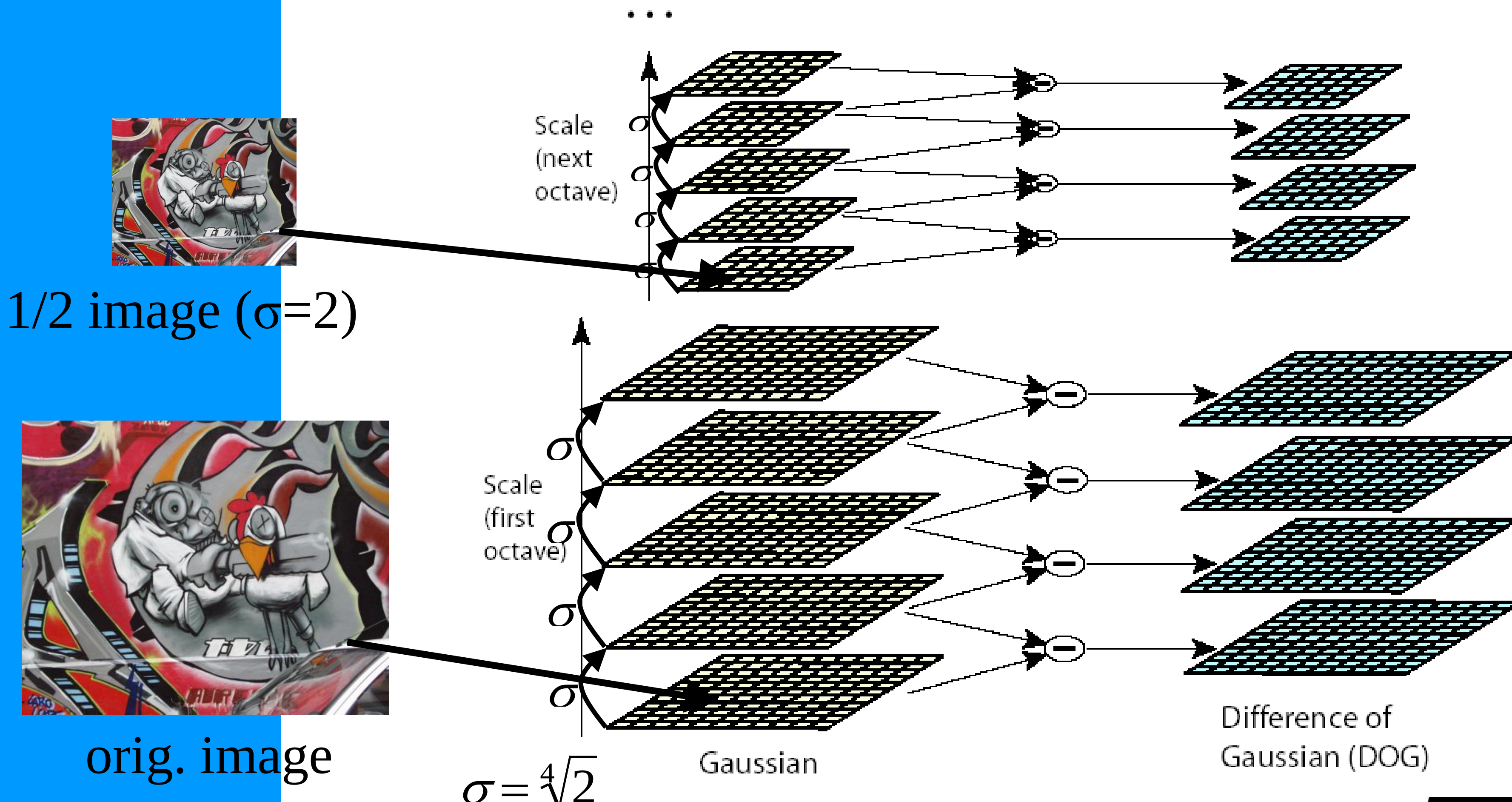




- Approximates **Laplacian of Gaussian** (**LoG**)
- Detects blob-like structures
- Only consider local extrema

Scale

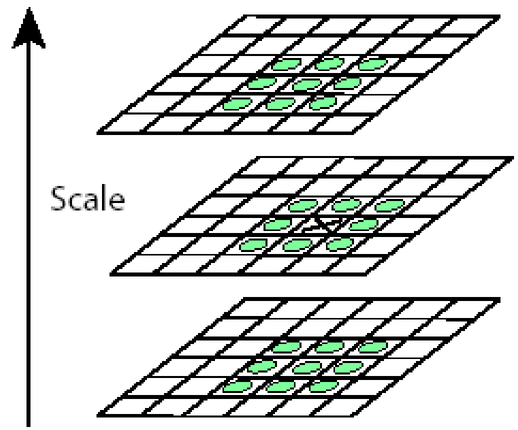
- Look for strong DoG responses over scale space



Slide credits: Bastian Leibe, Krystian Mikolajczyk

Scale

- Only consider **local maxima/minima** in both position and scale



- Fit quadratic around extrema for **sub-pixel** & **sub-scale** accuracy

Minimum Contrast and "Cornerness"

all features

Minimum Contrast and “Cornerness”

after suppressing edge-like features

Minimum Contrast and "Cornerness"

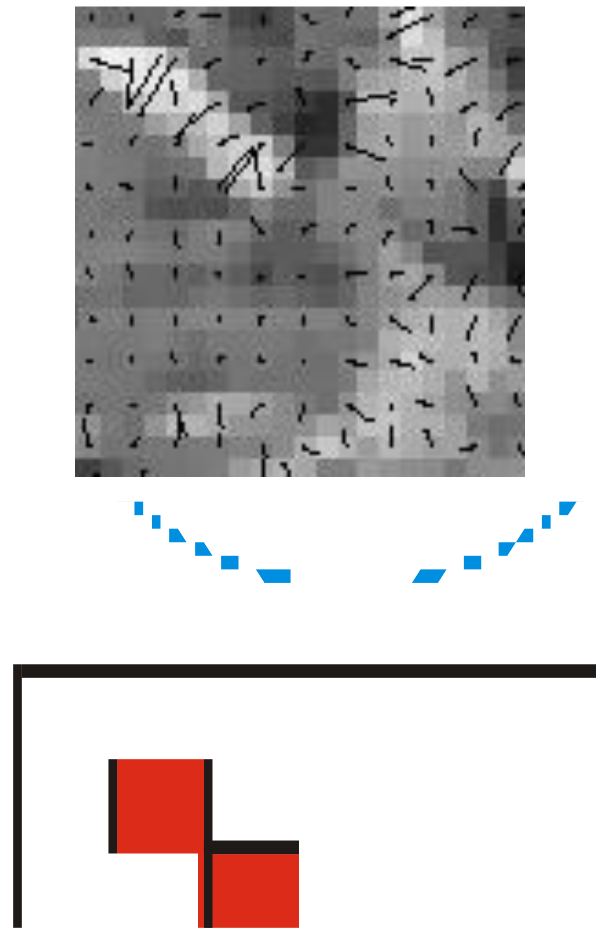
after suppressing edge-like features + small contrast features

Invariants So Far

- Translation? Yes
- Scale? Yes
- Rotation? Yes

Orientation Assignment

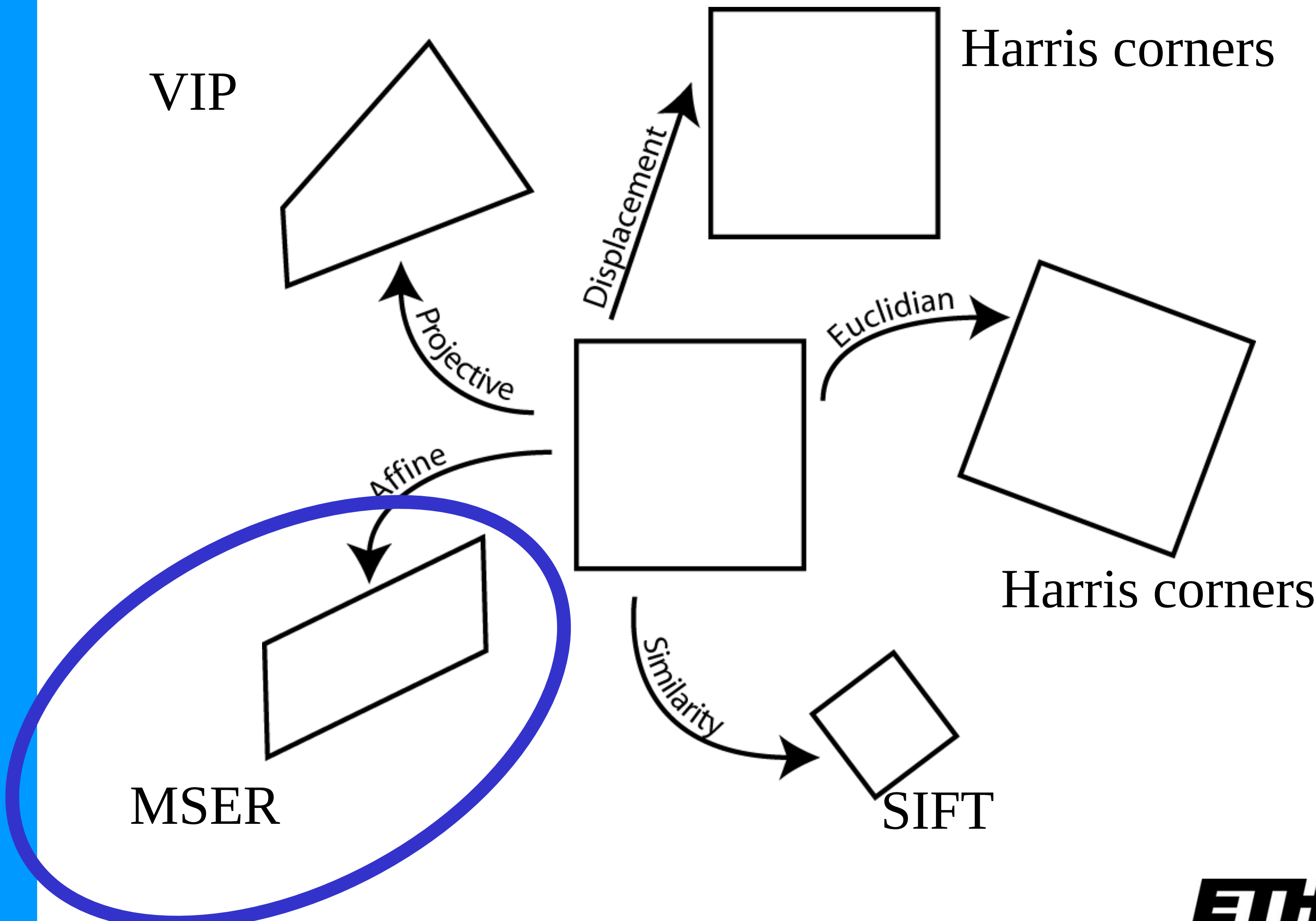
- Compute gradient for each pixel in patch at selected scale
- Bin gradients in histogram & smooth histogram
- Select **canonical orientation** at peak(s)
- Keypoint = 4D coordinate (x, y, scale, orientation)



Invariants So Far

- Translation
- Scale
- Rotation
- Brightness changes:
  - Additive changes?
  - Multiplicative changes?

2D Transformations of a Patch



Affine Invariant Features

Perspective effects can locally be approximated by affine transformation

ETH

Extreme Wide Baseline Matching

- Detect stable keypoints using the **Maximally Stable Extremal Regions** (**MSER**) detector
- Detections are **regions**, not points!

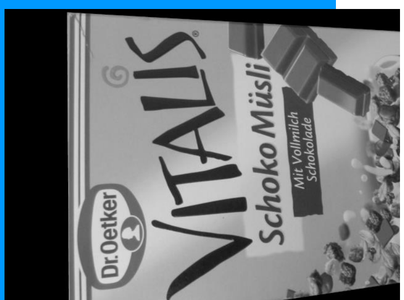
(Matas et al., Robust Wide Baseline Stereo from Maximally Stable Extremal Regions, BMVC'02)

Maximally Stable Extremal Regions

Extremal regions:

- Much brighter than surrounding
- Use intensity threshold

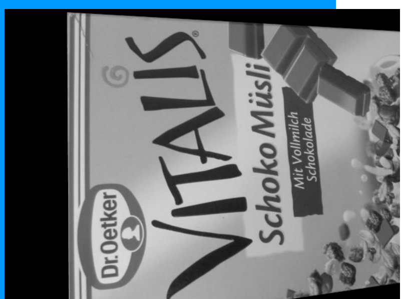
Above Intensity Threshold 000



Maximally Stable Extremal Regions

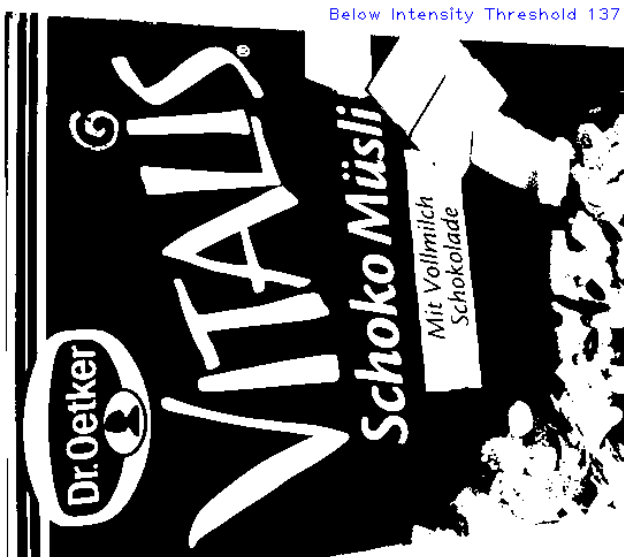
Extremal regions:

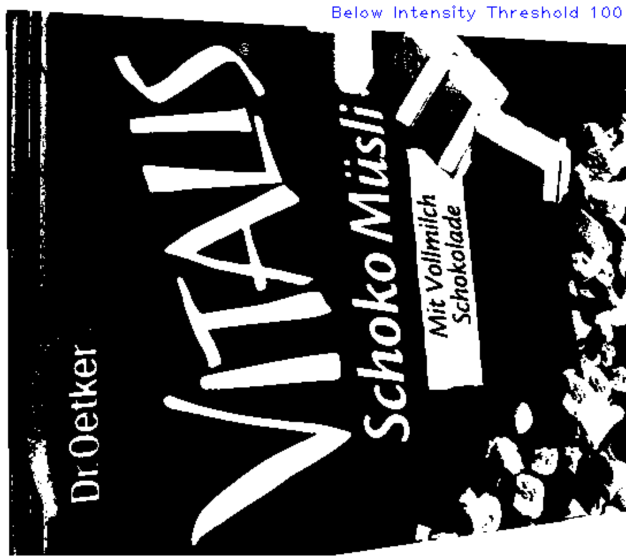
- OR: Much darker than surrounding
- Use intensity threshold



