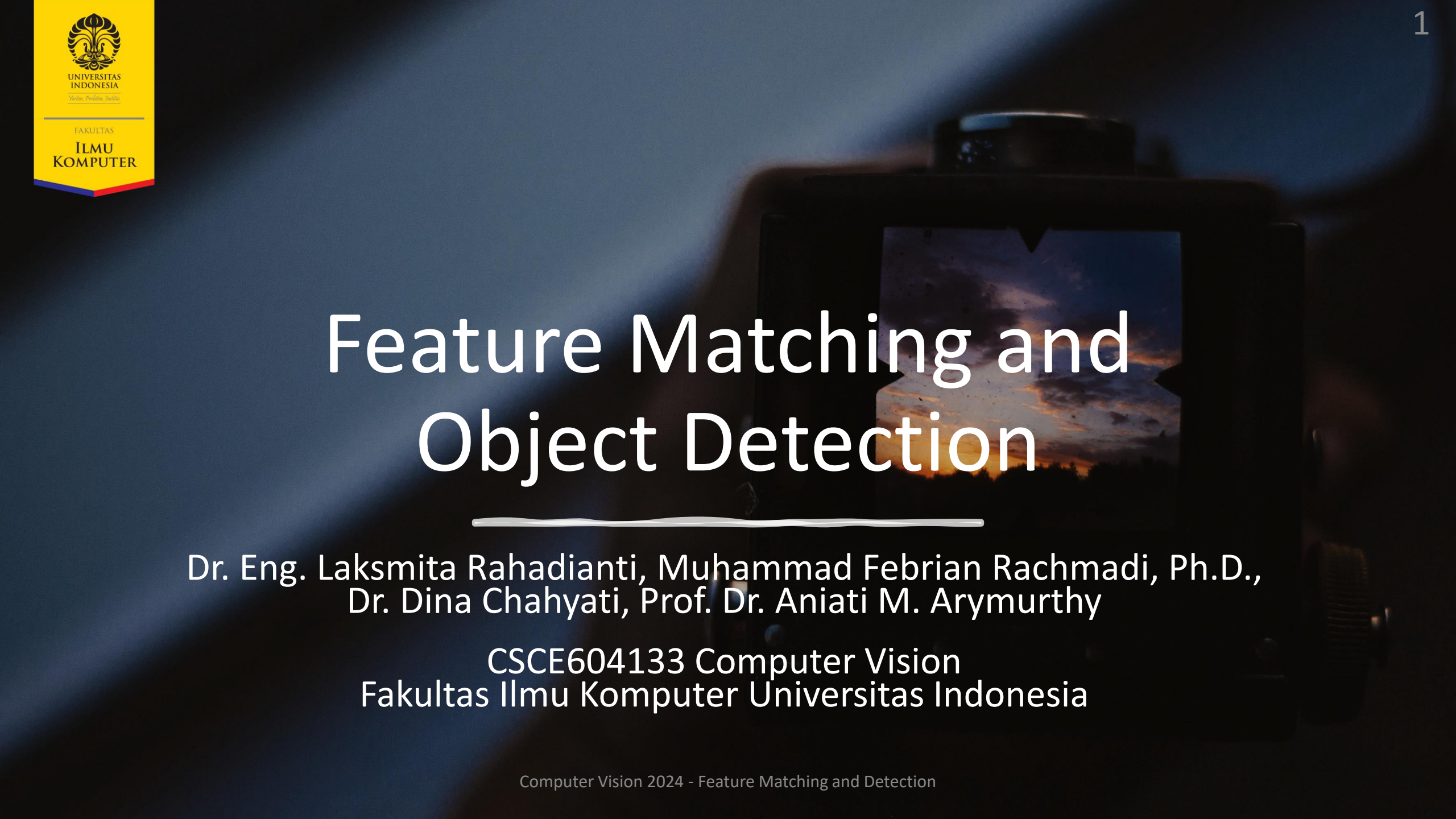


Feature Matching and Object Detection

Dr. Eng. Laksmita Rahadianti, Muhammad Febrian Rachmadi, Ph.D.,
Dr. Dina Chahyati, Prof. Dr. Aniati M. Arymurthy

CSCE604133 Computer Vision
Fakultas Ilmu Komputer Universitas Indonesia

Acknowledgements

- These slides are created with reference to:
  - Computer Vision: Algorithms and Applications, 2nd ed., Richard Szeliski https://szeliski.org/Book/
  - Digital Image Processing, Gonzales and Woods, 3rd ed, 2008.
  - Course slides for CSCE604133 Image Processing – Faculty of Computer Science, Universitas Indonesia
  - Introduction to Computer Vision, Cornell Tech https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html
  - Computer Vision, University of Washington https://courses.cs.washington.edu/courses/cse576/08sp/

Human vs Computer

- What you see

- What the computer sees

201	188	181	185	180	147	140	149	155	138	144	144	145
199	200	201	188	139	132	147	150	143	123	112	102	117
207	221	222	136	90	111	125	145	140	138	122	104	97
231	219	200	90	65	84	84	107	95	92	92	99	89
227	223	181	74	72	89	92	86	77	63	50	55	65
217	211	166	85	47	75	82	83	75	42	42	39	40
208	195	179	131	54	68	66	72	46	21	15	24	19
198	187	181	141	53	54	55	59	37	21	37	66	90
195	184	170	134	52	38	42	45	35	43	98	152	172
186	175	171	169	100	34	34	27	44	85	139	170	184
167	156	142	144	112	48	32	46	84	133	166	172	186
142	139	131	120	108	67	30	76	102	123	153	171	178
145	134	128	125	117	70	38	91	101	105	125	146	157

We need to be able to represent the image in a certain way so that the computer knows what it’s looking at

Image Features

- We use features as a representation of images.
- Features are attributes that:
  - are going to help us assign unique labels to objects in an image or
  - are going to be of value in differentiating images or families of images.
- A feature must be a **distinctive (unique)** attribute of something.



A (non-exhaustive) List of Features

Statistical Features

Histogram (Intensity, color, orientation)

Global Features

Hough Transform

Fourier Transform

Geometric Features

Shape extraction

Blobs and corners

Complex Features

SIFT (scale-invariant feature transform)

SURF (speeded up robust features)