Below Intensity Threshold 254

Maximally Stable Extremal Regions





- Regions: **Connected components** at a threshold
- Region size = #pixels
- **Maximally** stable: Region constant near some threshold

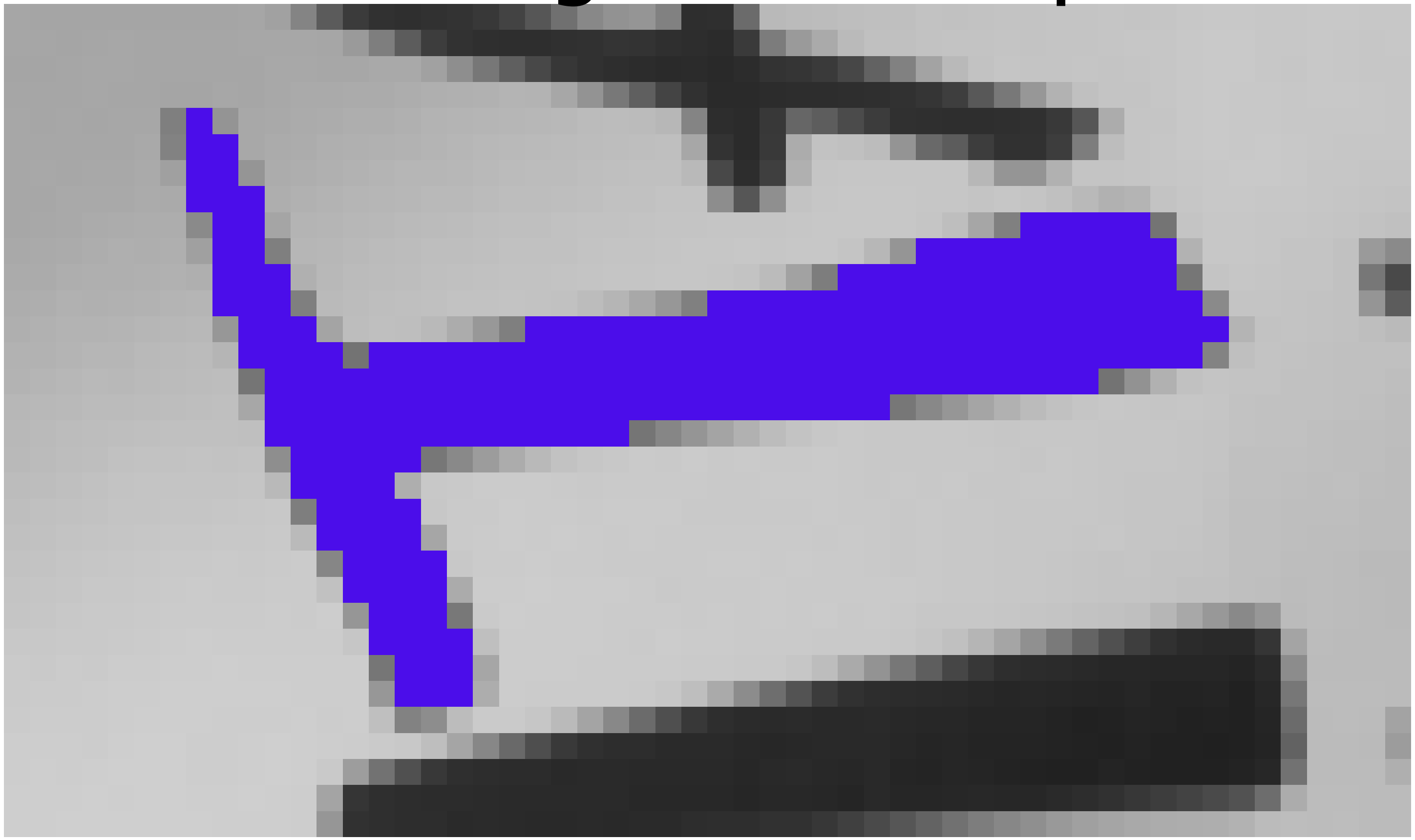
A Sample Feature

ETH

A Sample Feature

T is maximally stable wrt. surrounding

From Regions To Ellipses

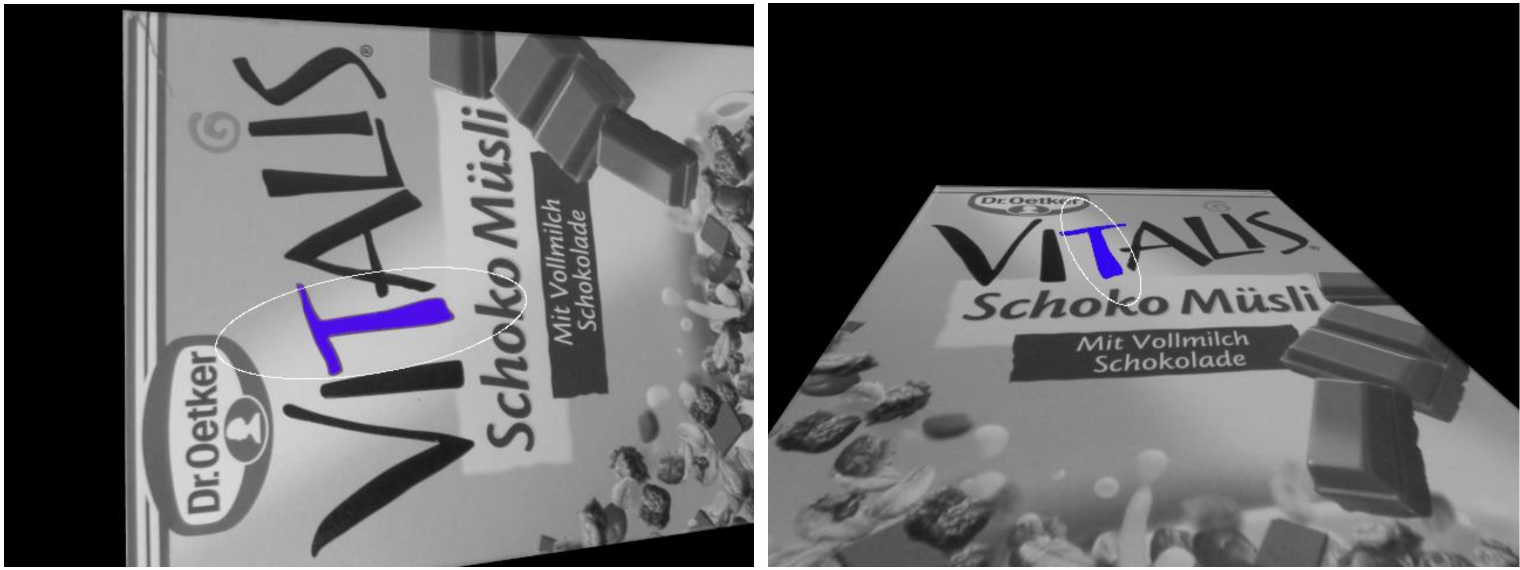


- Compute „center of gravity“
- Compute Scatter (PCA / Ellipsoid)

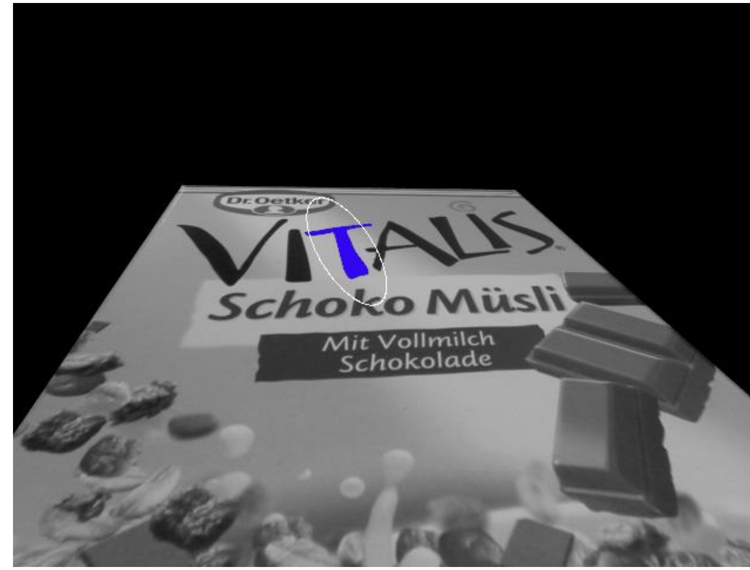
From Regions To Ellipses

- Ellipse abstracts from pixels!
- Geometric representation: position/size/shape

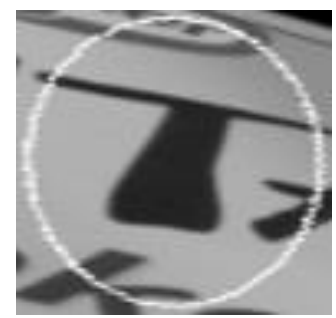
Achieving Invariance



- **Normalize** to „default“ position, size, shape
- For example: Circle of radius 16 pixels



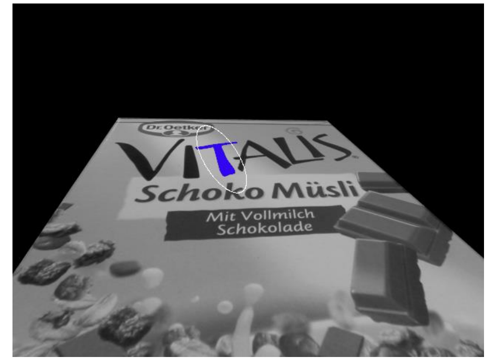
- Normalize ellipse to circle (affine transformation)



- 2D rotation still unresolved

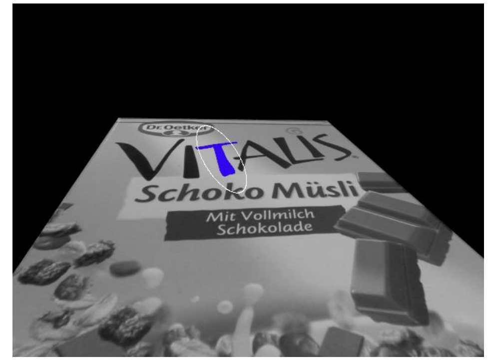
- Same approach as for SIFT:
  Compute histogram of local gradients
- Find dominant orientation in histogram
- Rotate local patch into dominant orientation

Summary: MSER Features



- Detect sets of pixels brighter/darker than surrounding pixels
- Fit elliptical shape to pixel set
- Warp image so that ellipse becomes circle
- Rotate to dominant gradient direction (other constructions possible as well)

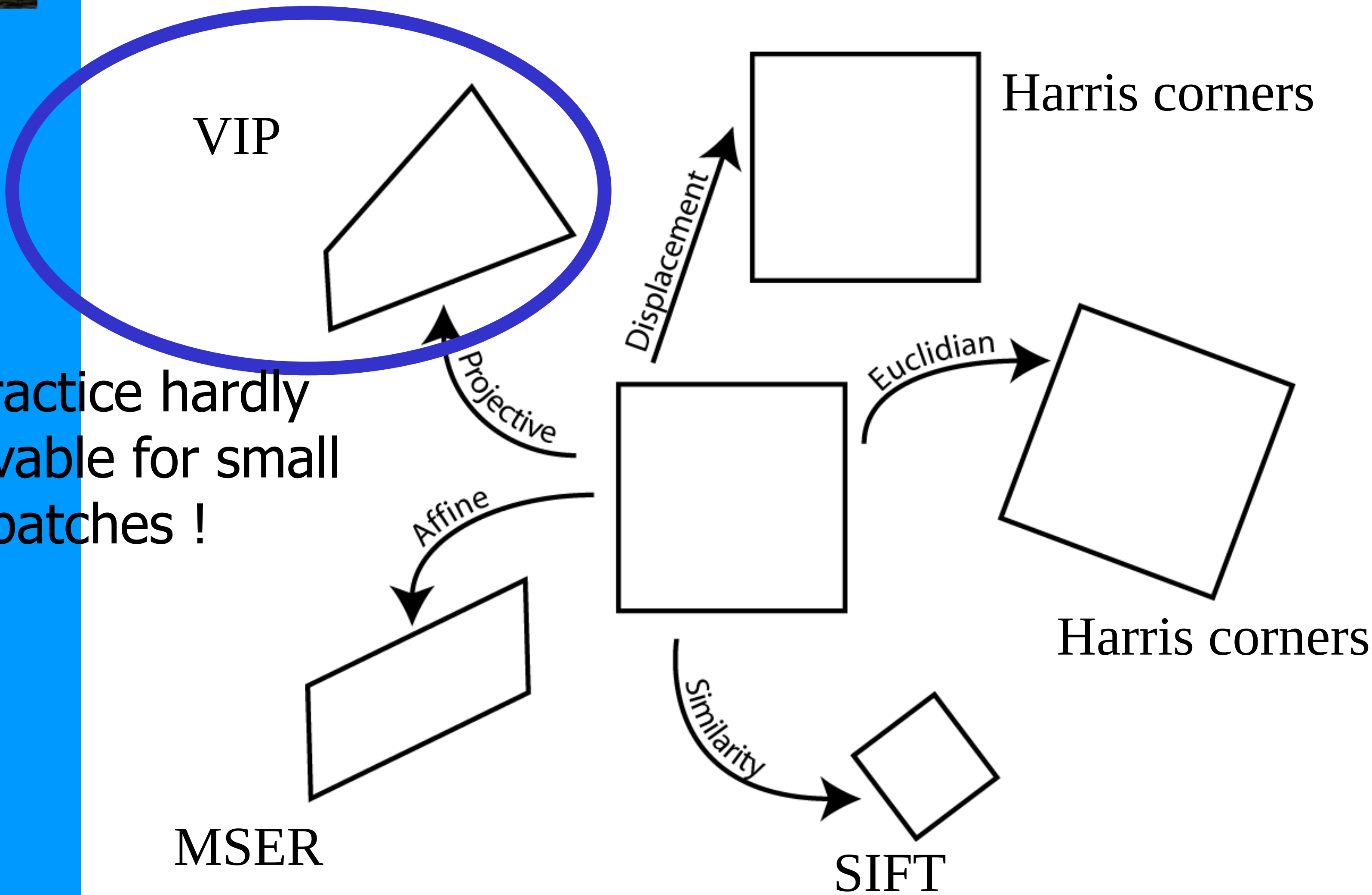
MSER Features - Invariants



- Constant brightness changes (additive and multiplicative)
- Rotation, translation, scale
- Affine transformations

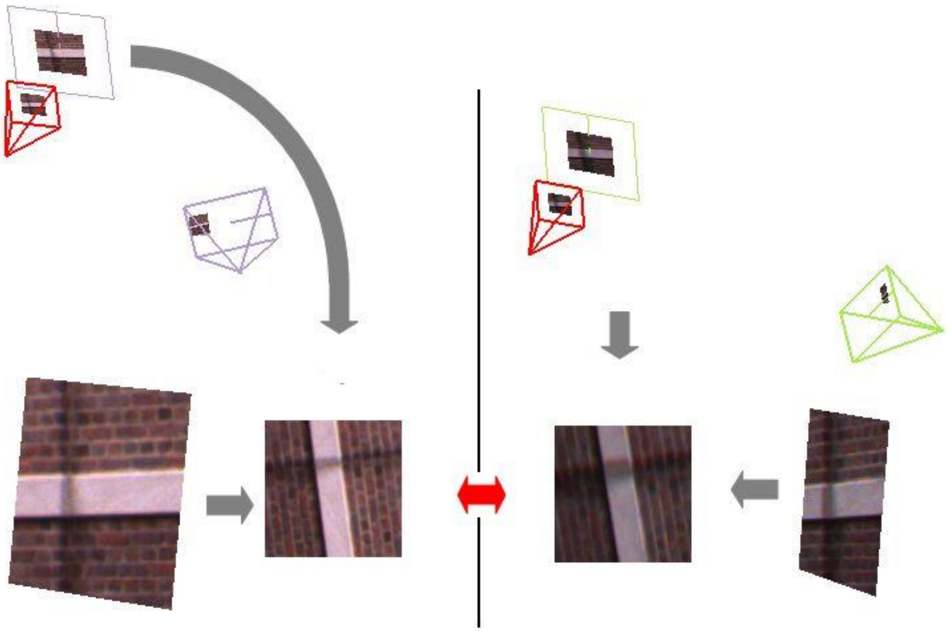
⇒ Affine normalization of feature leads to similar patches in different views !

2D Transformations of a Patch



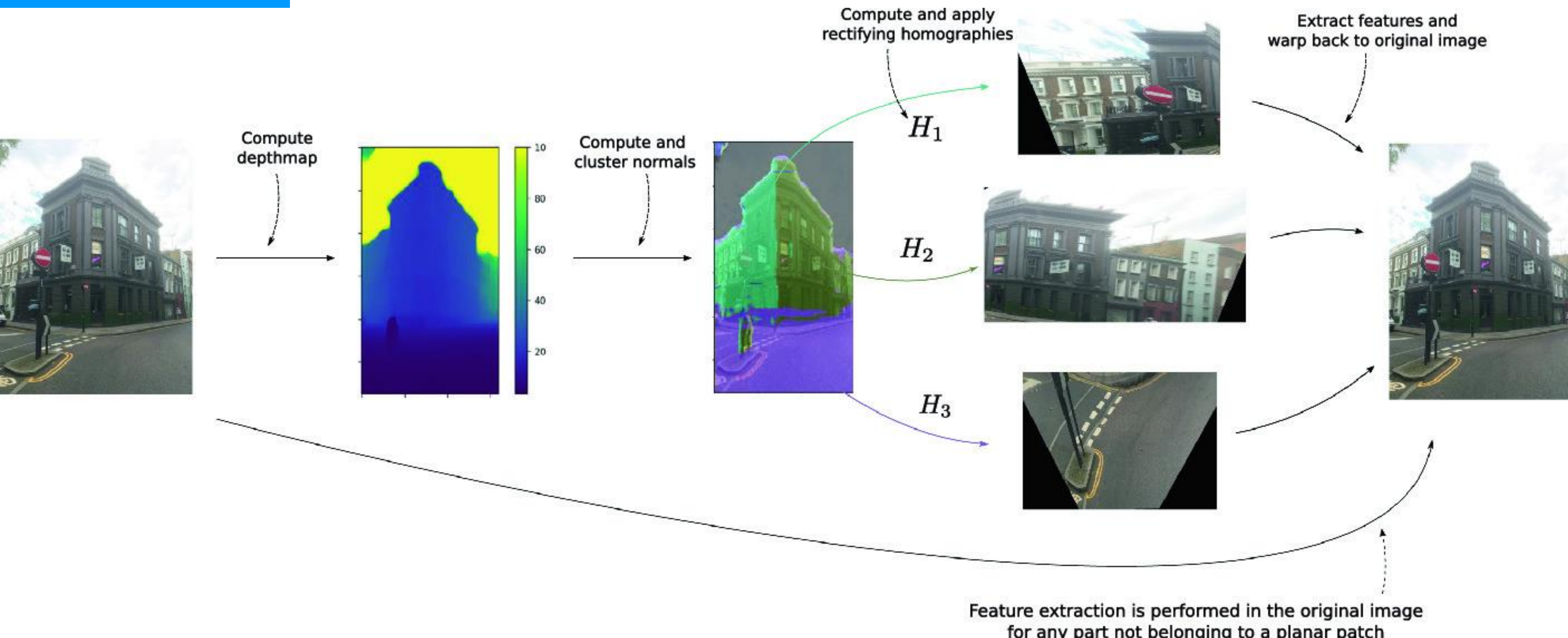
In practice hardly observable for small patches !

Viewpoint Invariant Patches (VIP)



- Use known planar geometry to remove perspective distortion
- Or: Use vanishing points to rectify patch

(Wu et al., 3D Model Matching with Viewpoint Invariant Patches (VIPs), CVPR'08)



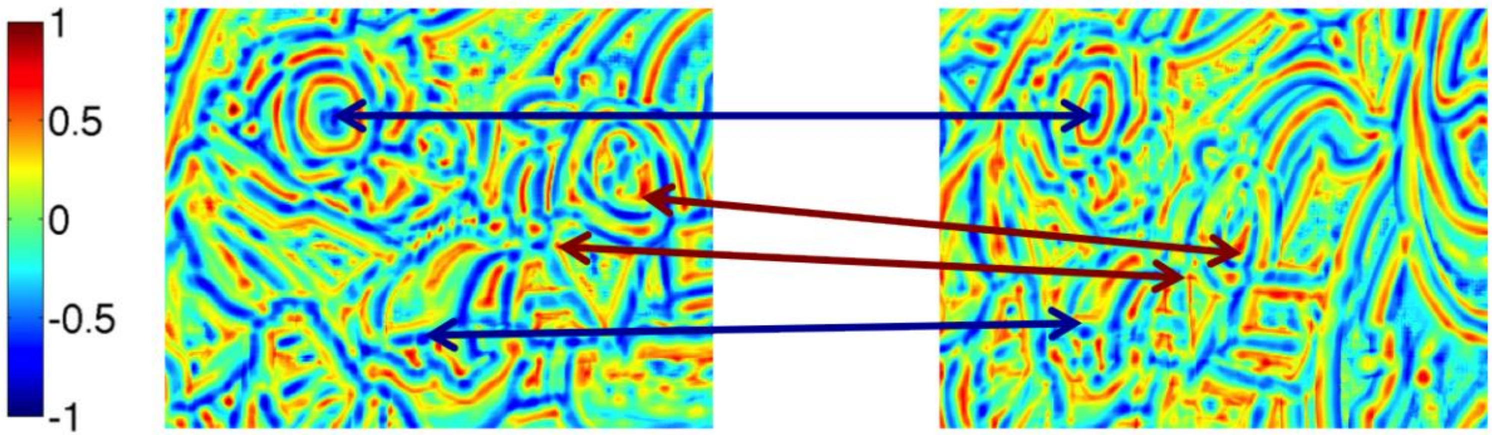
Use learned monocular depth to identify planes

Warp to fronto-parallel before feature detection

(Toft et al., Single-Image Depth Prediction Makes Feature Matching Easier, ECCV'20)

Learning Feature Detectors

- In the age of deep learning, can we learn good detectors from data?
- How can we model repeatable feature detection?

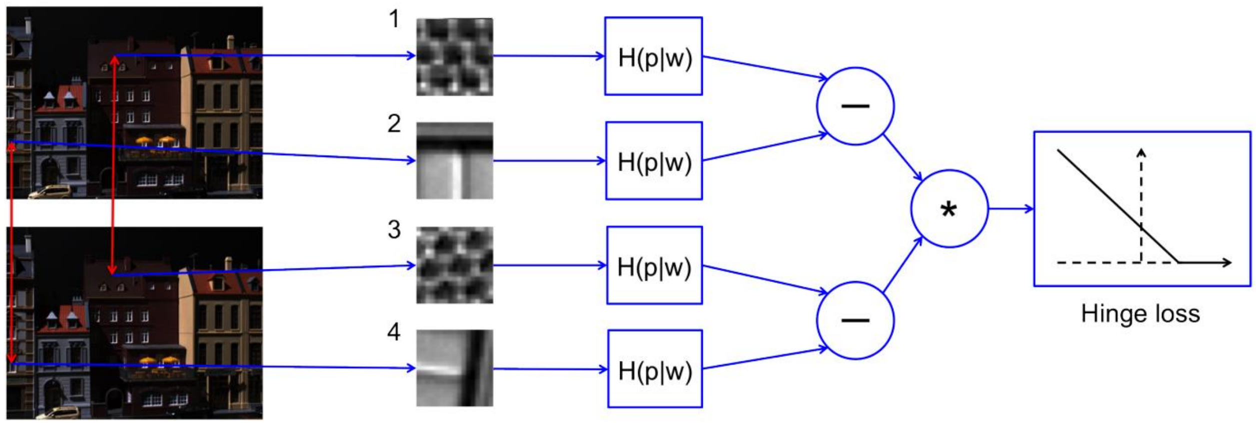


- Learn ranking function $\texttt{H(x|w)}: \mathrm{R}^2 \rightarrow [-1, 1]$ with parameters $\texttt{w}$
- Interesting points close to -1 or 1

(Savinov et al., Quad-networks: unsupervised learning to rank for interest point detection, CVPR'17)

Learning Feature Detectors

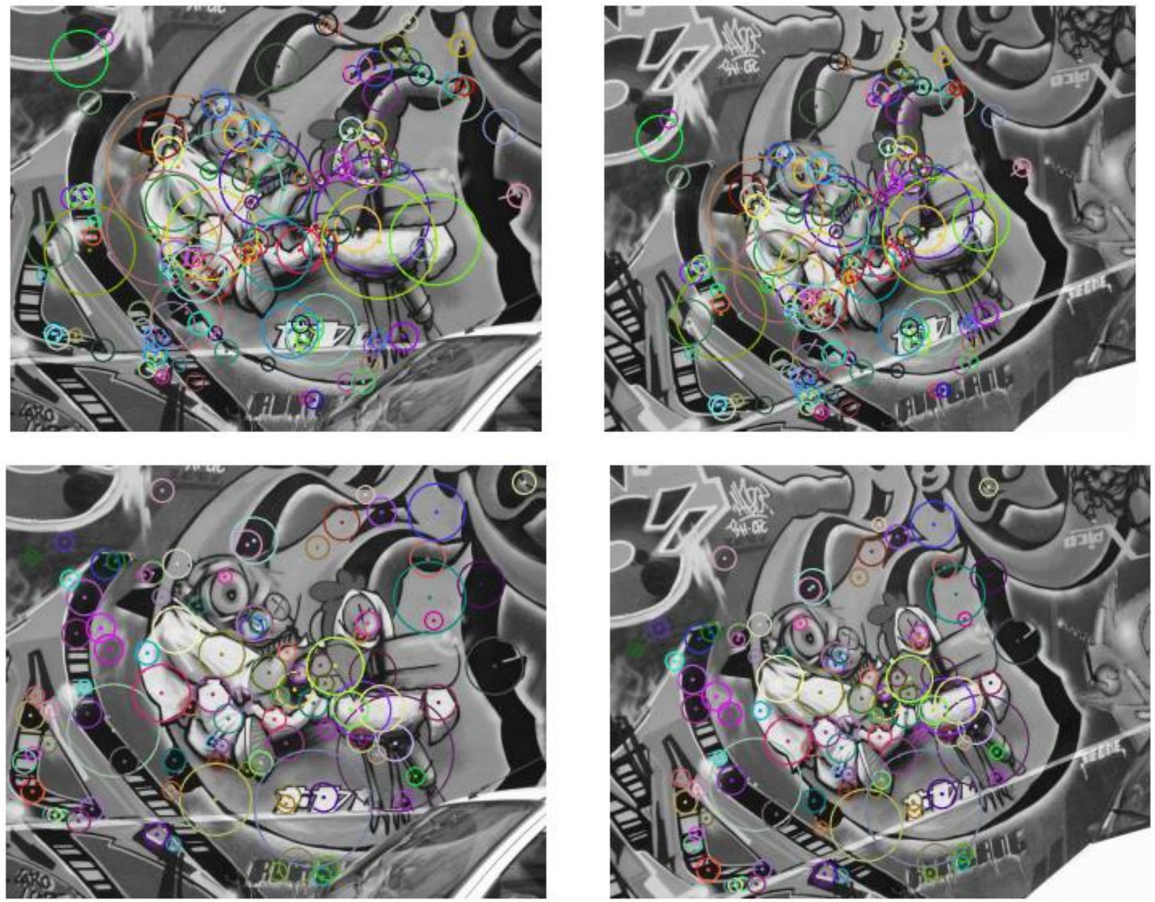
- Learn ranking function *H* for patches **p** such that $H(\mathbf{p}) > H(\mathbf{p}') \Leftrightarrow H(\mathrm{T}(\mathbf{p})) > H(\mathrm{T}(\mathbf{p}'))$



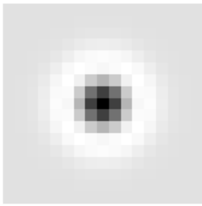
- Select keypoints as top / bottom quantiles
- Learn robustness to different transformations **T**

(Savinov et al., Quad-networks: unsupervised learning to rank for interest point detection, CVPR'17)

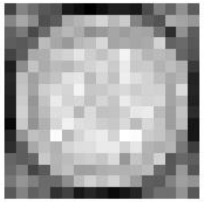
Detection Results



Difference-of-Gaussians



learned



(Savinov et al., Quad-networks: unsupervised learning to rank for interest point detection, CVPR'17)