Pixel Based

Edge detection

Texture

Embeddings

Embeddings

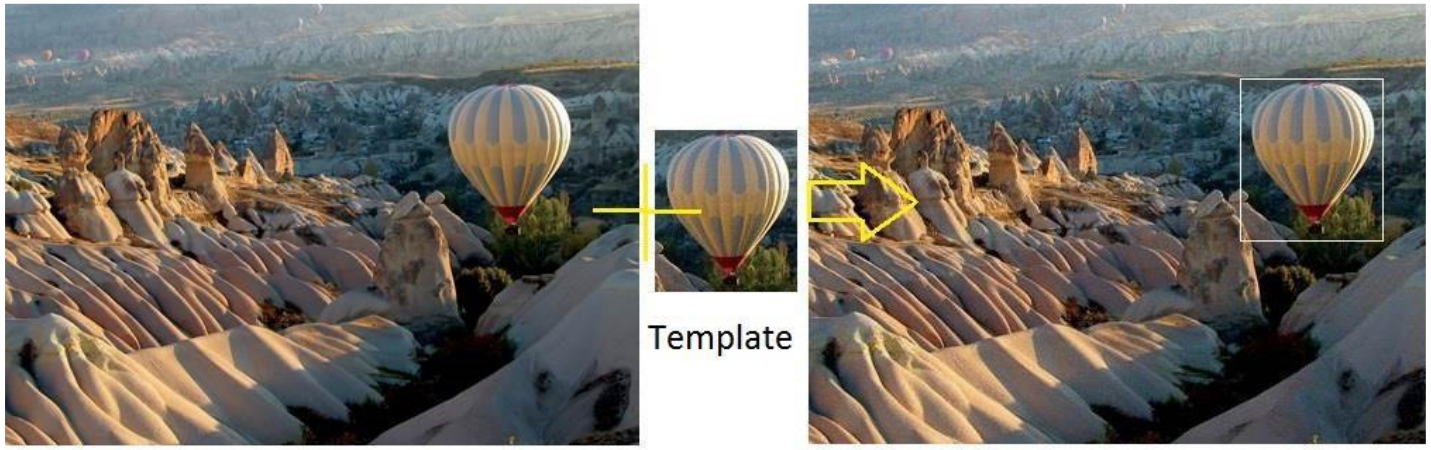
Learned by models

Template and Feature Matching

Template Matching

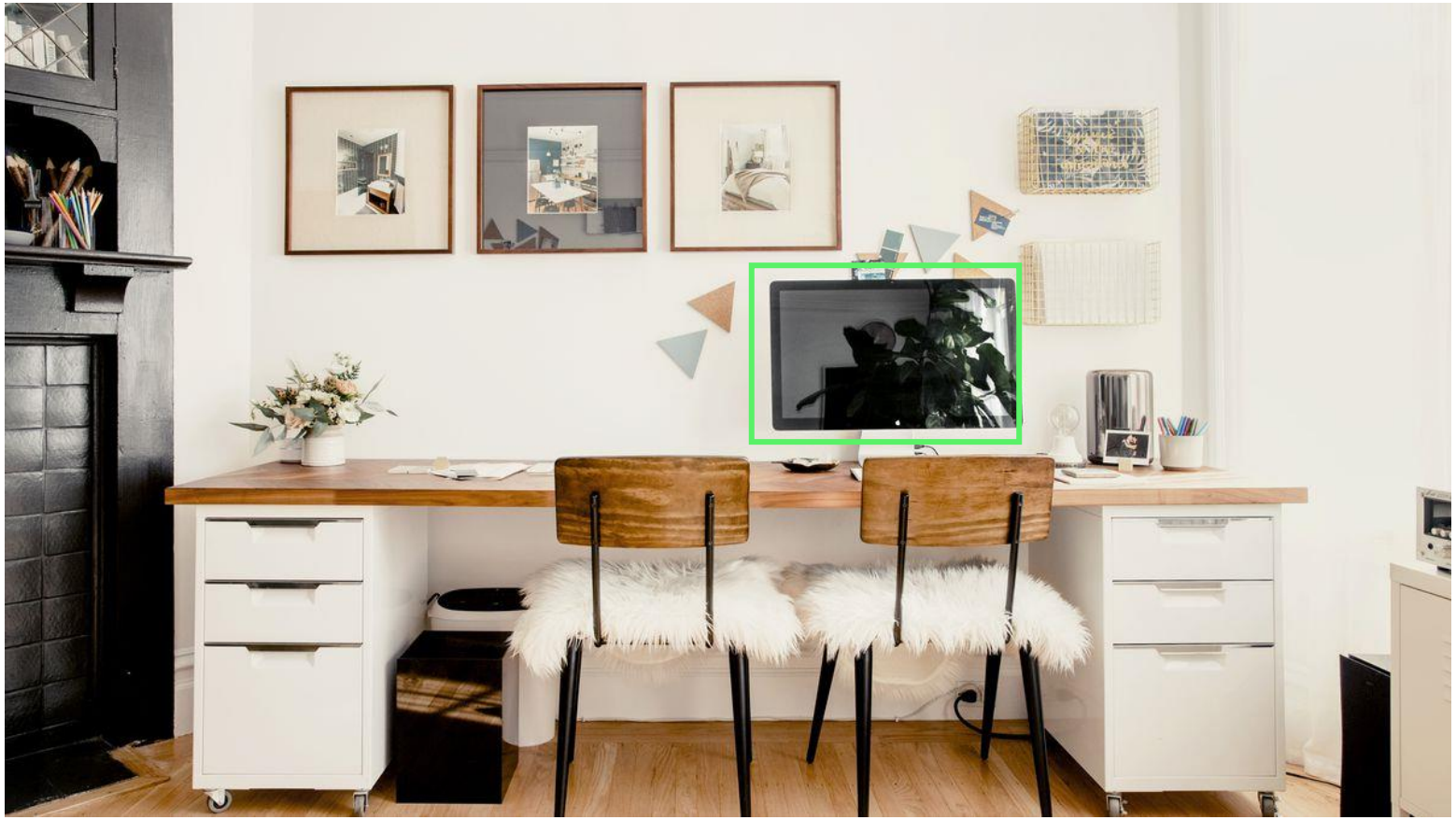
kita match gambar yang kita mau cari (jadi cari yang exact match, size orientation). Nah biasanya digunain untuk dapetin rodden (kayak hama) dalam sawah. Karena biasanya matchnya sama. Tapi downsidenya kalau bentuknya berbeda, maka gak bisa didetect.

- A method for finding the location of a template image in larger image.
- Slide the template image over the image (2D convolution) and compare the template and patch of image covered by the template

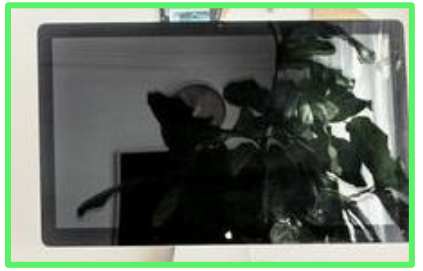


- ☹ Template matching can only find exact match (same size, same orientation)

Detection Tasks:



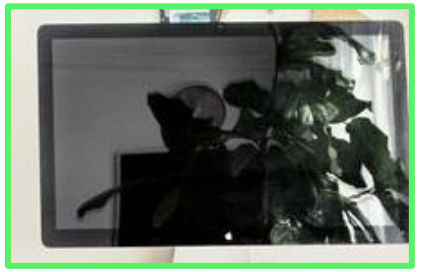
This is the monitor.



- Perhaps we could employ template matching……

Find the Monitor Now

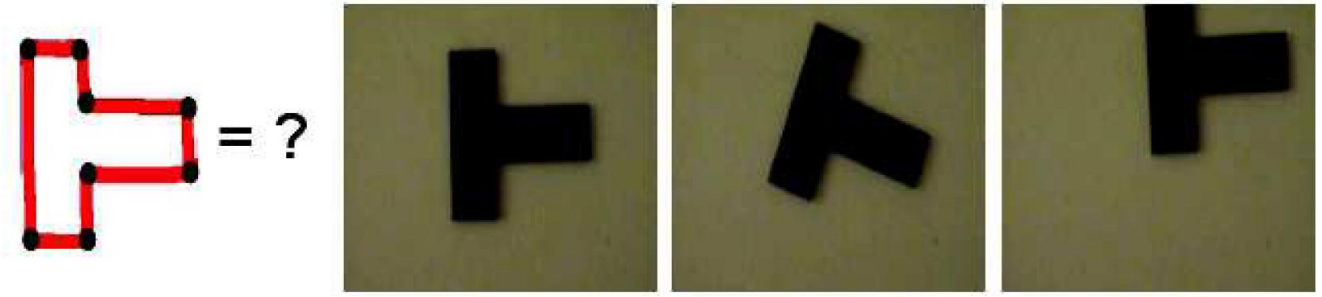
This is the monitor.



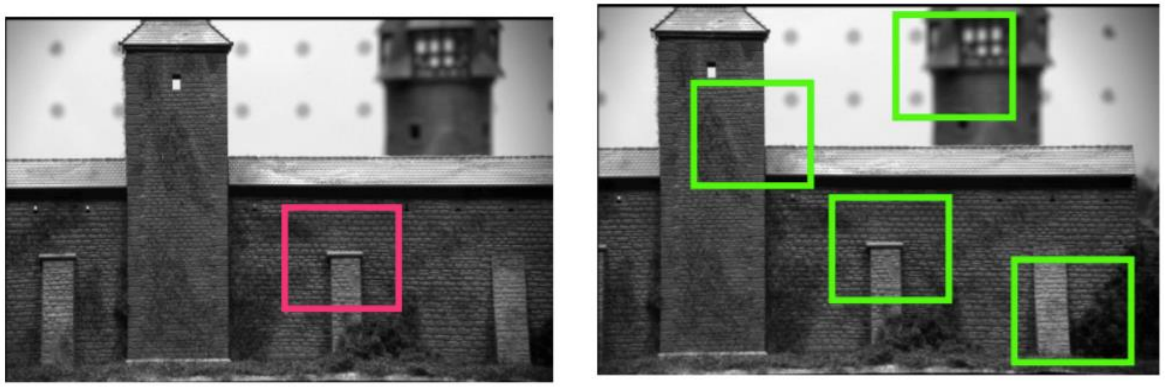
Template matching is not robust for variances in appearance.

Template and Patch Matching

- Template matching: we try to find a certain template.



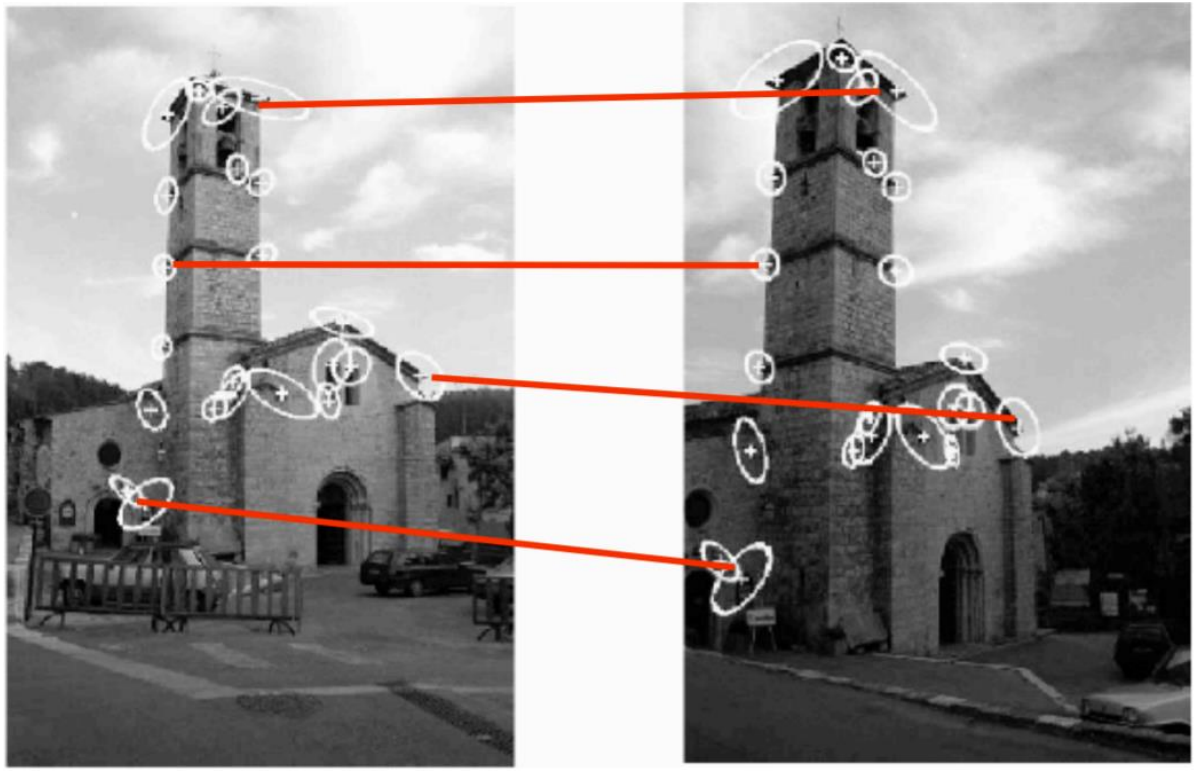
- Patch matching: Find the best (most similar) patch in a second image



Slide credit: Bob Fisher, Robert Collins

Matching Multiple Images

- Computer vision tasks such as stereo and motion estimation require finding **corresponding features** across two or more views.

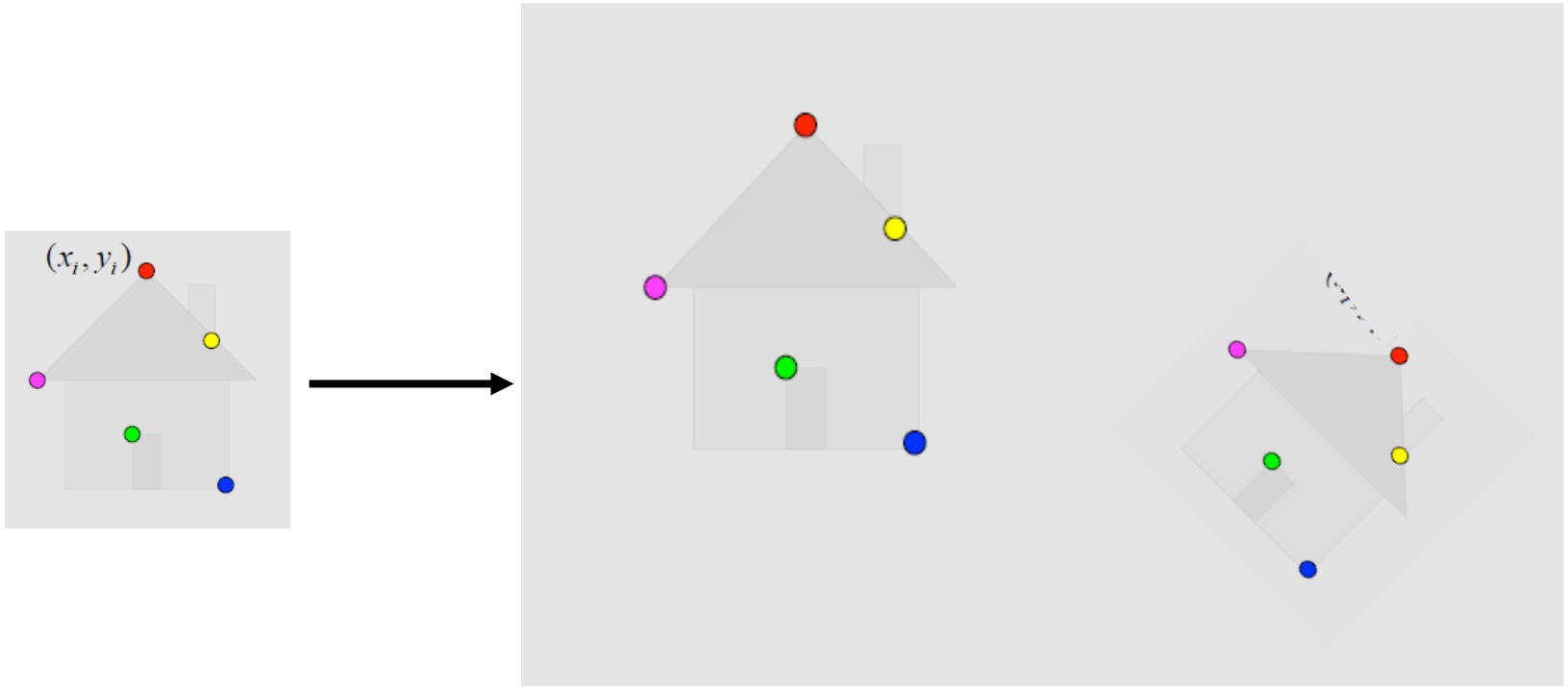


untuk nemuin kayak atasan pintu, walaupun dia beda tempat, rotasi, pencahayaan maka bisa di detect.

Nah kita bisa gunain feature correspondences.

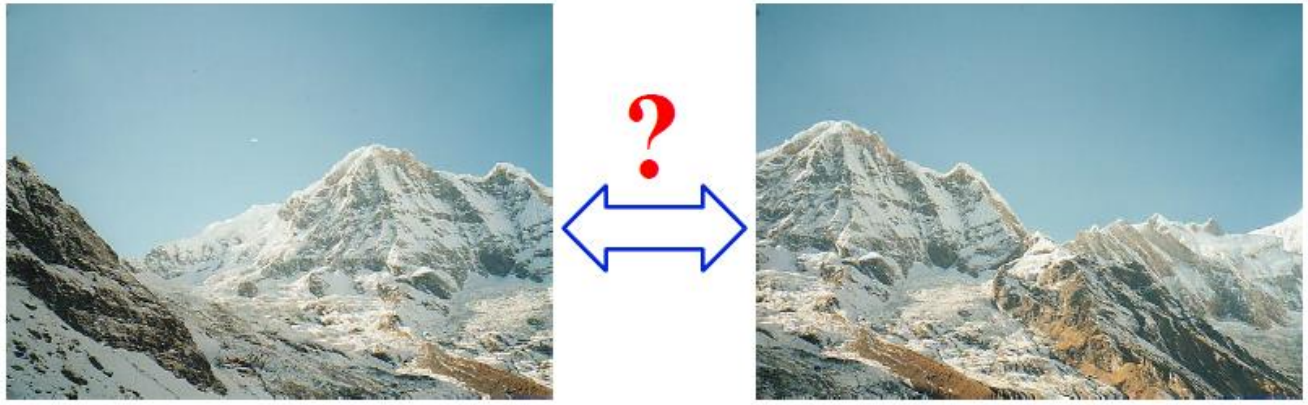
Feature Correspondences

- **Feature correspondences** are the foundation of template matching.

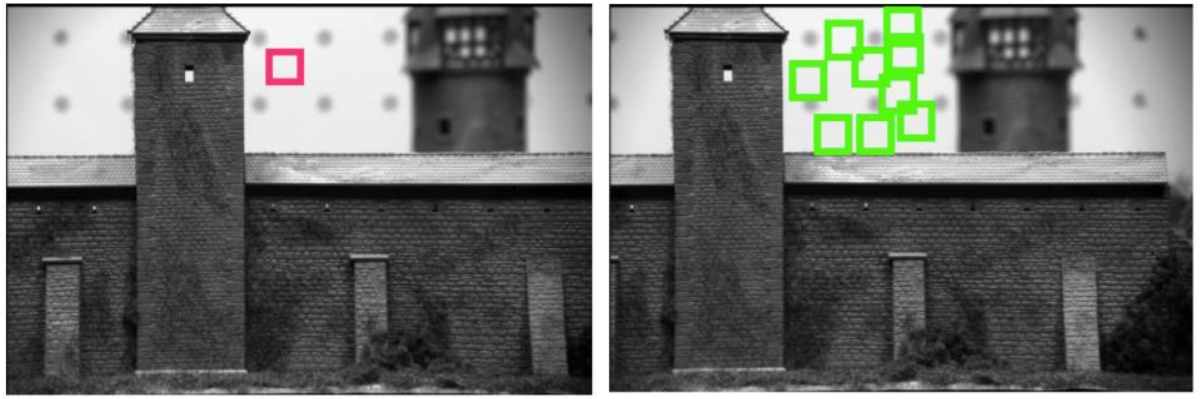


Finding Feature Correspondences

- How can we find the correspondences?