Summary Feature Detectors

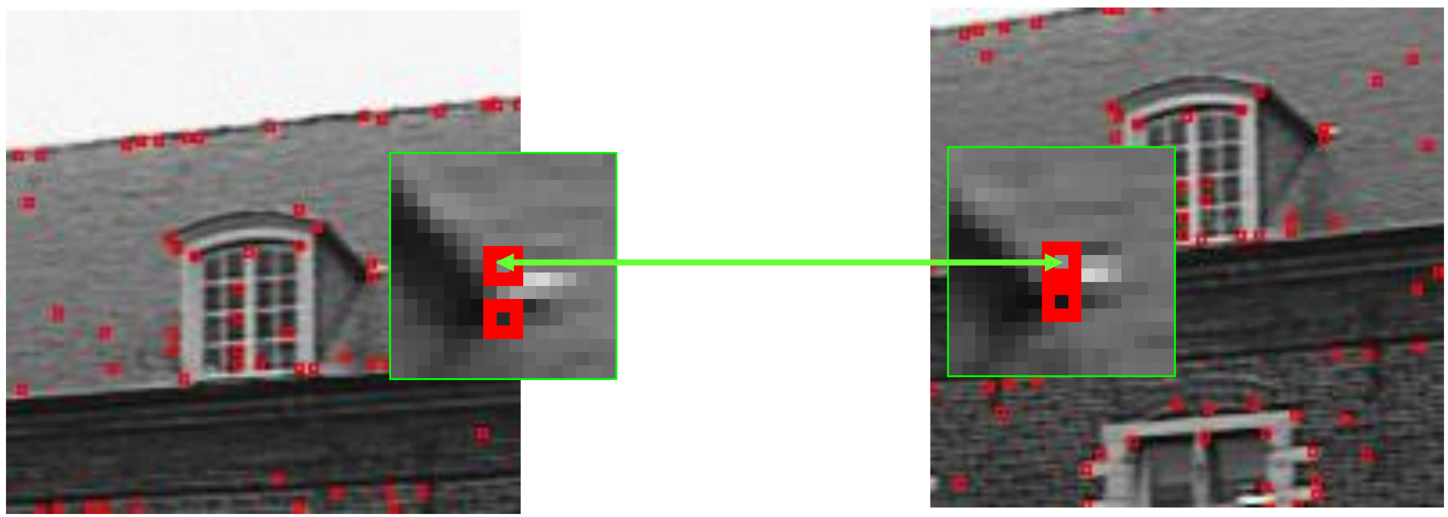
- Motivation: Detect points / regions that are
  - Repeatable
  - Invariant under different conditions
- Key ideas:
  - Detect keypoints as local extrema of suitable response function (e.g., DoG)
  - Scale-invariance by constructing scale space
  - Rotation-invariance from dominant gradient direction
  - Obtain frame of reference through normalization

Overview

- Local Features Intro
- Invariant Feature Detectors
- **Invariant Descriptors & Matching**
- Feature Tracking

Feature Matching

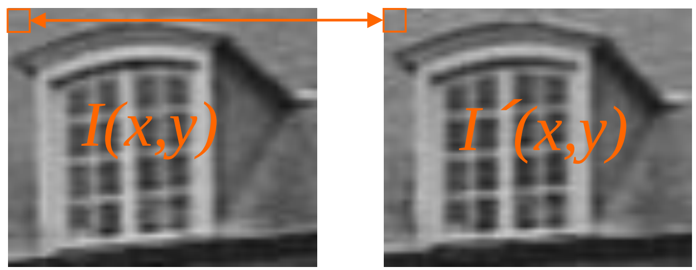
- For each feature in image 1 find the feature in image 2 that is **most similar** and vice-versa



- Keep mutual best matches
- What does most similar mean?
  - Compare **descriptor** per patch, compare descriptors
  - What is a good feature descriptor?

Comparing Image Regions

- Compare intensities pixel-by-pixel



- Dissimilarity measure: **Sum of Squared Differences / Distances** (**SSD**)

$$SSD = \sum_{x} \sum_{y} \left[I'(x,y) - I(x,y) \right]^2$$

- What transformations does this work for?
  - Shifts / translation?
  - Uniform brightness changes?

Comparing Image Regions

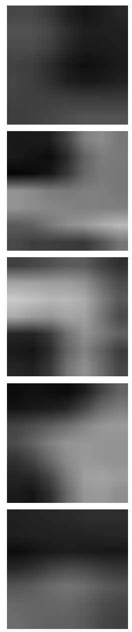
- Compare intensities pixel-by-pixel

- Dissimilarity measure: **Zero-Mean Normalized Cross Correlation (NCC)**

$$NCC = \frac{N(I', I)}{\sqrt{N(I', I')N(I, I)}}$$

$$N(I', I) = \sum_x \sum_y \left(I(x,y) - \bar{I}\right)\left(I'(x,y) - \bar{I}'\right)$$

Feature Matching Example

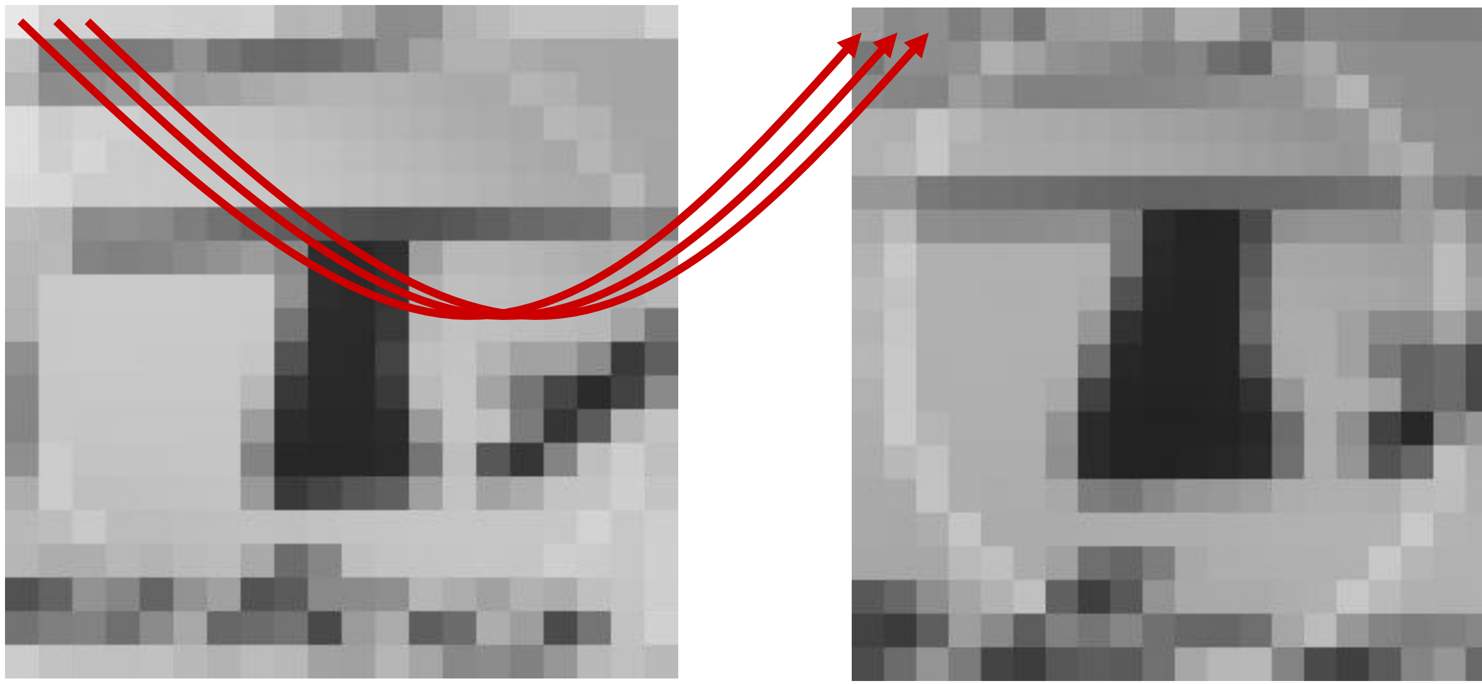


0.96	**-0.40**	**-0.16**	**-0.39**	**0.19**
-0.05	0.75	**-0.47**	**0.51**	**0.72**
-0.18	**-0.39**	0.73	**0.15**	**-0.75**
-0.27	**0.49**	**0.16**	0.79	**0.21**
0.08	**0.50**	**-0.45**	**0.28**	0.99

- What transformations does this work for?
  - Shift / translation, uniform brightness changes
  - Non-uniform brightness changes?

ETH

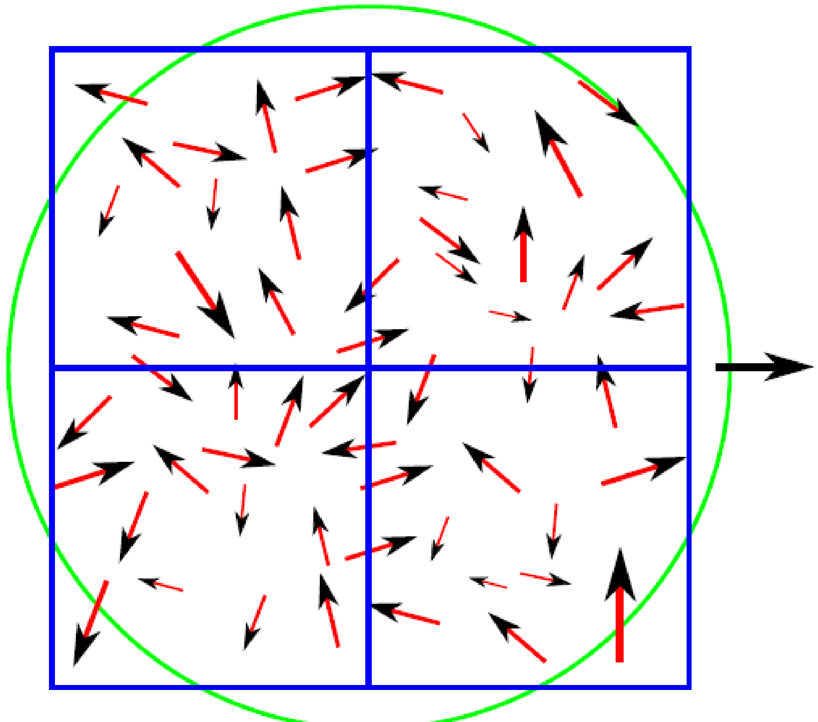
Local Patch Descriptors



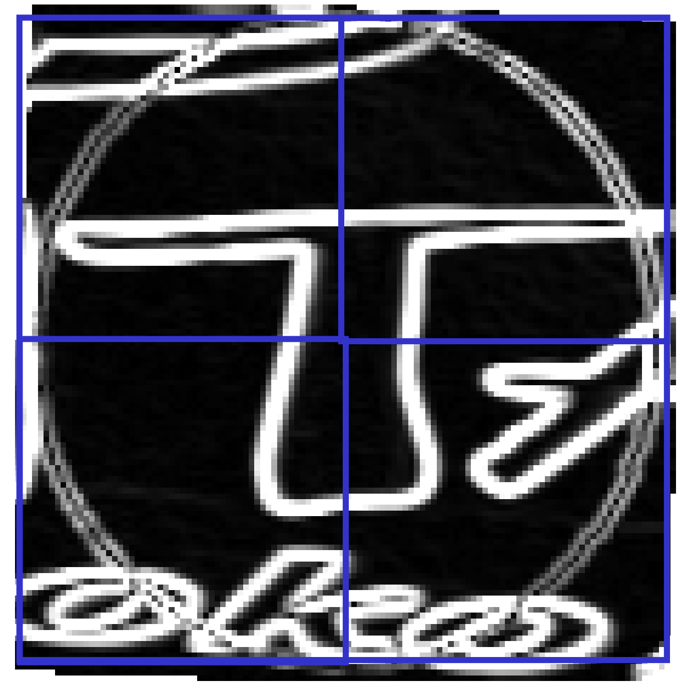
- Small misalignments cannot be avoided
- Non-uniform brightness changes

⇒ More tolerant comparison needed!

Lowe’s SIFT Descriptor



Gradient Orientation/Magnitude



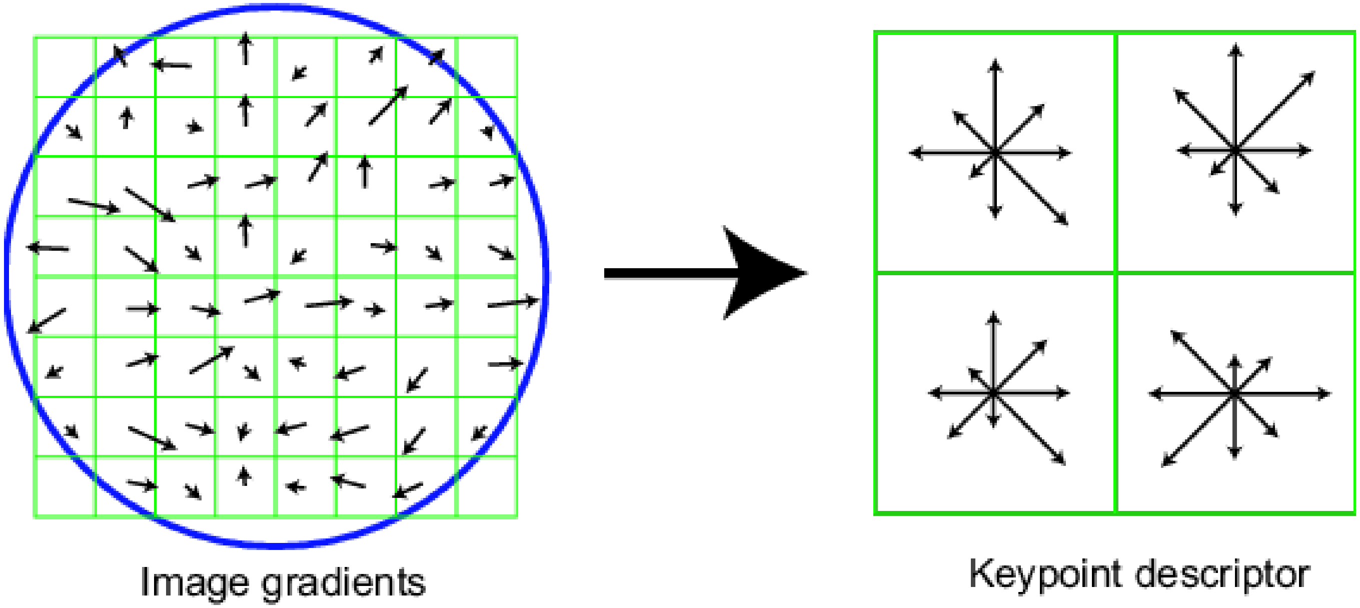
Gradient Magnitude

- Ignore pixel values, use only local gradients
- Gradient direction more important than positions
- Partition into sectors to retain spatial information