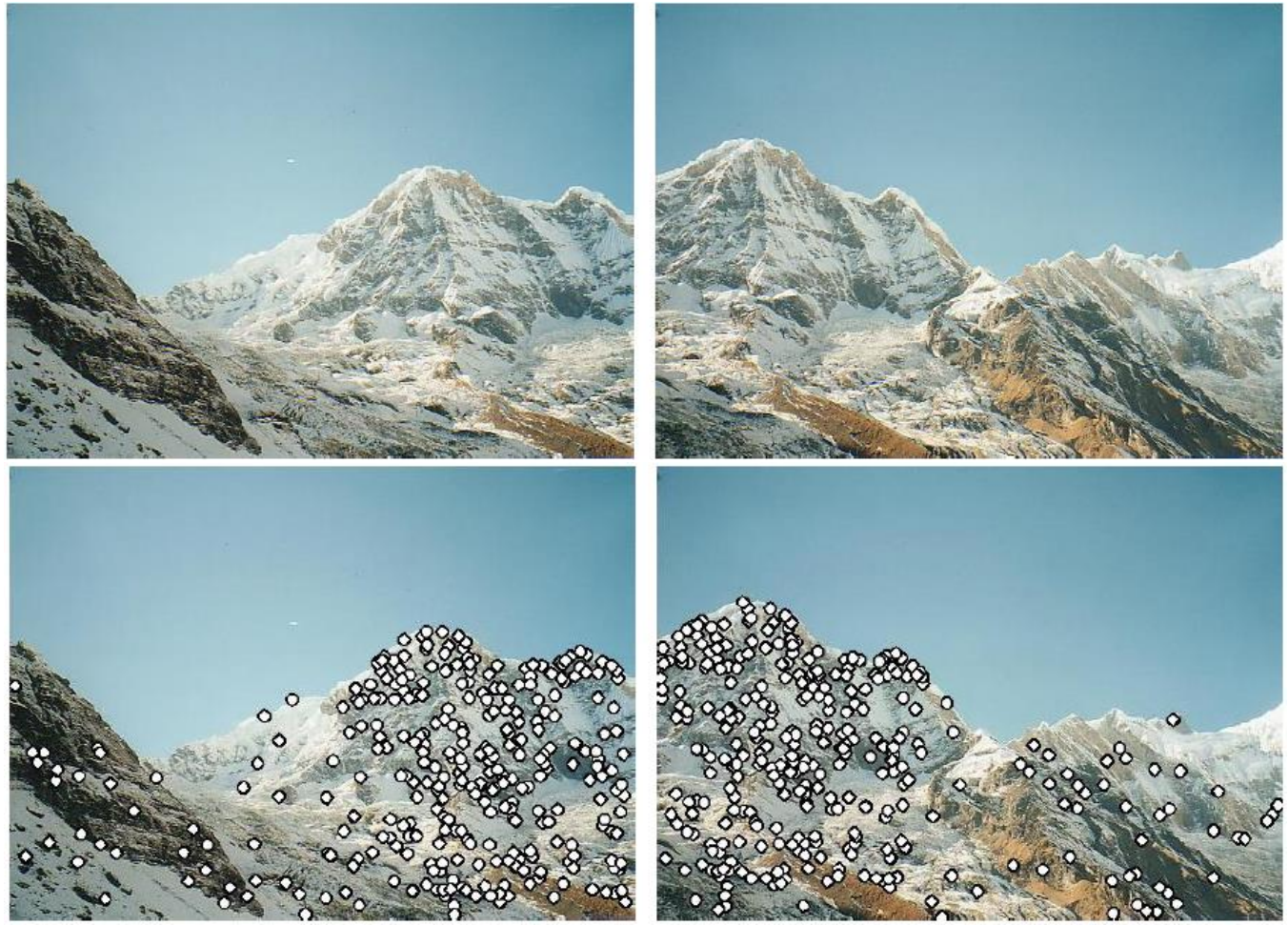


- Not all features are created equal!



Robust Feature-Based Fitting

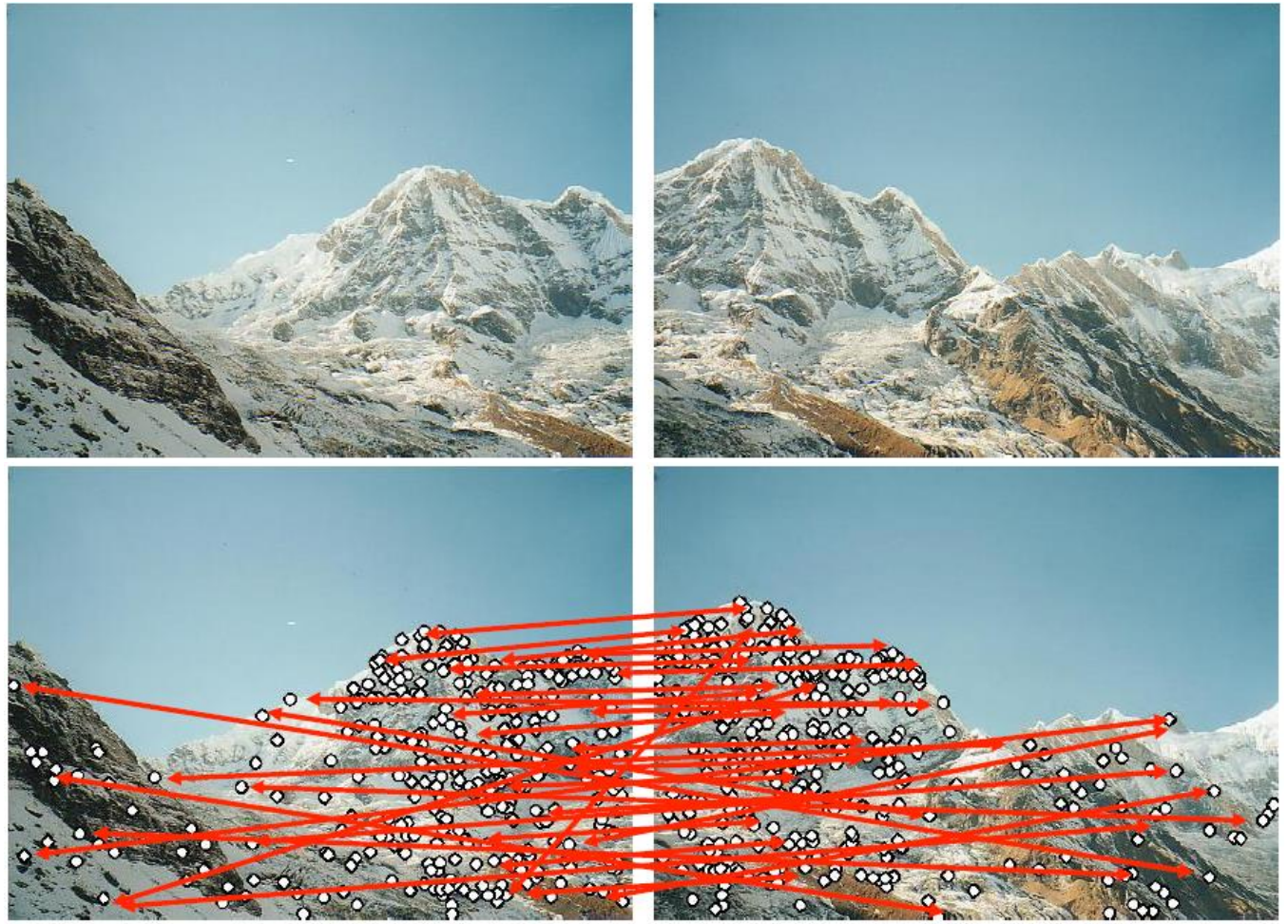
1. Extract features
2. Compute putative matches
3. Loop
   1) Hypothesize transformation T
   2) Verify transformation (search for other matches consistent with



Robust Feature-Based Fitting (2)

Jadi kita cari compute putative matches dari histogram yang ada. Terus dicari yang histogramnya paling match (gak harus exact)

1. Extract features
2. Compute putative matches
3. Loop
   1) Hypothesize transformation T
   2) Verify transformation (search for other matches consistent with



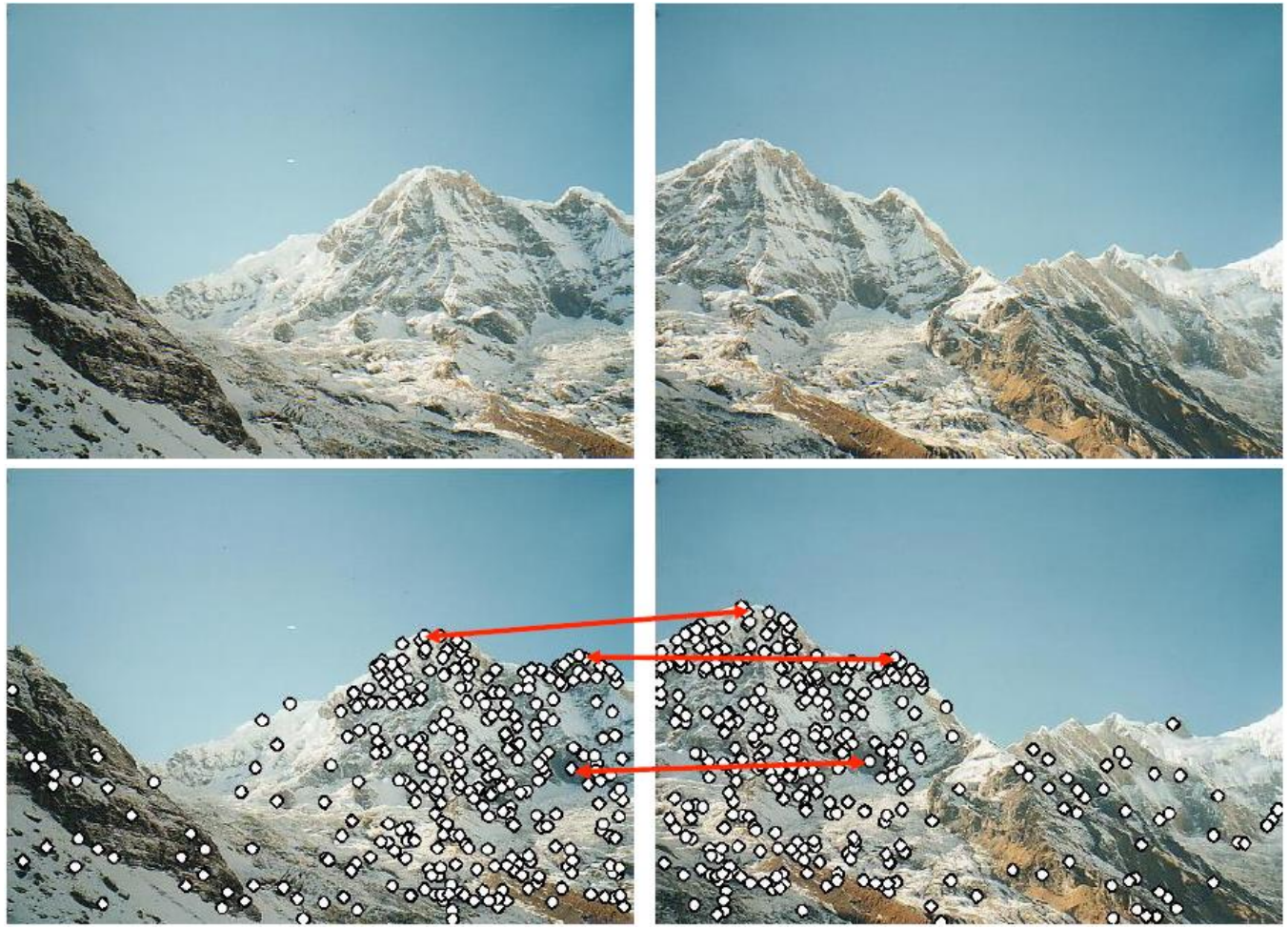
Robust Feature-Based Fitting (3)

1. Extract features
2. Compute
3. Loop
   1) Hypothesize transformation T
   2) Verify transformation (search for other matches consistent with T)

transformasi linear T, yang menmindahkan antara 1 titik lain menjadi titik lainnya (Baasically kayak SPL dimana titik (X, Y) itu berbeda nilainya dengan nilai (X2, Y2))

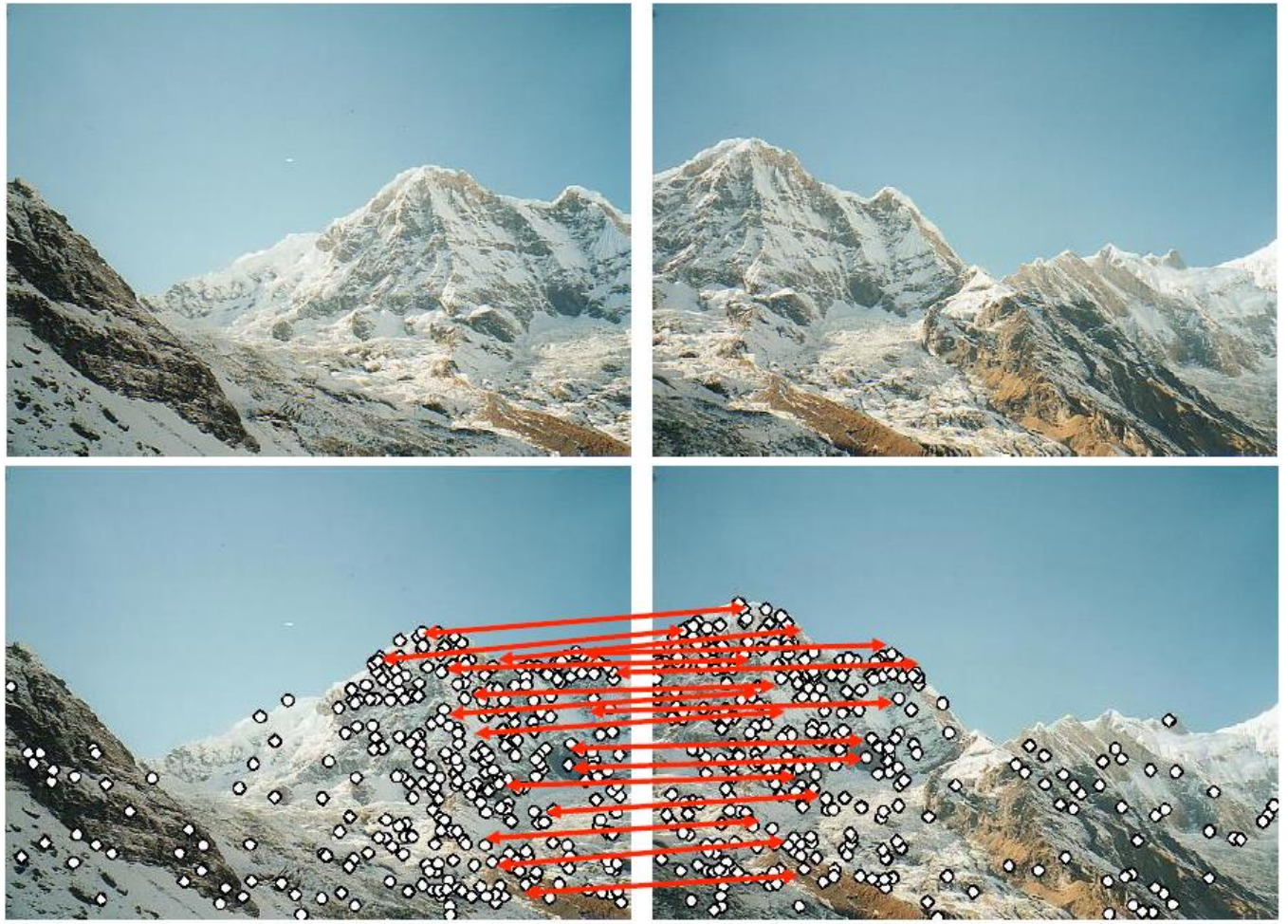
Mencari T yang paling banyak cocok

Biasanya dari T tersebut kita bisa combine kombinasi image tersebut utk jadi panorama



Robust Feature-Based Fitting (4)

1. Extract features
2. Compute
3. Loop
   1) Hypothesize transformation T
   2) Verify transformation (search for other matches consistent with T)



Robust Feature-Based Fitting Result

1. Extract features
2. Compute putative matches
3. Loop
   1) Hypothesize transformation T
   2) Verify transformation (search for other matches consistent with T)

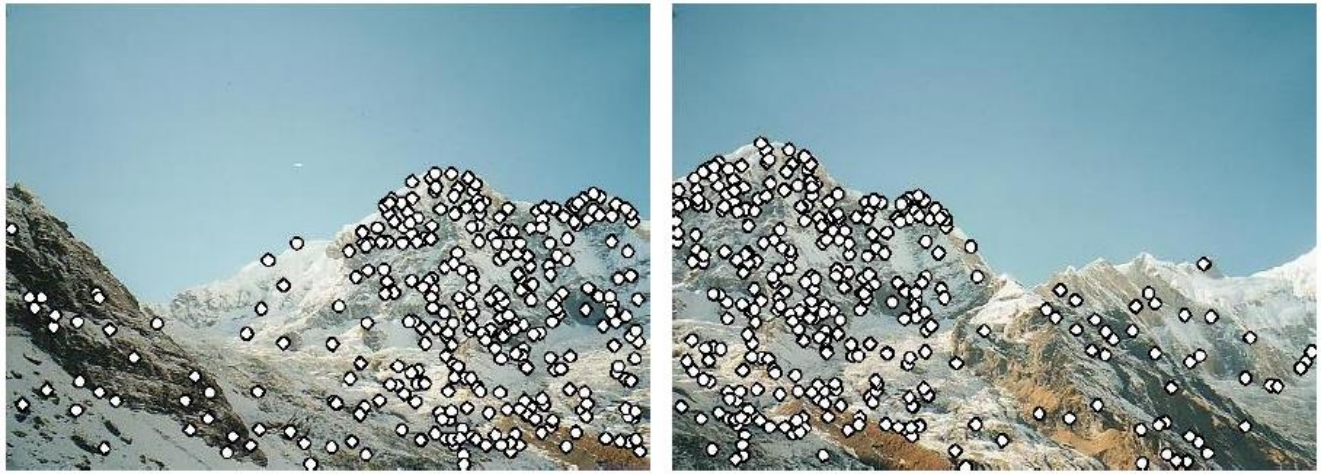
Bagaimana cara dapat titik titik/feature yang ada di image

This is the basic idea of RANSAC
(RANdom Sample Consensus)

Characteristics of Good Features

Semua karakteristik untuk menjadi sebuah fitur.

1. **Repeatability**: Corresponding features can be found in several images despite geometric and photometric transformations
2. **Distinctiveness**: Each feature is distinctive
3. **Compactness and efficiency**: Many fewer features than image pixels
4. **Locality**: A feature occupies a relatively small area of the image; robust to clutter and occlusion



Applications

- Template/image matching applications:
  - Image alignment
  - 3D reconstruction
  - Robot navigation
  - Indexing and database retrieval
- Features commonly used are corners.
  - Independent reading: Blobs
- Feature description commonly used: SIFT.
  - Independent reading: SURF, KAZE, etc.

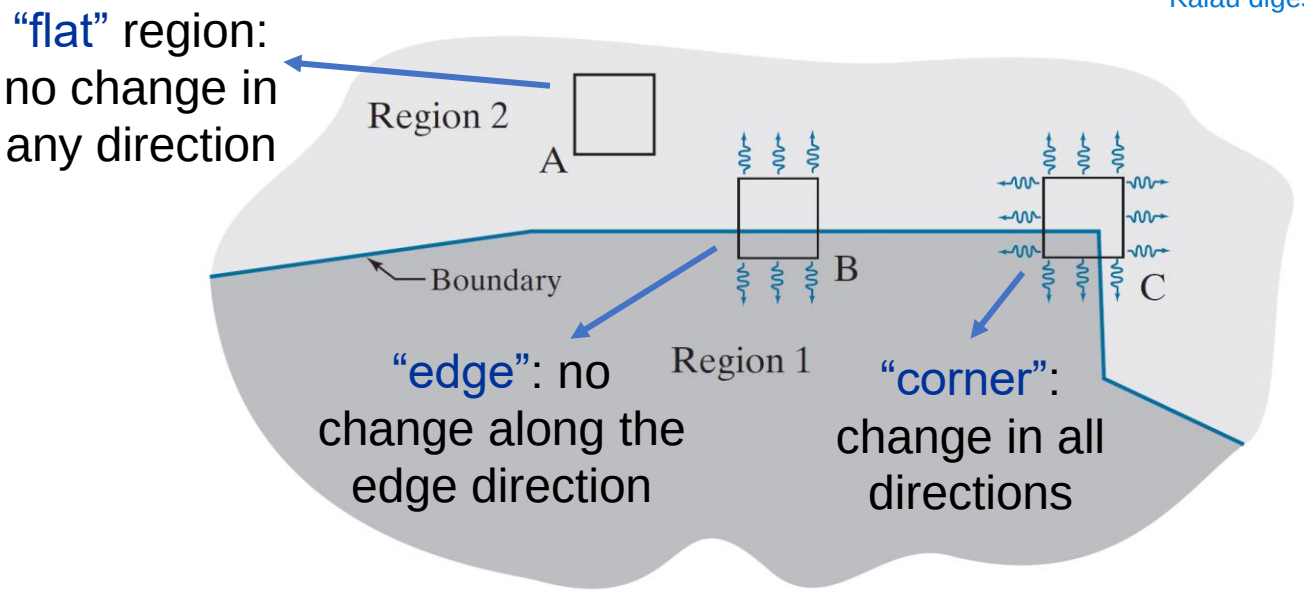
Slide credit: Svetlana Lazebnik, Vittorio Ferrari, Chris Williams

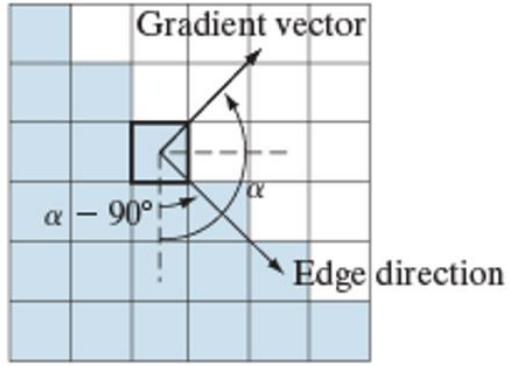
Feature Invariance and Covariance

Salah satu cara untuk menentukan fitur.