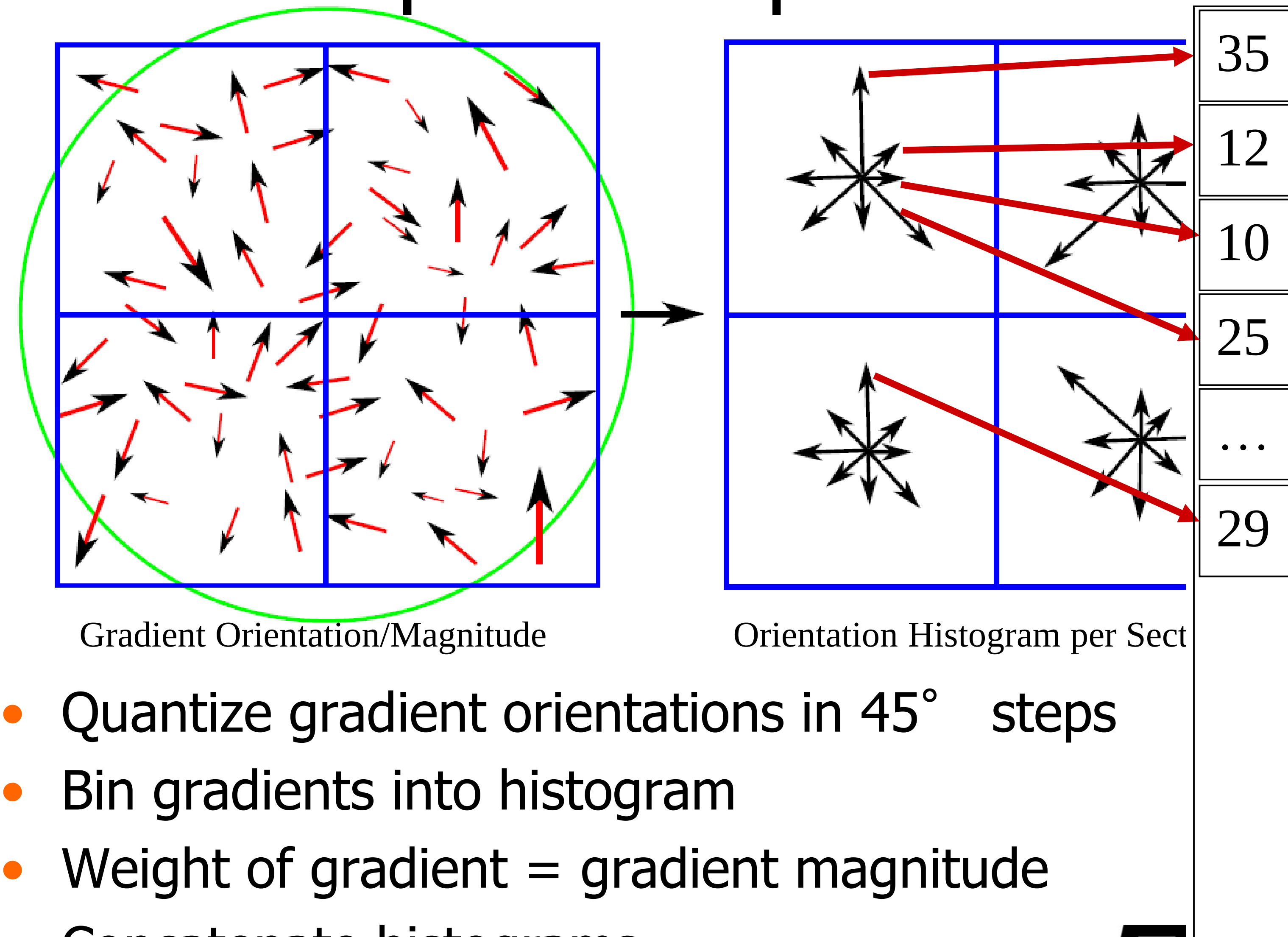
(Lowe, Distinctive Image Features From Scale-Invariant Keypoints, IJCV’04)

Lowe's SIFT Descriptor

- Thresholded image gradients are sampled over 16x16 array of locations in scale space
- Create array of orientation histograms
- 8 orientations x 4x4 histogram array = 128D



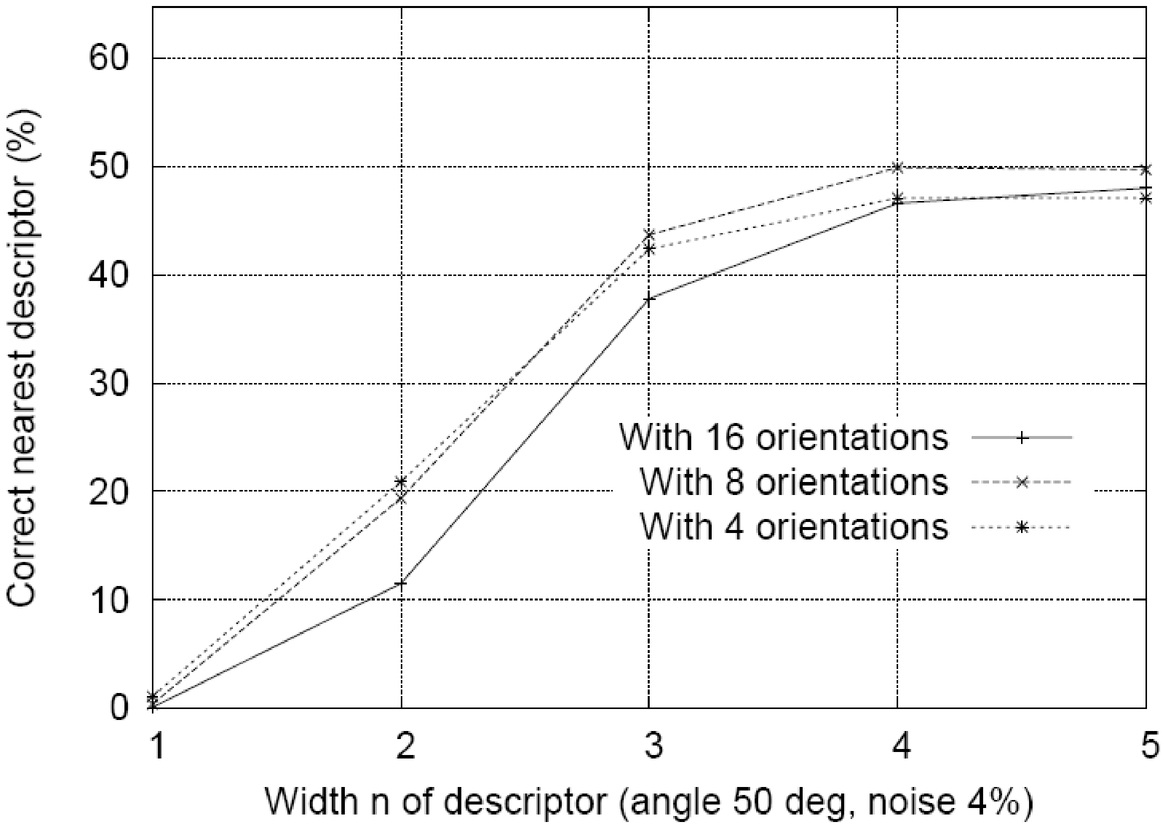
Descriptor Computation



- Quantize gradient orientations in 45° steps
- Bin gradients into histogram
- Weight of gradient = gradient magnitude
- Concatenate histograms

Descriptor Computation

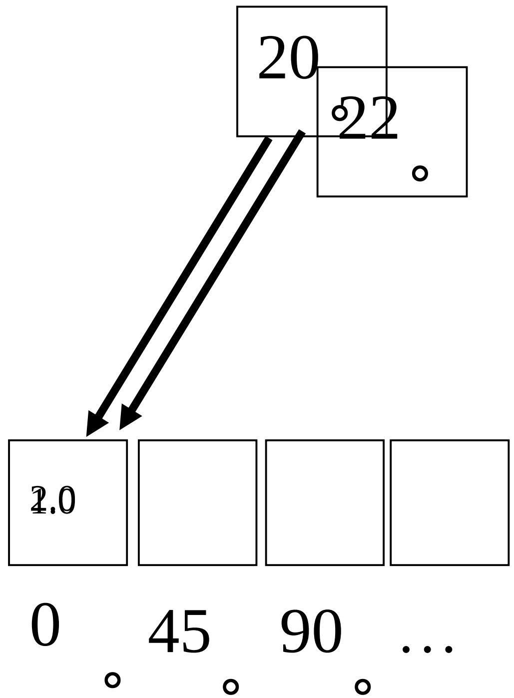
- Why 4x4 regions and 8 histogram bins?
  - Careful parameter tuning!



Descriptor Computation

- Quantization errors: Small differences can lead to different bins !
  - 22° quantized/rounded to 0°
  - 23° quantized/rounded to 45°
- Can be caused by
  - Small errors in feature position, size, shape, or orientation
  - Image noise
- Descriptor **must** be robust against this!

Hard Binning vs. Soft Binning

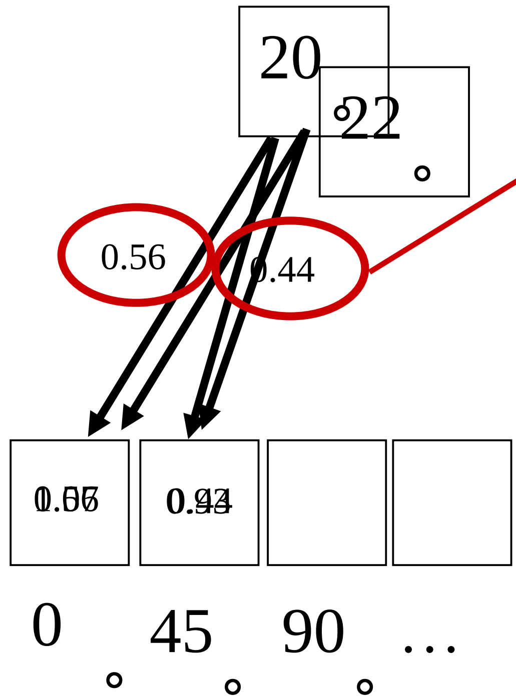


Hard Binning

If orientation is 3° different, all measurements go to second bin!

➔ Sudden change in histogram from (2 0 0 0) to (0 2 0 0)

Hard Binning vs. Soft Binning



Soft Binning

Soft weights:
„Bin Correctness“

If orientation is 3° different, descriptor changes only gradually !

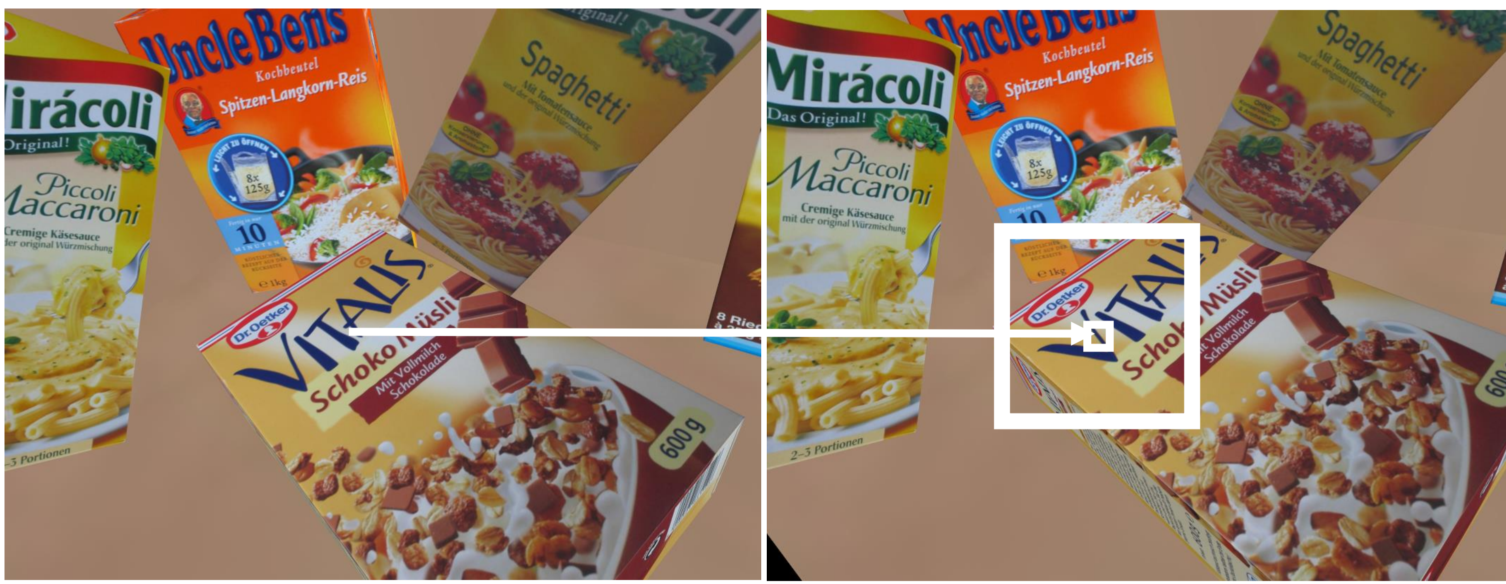
Descriptor Invariance?