Corner Detection: The Idea

Region B
Kalau berubah keatas bawah berubah
Kalau kanan dan kiri digeser, gak berubah
Region C:
Kalau digeser, akan berubah kesagala arah





- We should easily recognize the point (of corner) by looking at intensity values through a small window. Shifting a window in *any direction* should give a *large change* in intensity.
- **Key property**: In the region around a corner, image gradient has two or more dominant directions. Corners are repeatable and distinctive.

Slide credit: A. Efros, Svetlana Lazebnik, Vittorio Ferrari, Chris Williams

Harris Corner Detection in Brief

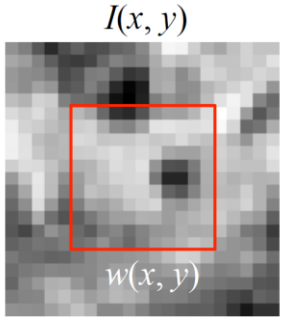
Mengambil Local Patch Area , misal 3x3, 3x2, dll.

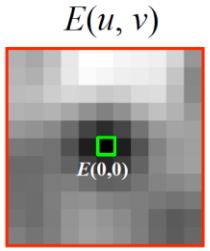
Terus diobservasi perubahan terjadi apabila ada shift u, v.

Yang kita mau adalah fitur yang berubah secara signifikan apabila ada perubahan arah.

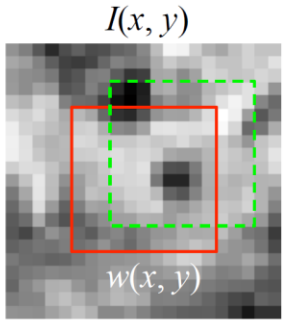
- Compute the change in appearance of window $w(x, y)$ for the shift $[u, v]$:

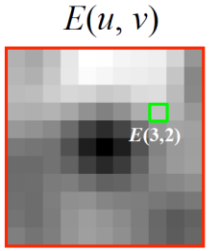
For $u = 0, v = 0$:



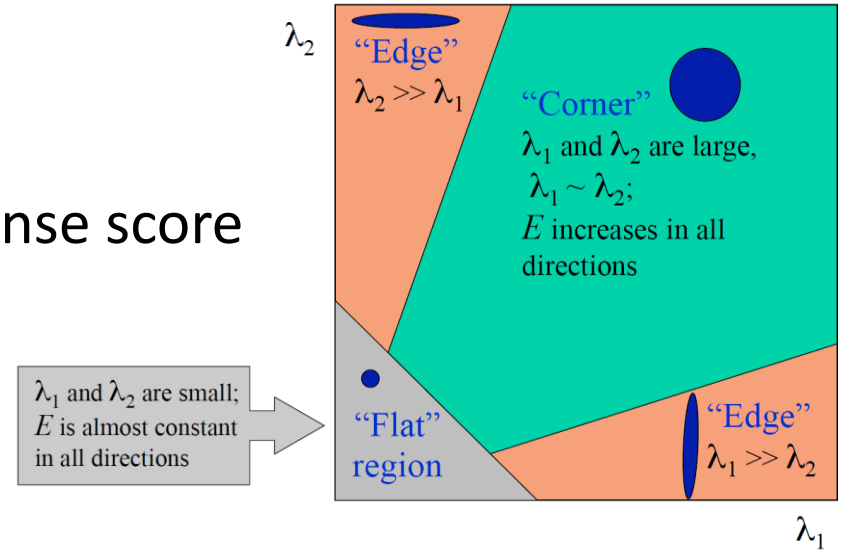


For $u = 3, v = 2$:





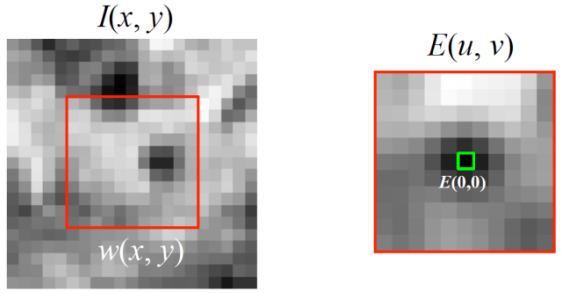
- Compute *R* as a score to detect a corner: Harris response score



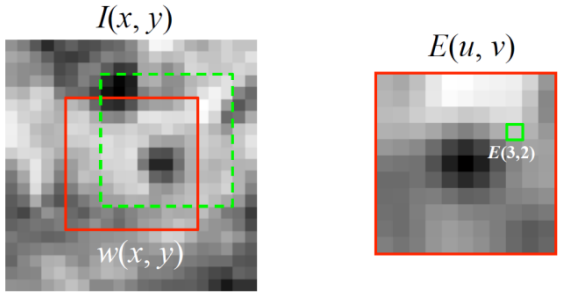
[Mathematics] Harris Corner Detection (1/3)

- Change in appearance of window $w(x, y)$ for the shift $[u, v]$:

For $u = 0, v = 0$:



For $u = 3, v = 2$:



$$E(u,v) = \sum_x \sum_y w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

Window function (weight)

Shifted intensity

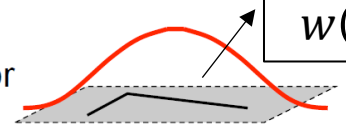
Intensity

For nearly *constant patches*, this will be *near 0*.
For nearly *distinctive patches*, this will be *larger*.
Hence, we want patches where $E(u, v)$ is LARGE

Window function $w(x,y)$ =

1 in window, 0 outside

or



Gaussian

$$w(s,t) = e^{-(s^2+t^2)/2\sigma^2}$$

Slide credit: Vittorio Ferrari, Chris Williams, Robert Collins

[Mathematics] Harris Corner Detection (2/3)

$$E(u,v) = \sum_x \sum_y w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

- The shifted patch can be approximated by the linear terms of a Taylor expansion:

$$I(x+u, y+v) \approx I(x,y) + uI_x(x,y) + vI_y(x,y)$$

where $I_x(x,y) = \partial I/\partial x$ and $I_y(x,y) = \partial I/\partial y$, both evaluated at (x,y).

- We can then write the equation as

$$E(u,v) = \sum_x \sum_y w(x,y)\left[uI_x(x,y) + vI_y(x,y)\right]^2$$

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} \mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\text{where, } \mathbf{M} = \sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

https://docs.opencv.org/master/dc/d0d/tutorial_py_features_harris.html

Slide credit: Vittorio Ferrari, Chris Williams

[Mathematics] Harris Corner Detection (3/3)

- **Next:** Use R as a score to detect a corner

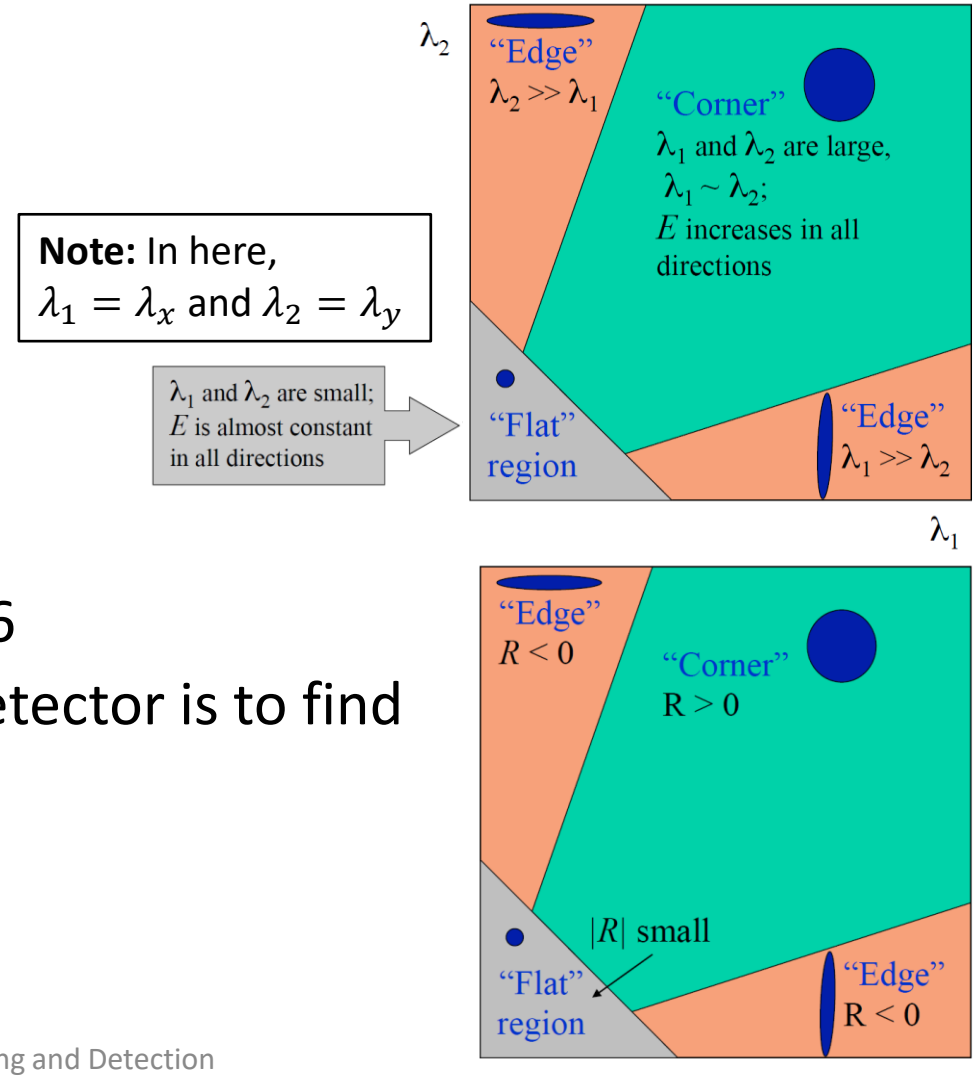
$$R = \det(\mathbf{M}) - k(trace(\mathbf{M}))^2$$