- Translation, scale, affine deformations?
  - Inherited from detector
- Rotation?
  - Align bins / histograms with dominant orientation of patch
- Uniform intensity / illumination changes?
  - Adding constant value does not affect gradient
  - Normalize vector to handle multiplicative changes
- Robustness to non-uniform changes?
  - Idea: Change affects gradient magnitude but not direction
  - After normalization: Clip descriptor entries to be ≤ 0.2
  - Renormalize!
  - **But no true invariance!**

Descriptor Matching - Scenario I

Two images in a dense image sequence:

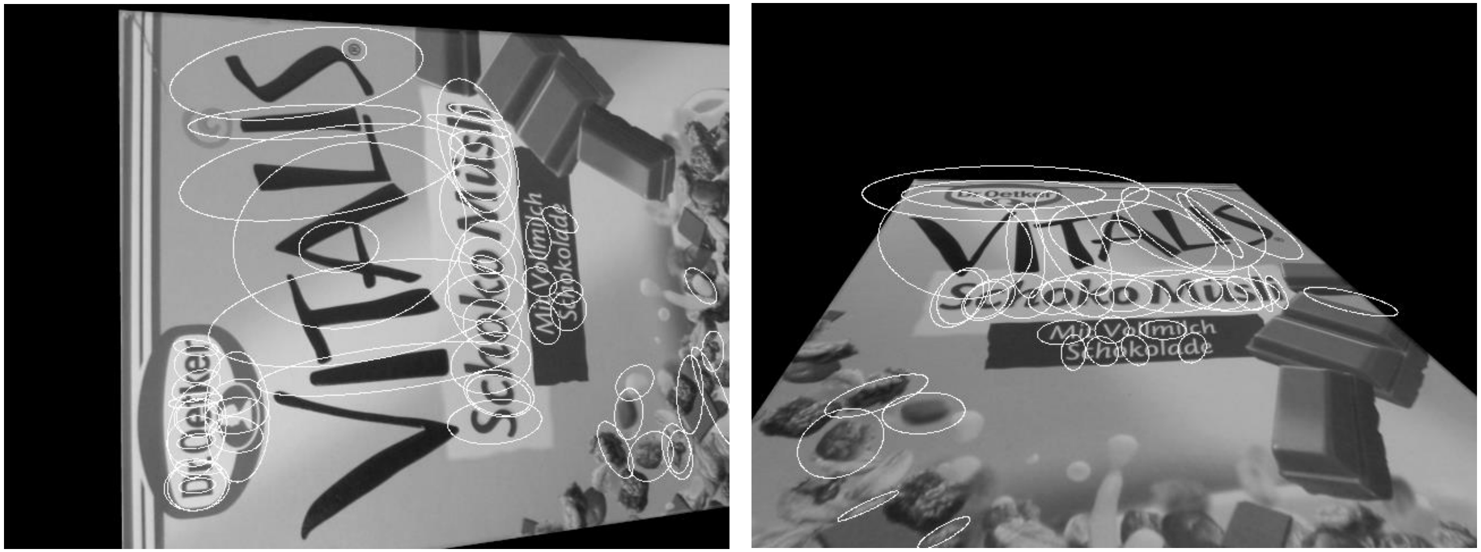
- Think about maximum movement *d* (e.g. 50 pixel)
  - Search in a window +/- *d* of old position
- Compare descriptors (Euclidean distance), choose nearest neighbor



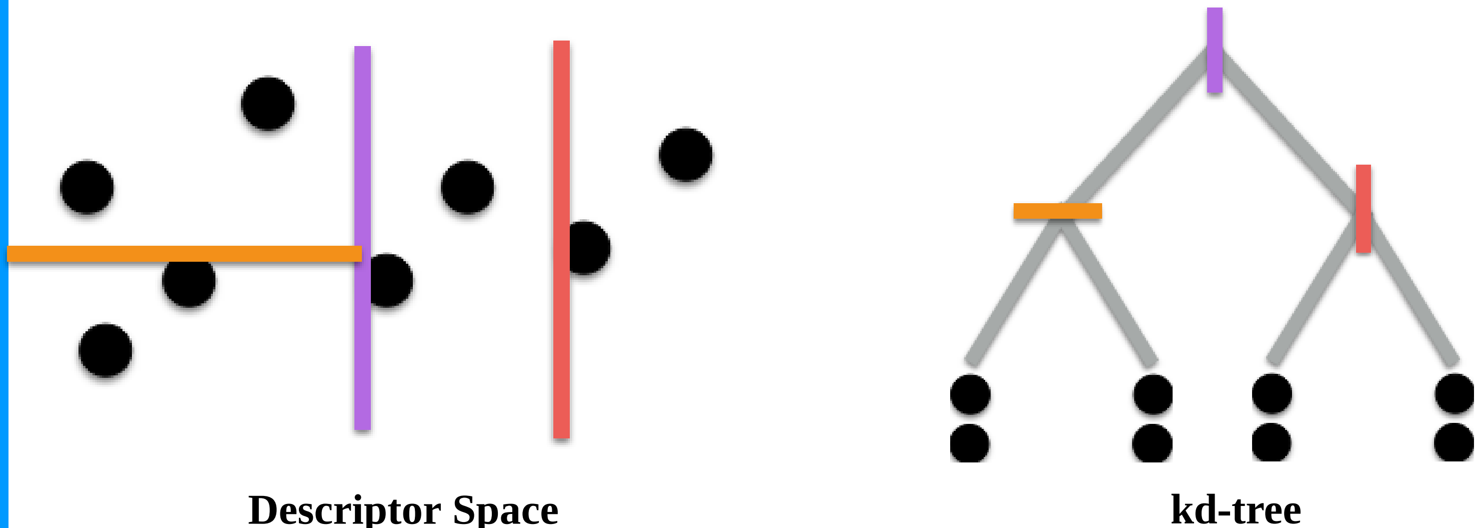
Descriptor Matching - Scenario II

Two arbitrary images / wide baseline

- Brute force search (e.g. GPU)
- **OR: Approximate nearest neighbor search in descriptor space (kd-tree)**
- OR: Find small set of matches, predict others



kd-tree-based Matching



- Iteratively split dimension with largest variance
- Matching: Traverse tree based on splits
- Depth 30 ≈ 1B descriptors (~119GB for SIFT)
  - Curse of Dimensionality: Need to visit all leaves to guarantee finding nearest neighbor
  - **Approximate search**: Visit *N* leaf nodes

Descriptor Matching

Spatial Search Window:

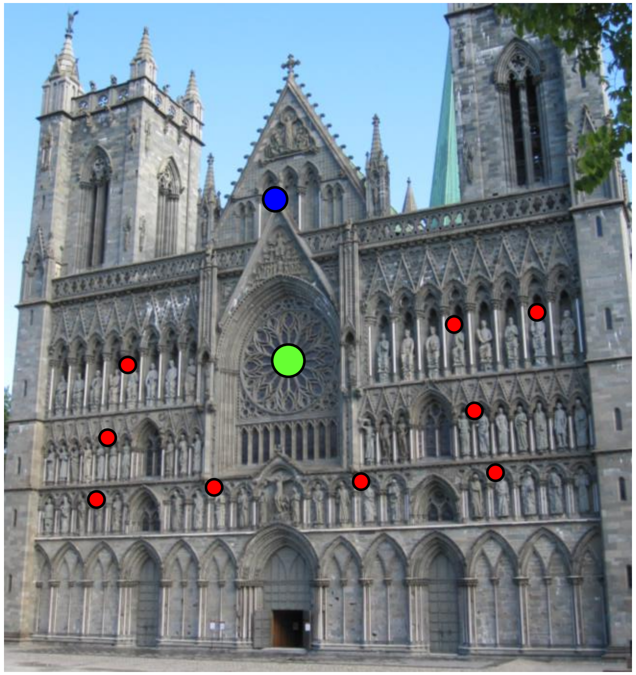
- Requires/exploits good prediction
- Can avoid far away similar-looking features
- Good for sequences

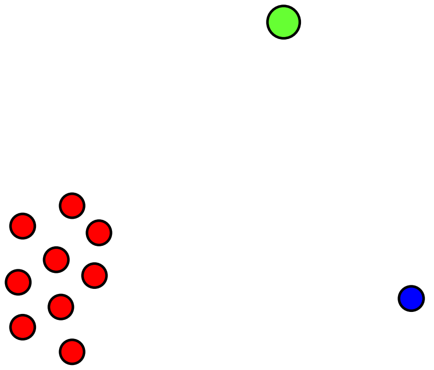
Descriptor Space:

- Initial tree setup
- Fast lookup for huge amounts of features
- More sophisticated outlier detection required
- Good for asymmetric (offline/online) problems, registration, initialization, object recognition, wide baseline matching

Correspondence Verification

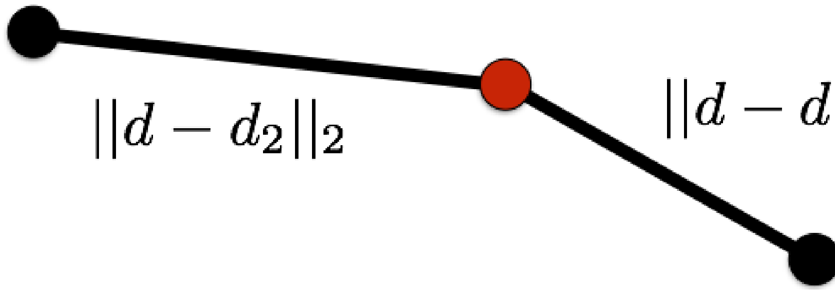
- Not every feature repeats / has nearest neighbor
- How to detect such wrong matches?





- Thresholding on Euclidean distance not meaningful

Correspondence Verification



$$\frac{||d - d_1||_2}{||d - d_2||_2} < 0.8$$

- Discard „non-distinctive“ matches through Lowe‘s ratio test / SIFT ratio test
- Check for bi-directional consistency
- Such heuristics will not eliminate all wrong matches

Binary Descriptors

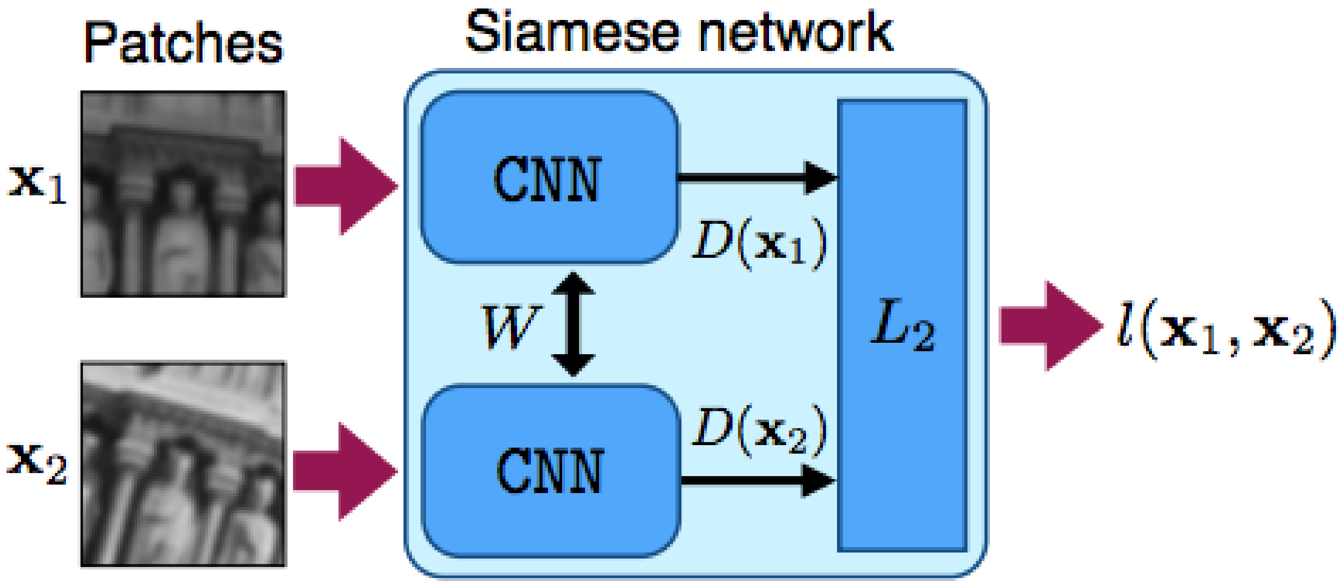
- SIFT is powerful descriptor, but slow to compute
- Faster alternative: **Binary Descriptors**:
  - Idea: Compute only **sign of gradient**
  - Efficient test: **Compare** pixel intensities
  - **Random** comparisons work already very well
- Pros:
  - Efficient computation
  - Efficient descriptor comparison via **Hamming distance** (1M comparison in ~2ms for 64D)
- Cons:
  - Not as good as SIFT / real-valued descriptors
  - Many bits rather random = problems for efficient nearest neighbor search

Binary Descriptors

- **BRIEF**[Calonder10]: binary descriptor (tests=position a darker than b), compare descriptors by XOR (Hamming) + POPCNT
- **RIFF**[Takacs10]: CENSURE + gradients tangential/radial
- **ORB**[Rublee11] FAST+orientation
- **BRISK**[Leutenegger11] FAST+scale+BRIEF
- **FREAK**[Alahi12] FAST + “daisy”-BRIEF
- **Lucid**[Ziegler12]: “sort intensities”
- **D-BRIEF**[Trzcinski12]:Box-Filter+learned projection+BRIEF
- **LDA HASH**[Strecha12]: binary tests on descriptor

Learning Feature Descriptors

- In the age of deep learning, can we learn good descriptors from data?
- Idea: Learn a mapping such that descriptors of same physical point have small L_2 distance

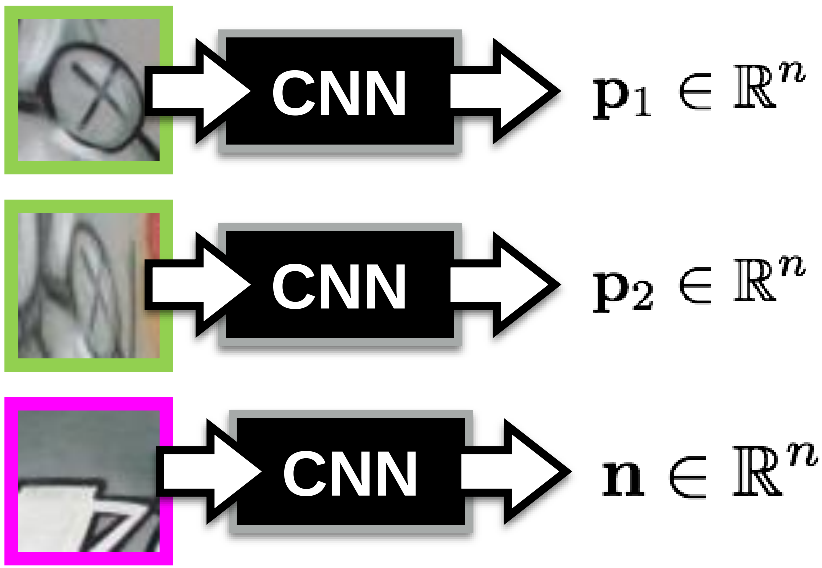


(Simo-Serra et al., Discriminative Learning of Deep Convolutional Feature Point Descriptors, ICCV'15)

ETH

Learning Feature Descriptors

- Learn mapping from patch to descriptor in $\mathbb{R}^n$
- Popular approach: Learning via *triplets*



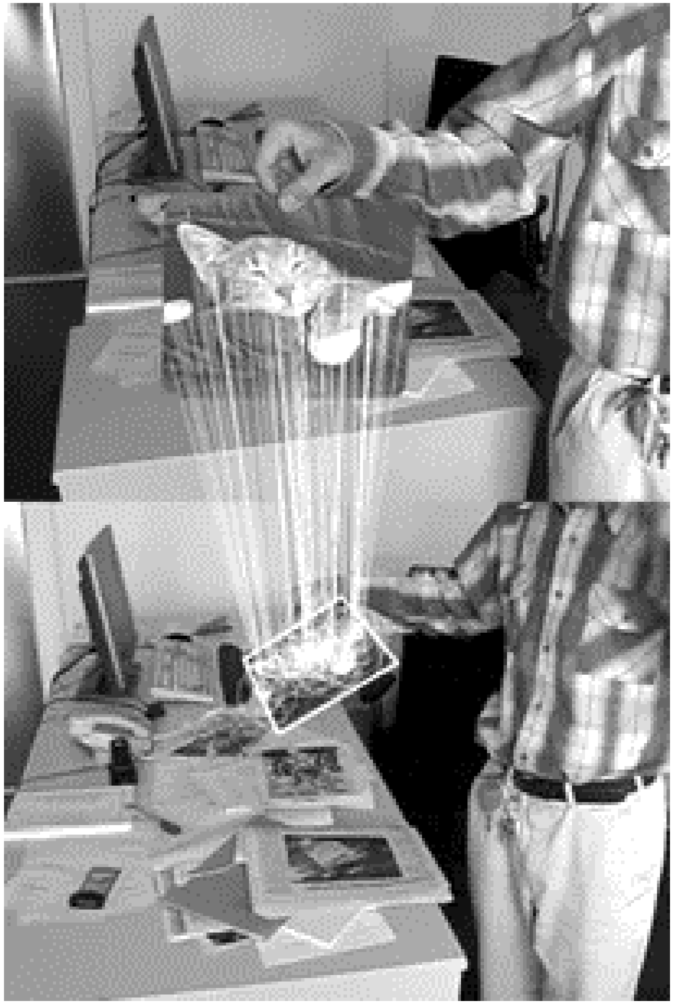
triplet loss:

$$\max(0, \eta + ||\mathbf{p}_1 - \mathbf{p}_2||_2 - ||\mathbf{n} - \mathbf{p}_2||_2)$$

(Schönberger et al., Evaluation of Hand-Crafted and Learned Local Features. CVPR 2017)

ETH

Learning Feature Descriptors



- But idea is actually much older (>10 years)

(Özuysal et al., Fast Keypoint Recognition in Ten Lines of Code, CVPR’**07**)

Some Feature Resources

- Affine feature evaluation + binaries:
  http://www.robots.ox.ac.uk/~vgg/research/affine/
- SIFT, MSER & much more (mostly Matlab):
  http://vlfeat.org
- SURF:
  http://www.vision.ee.ethz.ch/~surf/
- GPU-SIFT:
  http://www.cs.unc.edu/~ccwu/siftgpu/
- DAISY (dense descriptors)
  http://cvlab.epfl.ch/~tola/daisy.html
- FAST[er] corner detector (simple but …)
  http://svr-www.eng.cam.ac.uk/~er258/work/fast.html
- OpenCV (MSER, binary descriptors, matching, …)
  http://opencv.org

Summary Feature Descriptors

- Representation of normalized patches
  - Inherit geometric invariances from detector
- Feature matching by comparing descriptors
- Key ideas:
  - Robustness against small changes in illumination
  - Robustness against small shifts
  - Pool information (e.g., gradients in SIFT)
- More invariance = less powerful descriptors
  - What invariances do you need?
    - E.g.: Rotation-invariance not needed?
    - If not, disable rotation estimation in SIFT

Overview

- Local Features Intro
- Invariant Feature Extraction & Matching
- **Feature Tracking**

Feature Tracking

- Identify features and track them over video
  - Small difference between frames
  - Potential large difference overall
- Standard approach:
  KLT (Kanade-Lukas-Tomasi)

Tracking Corners in Video

Good Features to Track

- Use same window in feature selection as for tracking itself (see first part of lecture)

$$\text{with } \mathrm{M} = \iint_W \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} w(x,y)\mathrm{d}x\mathrm{d}y$$

- Compute motion assuming it is <u>small</u>

$$I(\boldsymbol{x} + \boldsymbol{\Delta}) \approx J(\boldsymbol{x})$$

$$min \iint_W \left(I + \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \Delta - J\right)^2 w(x,y)\mathrm{d}x\mathrm{d}y$$

differentiate wrt. Δ: $\iint_W 2 \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \left(I + \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \Delta - J\right) w(x,y)\mathrm{d}x\mathrm{d}y$

linear system in Δ

$$\boxed{\iint_W \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} w(x,y)\mathrm{d}x\mathrm{d}y\Delta = \iint_W \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} (J - I) w(x,y)\mathrm{d}x\mathrm{d}y}$$

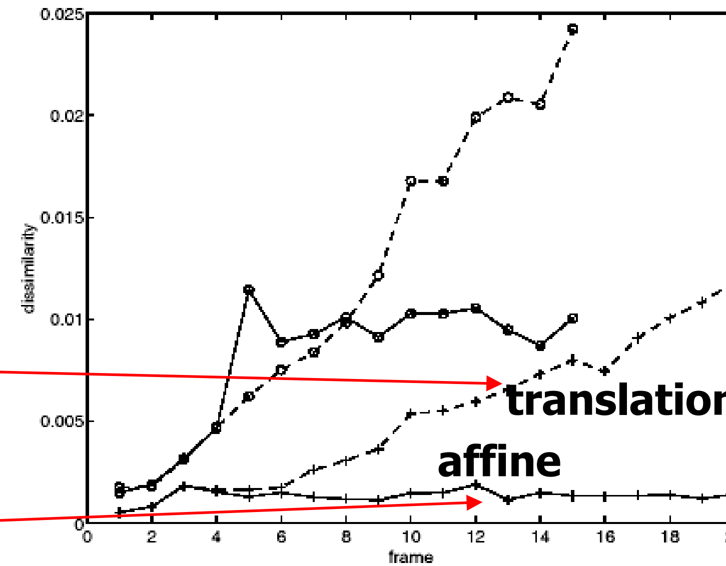
Good Features to Track

$$\iint_W \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} w(x,y)\,\mathrm{d}x\mathrm{d}y\Delta = \iint_W \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} (J-I)w(x,y)\,\mathrm{d}x\mathrm{d}y$$

- Solve equation by iterative minimization:
  - Linearize around current position (zero displacement)
  - Solve for displacement locally around point & iterate
  - Can be computed efficiently
- Can be extended to affine transformation as well
  - … but a bit more complex
  - Solve 6x6 instead of 2x2 system

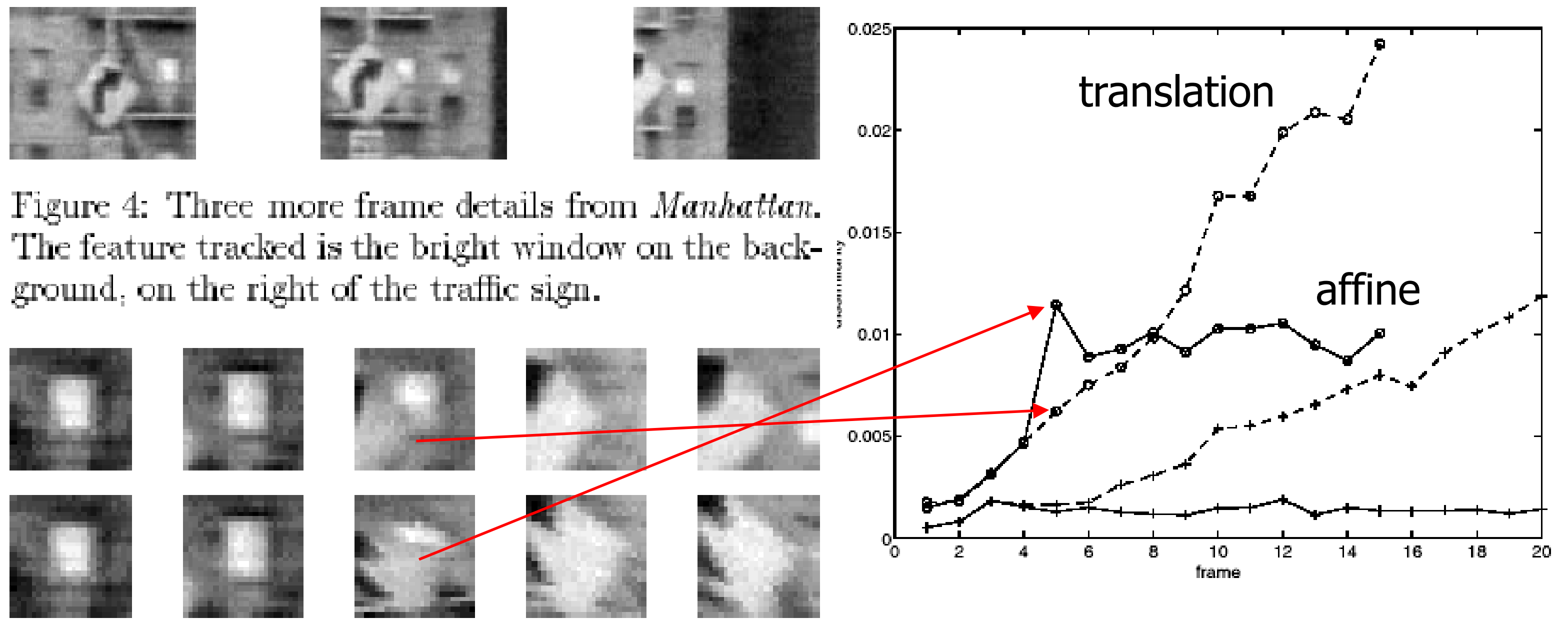
Example

Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.



Simple displacement is sufficient between consecutive frames, but not to compare to reference template

Example



- Problem: Affine model tries to deform sign to shape of window, tries to track this shape instead
- Solution: Perform affine alignment between first and last frame, stop tracking features with too large errors

Intensity Linearization

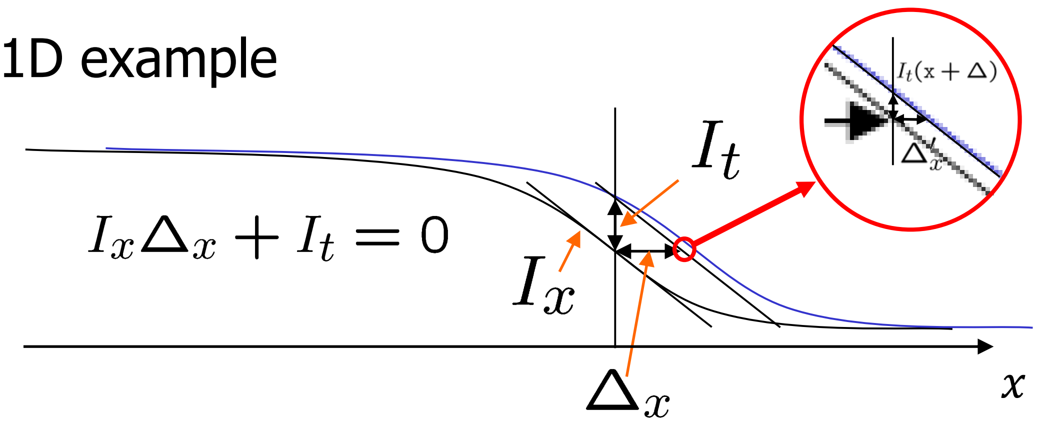
- Brightness constancy assumption:

$$I(x+\Delta_x, y+\Delta_y, t+1) = I(x,y,t)$$

$$I(x+u, y+v, t+1) = I(x,y,t) + \mathtt{I}_x\Delta_x + \mathtt{I}_y\Delta_y + \mathtt{I}_t \quad \text{(small motion)}$$

$$I_x\Delta_x + I_y\Delta_y + I_t = 0$$

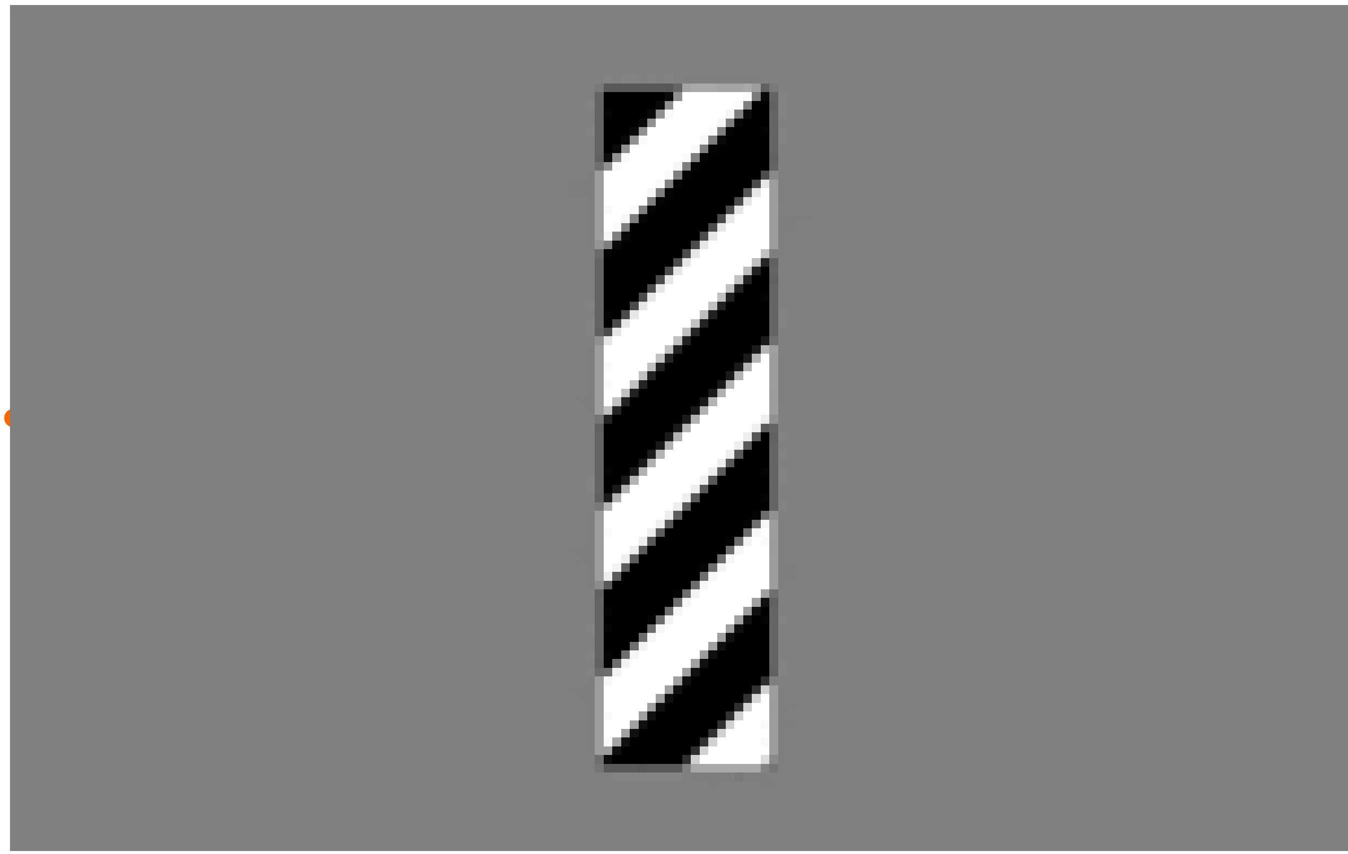
- 1D example



possibility for iterative refinement

Intensity Linearization

- Brightness constancy assumption



motion)

n

=I

Barberpole illusion (image source: Wikipedia)