where,

  - $\det(\mathbf{M}) = \lambda_x \lambda_y$
  - $trace(\mathbf{M}) = \lambda_x + \lambda_y$
  - λ_x and λ_y are the eigenvalues of $\mathbf{M}$
  - k is a sensitivity factor, usually 0.04 to 0.06
    - The smaller k is, the more likely the detector is to find corners.

- R is usually called Harris response score

Note: In here, $\lambda_1 = \lambda_x$ and $\lambda_2 = \lambda_y$



Slide credit: Vittorio Ferrari, Chris Williams

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Harris Corner Detection - Visualized

Color to grayscale

Compute Gaussian derivatives at each pixel - $w(x, y)$

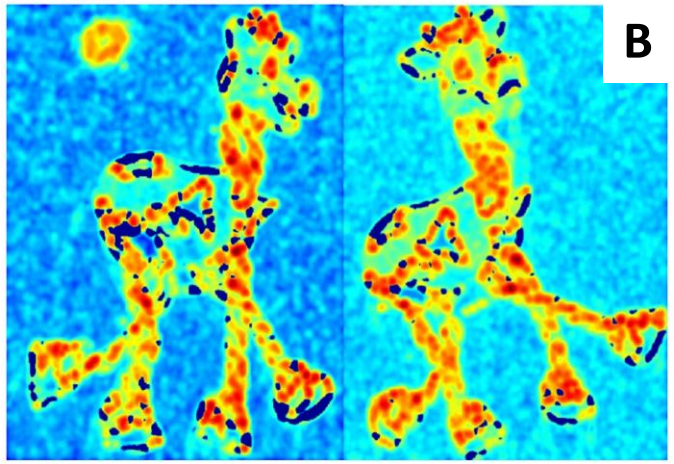
Compute second moment matrix **M** in a Gaussian window around each pixel

Compute corner response function *R* (Harris response score)

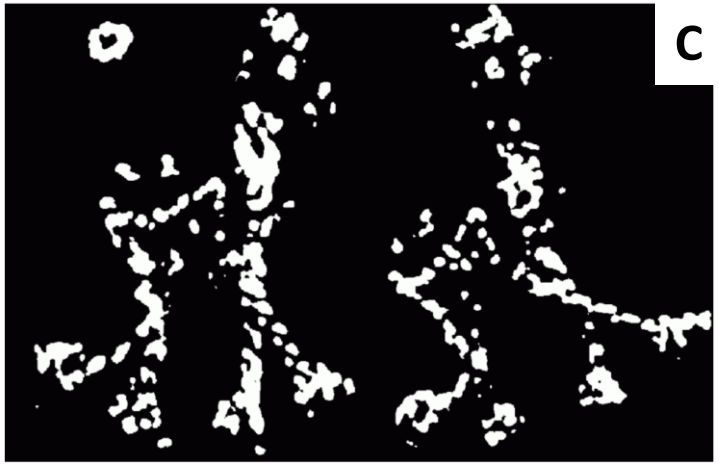
Threshold *R* using *T*

Find local maxima of response function (non-maximum suppression)

Compute corner response *R*



Find points with large corner response: *R*>threshold



Slide credit: Svetlana Lazebnik, Vittorio Ferrari, Chris Williams

Invariance and Equivariance

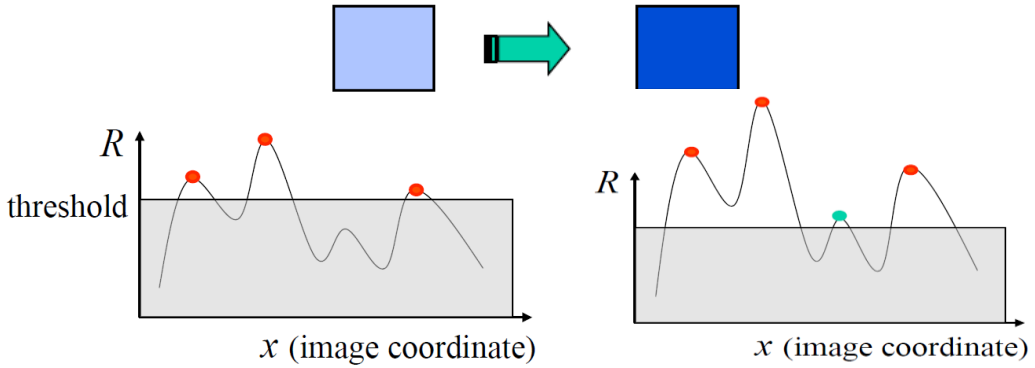
- We want corner locations to be *invariant* to photometric transformations and *equivariant* to geometric transformations.
  - **Invariance:** Image is transformed, and feature locations do not change.
  - **Equivariance:** If we have two transformed versions of the same image, features are detected in corresponding (different) locations.
    - Note: Often the term **covariance** is used instead.

covariance, fitur yang ada disebelah kiri, pasti ada juga dibagian kanan.

Invariance and Equivariance of Image Features

Affine Intensity Change

- Intensity change: $I \rightarrow aI + b$
  a. Intensity shift: $I \rightarrow I + b$
  b. Intensity scaling: $I \rightarrow aI$

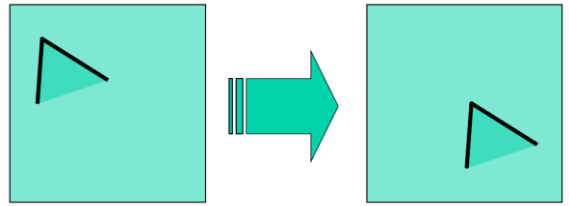


Corners are partially equivariant to affine intensity change

Image Translation

kalau ada translasi berubah, maka corner pasti termasuk covariance (equivariance. Corner location pasti tidak berubah (equivaraiant)

- Just like convolution, derivatives and window function are translation invariant.

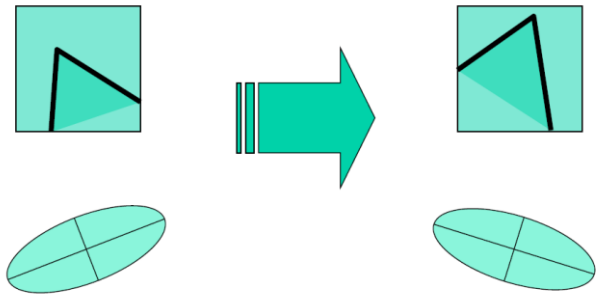


Corner location is equivariant with respect to translation

Invariance and Equivariance of Image Features (2)

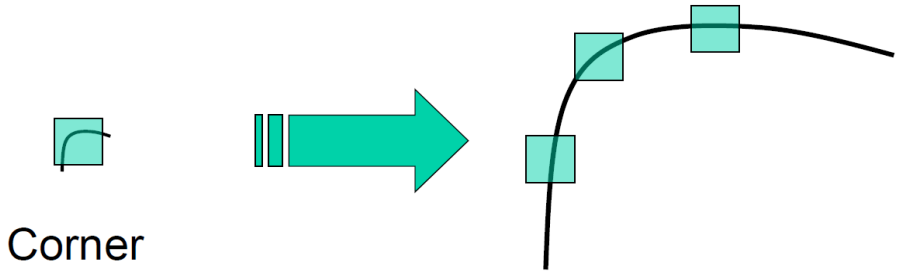
Image Rotation

- Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same.