Intensity Linearization

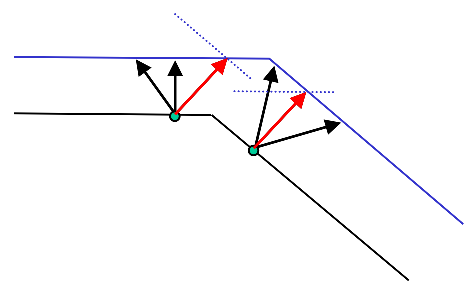
- How to deal with aperture problem?

$$R_x\Delta_x + R_y\Delta_y + R_t = 0 \quad G_x\Delta_x + G_y\Delta_y + G_t = 0 \quad B_x\Delta_x + B_y\Delta_y + B_t = 0$$

(3 constraints if color gradients are different)

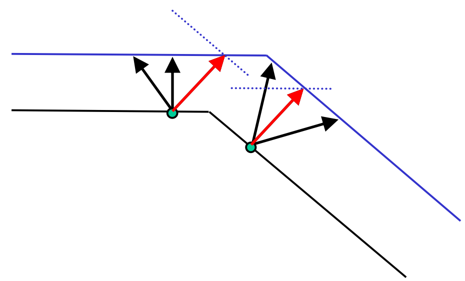
Assume neighbors have same displacement

$$I_x(\mathtt{x})\Delta_x + I_y(\mathtt{x})\Delta_y + I_t(\mathtt{x}) = 0 \quad I_x(\mathtt{x}')\Delta_x + I_y(\mathtt{x}')\Delta_y + I_t(\mathtt{x}') = 0 \quad \ldots$$



Lucas-Kanade

Assume neighbors have same displacement



least-squares:

$$\begin{bmatrix} I_x(\mathrm{x}) & I_y(\mathrm{x}) \\ I_x(\mathrm{x}) & I_y(\mathrm{x}) \\ I_x(\mathrm{x}) & I_y(\mathrm{x}) \end{bmatrix} \Delta = \begin{bmatrix} -I_t(\mathrm{x}) \\ -I_t(\mathrm{x}') \\ -I_t(\mathrm{x}'') \end{bmatrix}$$

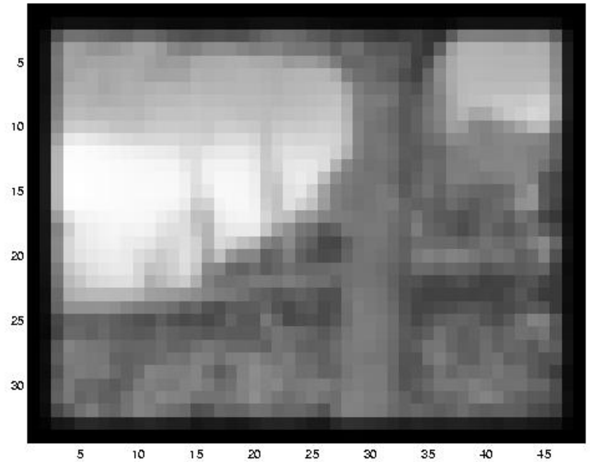
$$\mathbf{A}\Delta = \mathbf{b}$$

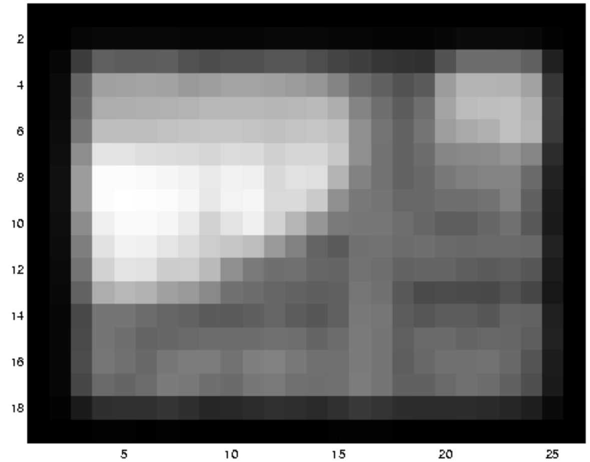
Revisiting the Small Motion Assumption

- Is this motion small enough?
  - Probably not—it's much larger than one pixel (1^{st} order Taylor not sufficient)
  - How might we solve this problem?

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Reduce the Resolution!

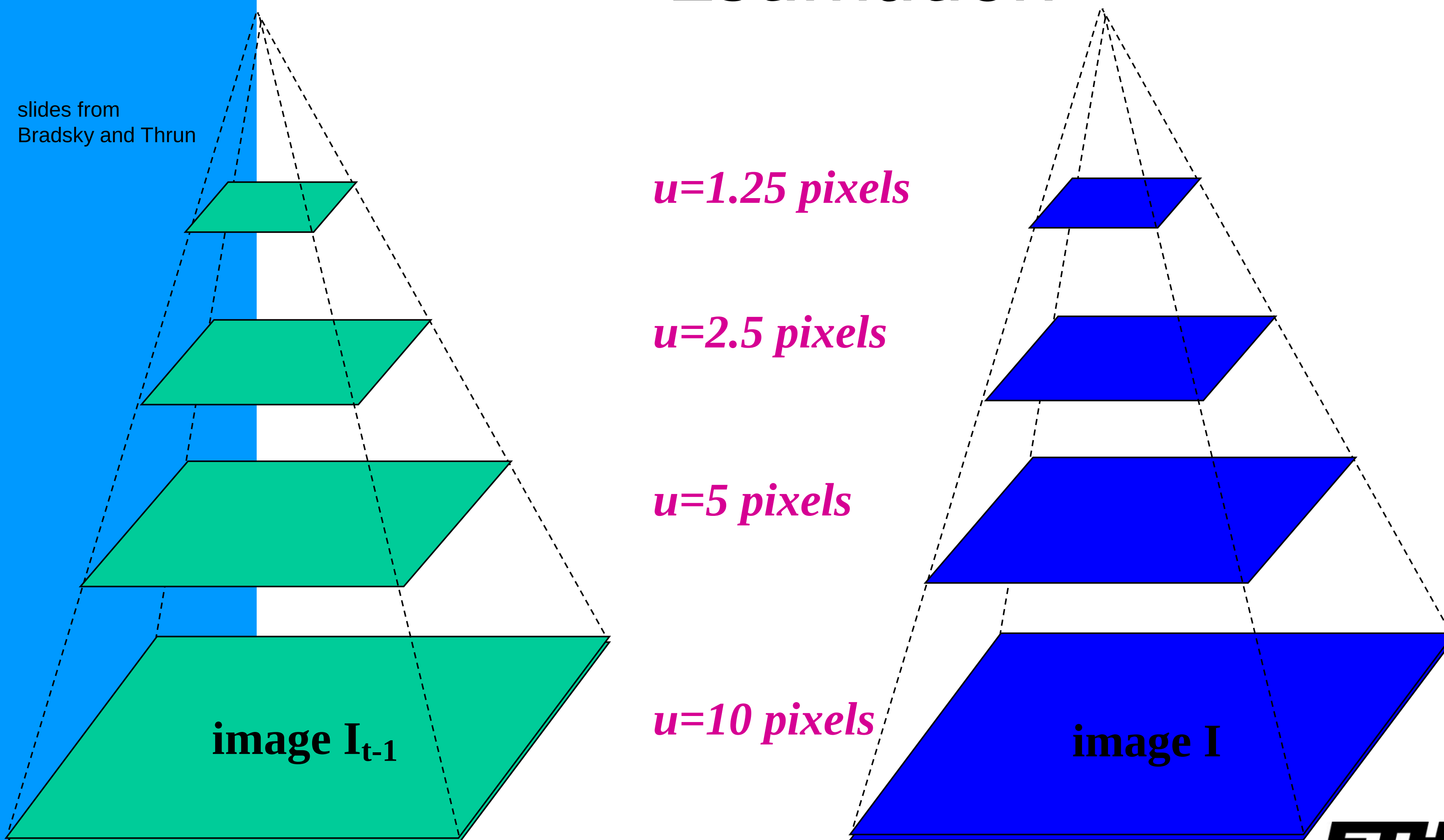




* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Coarse-to-Fine Optical Flow Estimation

slides from
Bradsky and Thrun

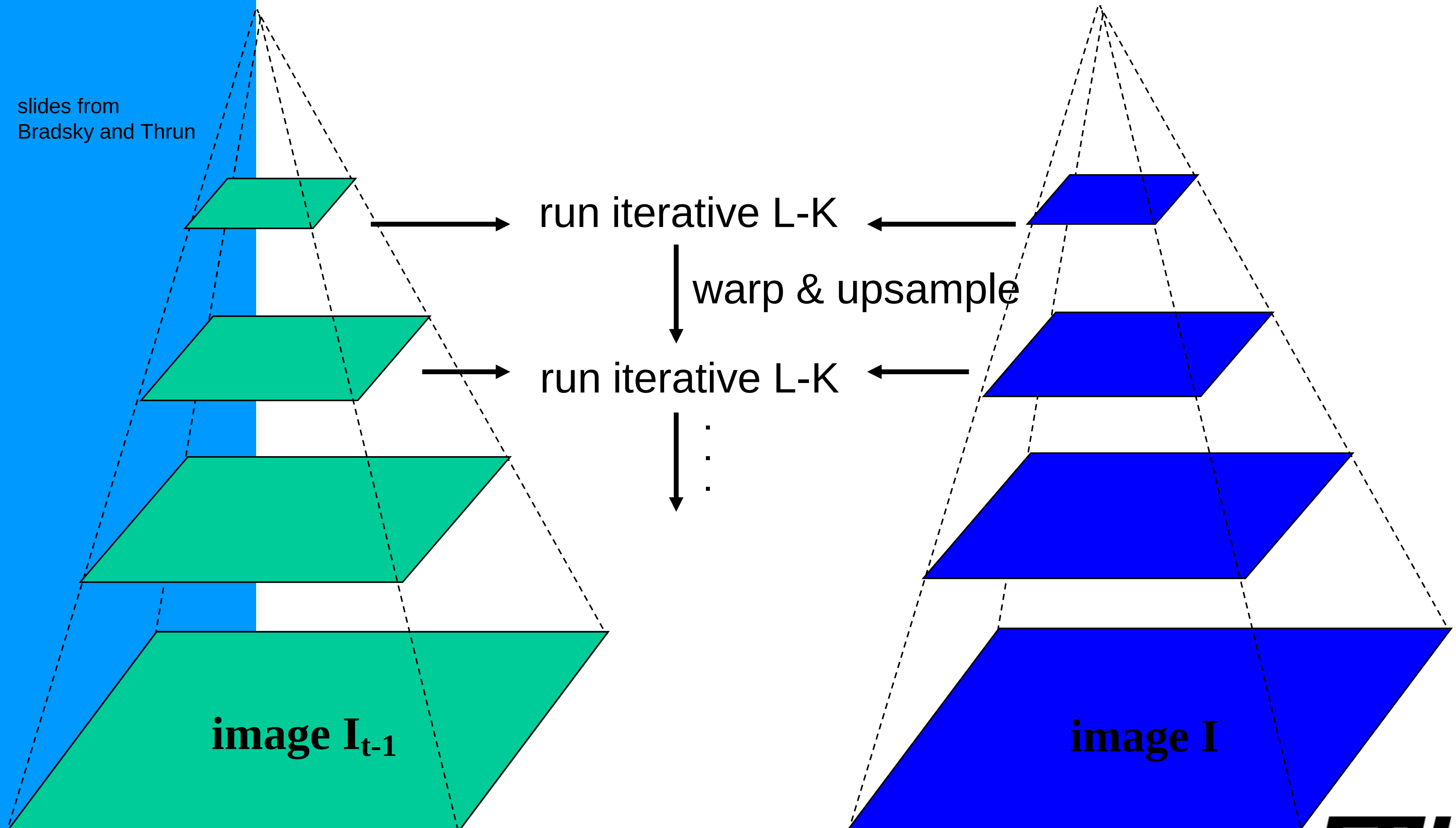


Gaussian pyramid of image I_{t-1}

Gaussian pyramid of image I

Coarse-to-Fine Optical Flow Estimation

slides from
Bradsky and Thrun



Gaussian pyramid of image I_{t-1}

Gaussian pyramid of image I

Summary Feature Tracking

- Motivation: Exploit small motion between subsequent (video) frames
- Key ideas:
  - Brightness constancy assumption
  - Linearize complex motion model and solve iteratively
  - Use simple model (translation) for frame-to-frame tracking
  - Compute affine transformation to first occurrence to avoid switching tracks

This Lecture

- **Feature detectors**: Reliably detect “interesting” regions in image under
  - Geometric transformations
  - Brightness changes
- **Feature descriptors**: Representation of patches
  - Input: Normalized patch from detector
  - Compute descriptor (=point in d-dimensional space)
  - Descriptor matching = approx. nearest neighbor search
- **Feature tracking**

Schedule

Feb 22	Introduction
Mar 1	Geometry, Camera Model, Calibration
Mar 8	**Features, Tracking / Matching**
Mar 15	**Project Proposals by Students**
Mar 22	Structure from Motion (SfM) + papers
Mar 29	Dense Correspondence (stereo / optical flow) + papers
Apr 5	Easter break
Apr 12	Bundle Adjustment & SLAM + papers
Apr 19	**Student Midterm Presentations**
Apr 26	Multi-View Stereo & Volumetric Modeling + papers
May 3	3D Modeling with Depth Sensors + papers
May 10	Guest lecture + papers
May 17	Guest lecture + papers
May 31	**Student Project Demo Day = Final Presentations**

Next week: Project Proposals

Reminder: Submit your proposal until Monday!

Reminder: Prepare short presentation for Monday!

(more details on Moodle!)