Corner location is equivariant with respect to rotation

Scaling Size and scale change

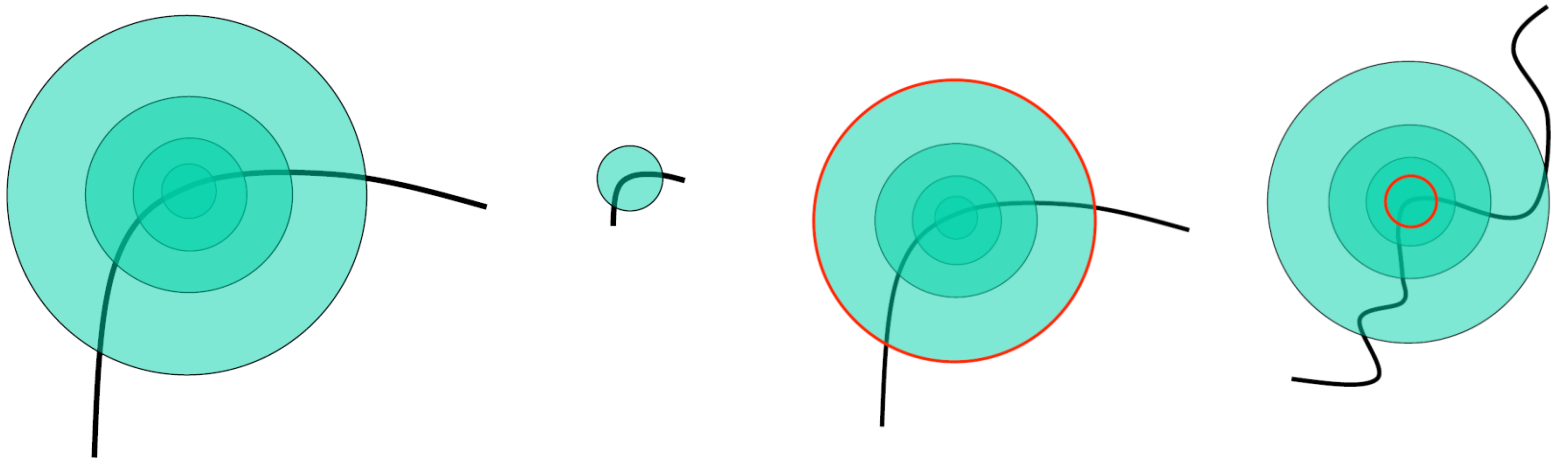


All points will be classified as edges

Corner location is **not** equivariant to scaling!

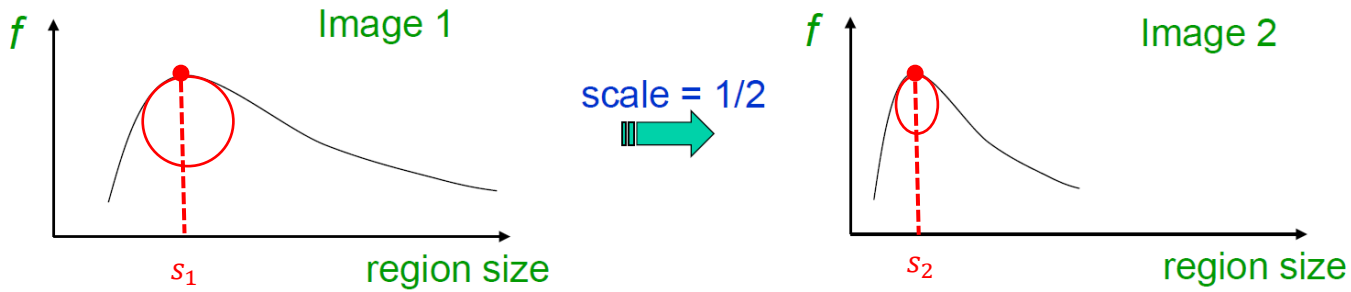
Scale Equivariant Detectors?

- Note: Regions of corresponding sizes will look the same in both images.
- A possible approach (close to brute force search):
  - Consider regions (e.g., circles) of different sizes around a point.
- **The problem:** How do we choose corresponding circles independently in each image?



Scale Equivariant Detectors

- Design a function on the region (circle), which is “scale equivariant” (the same for corresponding regions, even if they are at different scales)
- For a point in one image, we can consider it as a function of region size (circle radius)
- **How?**
  - Take a local maximum of some function.
  - Observation: region size, for which the maximum is achieved, should be equivariant with image scale.

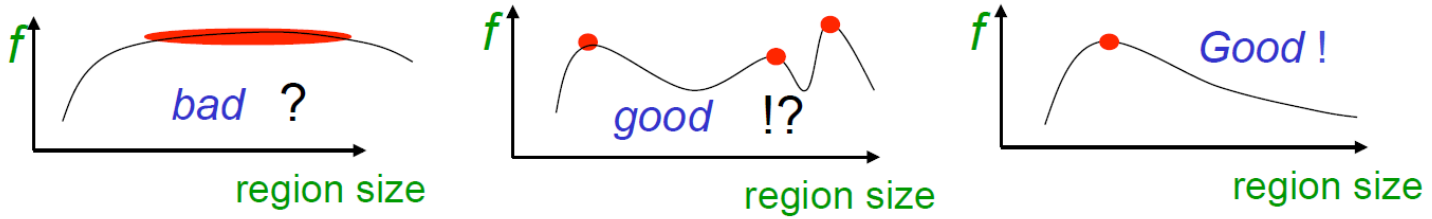


this scale equivariant region size is found in each image independently!

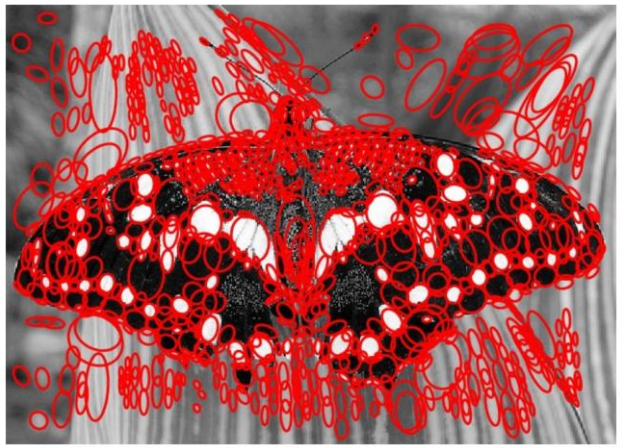
Slide credit: Vittorio Ferrari, Chris Williams

Scale Covariant Detectors

- A “good” function for scale detection: has stable sharp peak(s)

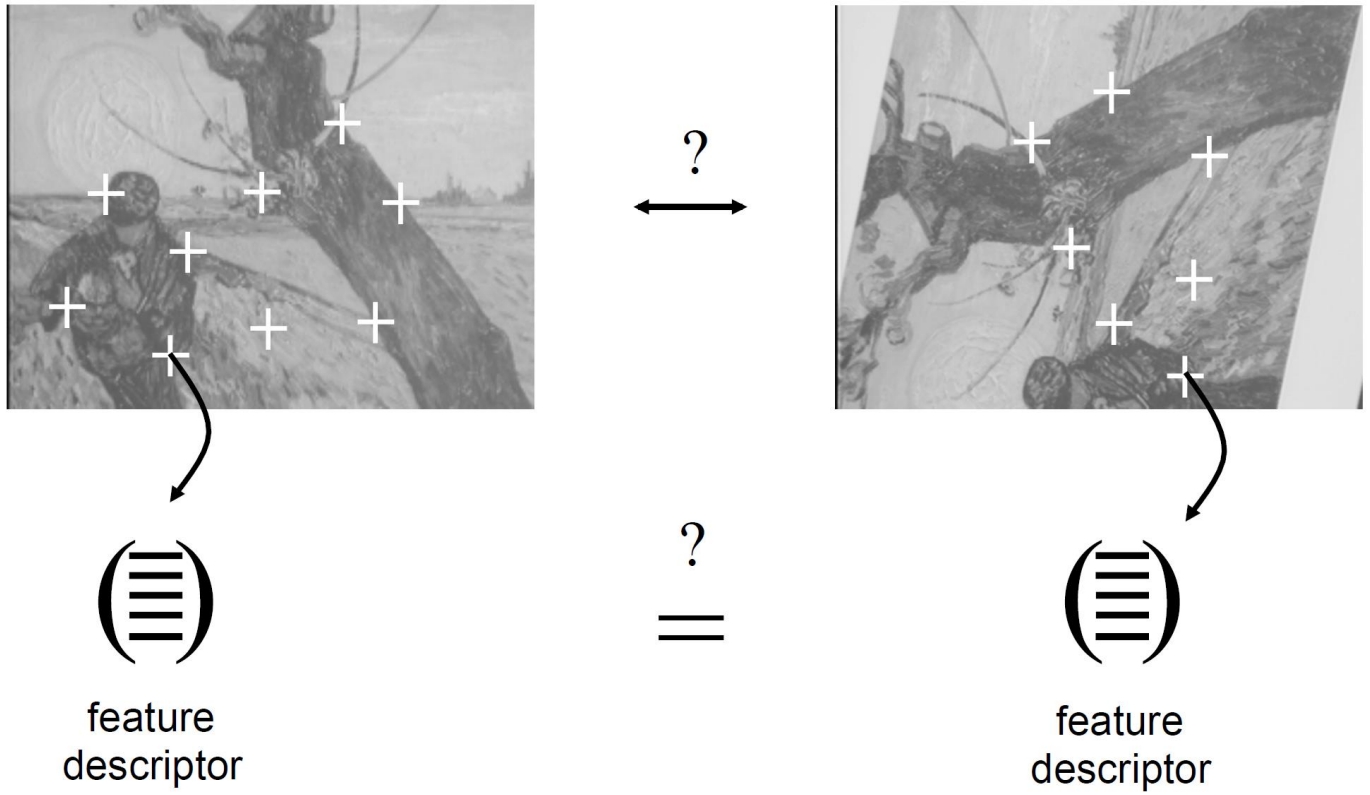


- For usual images: a good function would be a one which responds to contrast (sharp local intensity change)
- **Independent reading:**
  - *Scale equivariant blob detection*



Slide credit: Svetlana Lazebnik, Vittorio Ferrari, Chris Williams

Motivation: Putative Correspondences



- We need to compare *feature descriptors* of local patches surrounding interest points

Slide credit: Vittorio Ferrari, Chris Williams

Equivariant Detection and Invariant Description

Corner tidak covariant terhadap scaling. Namun covariance terhadap rotasi, translasi, etc.

- What to do if we want to compare the appearance of these image regions?
  - Note: In figures below, *scale equivariant blob detection* is used.

Images: K. Mikolajczyk

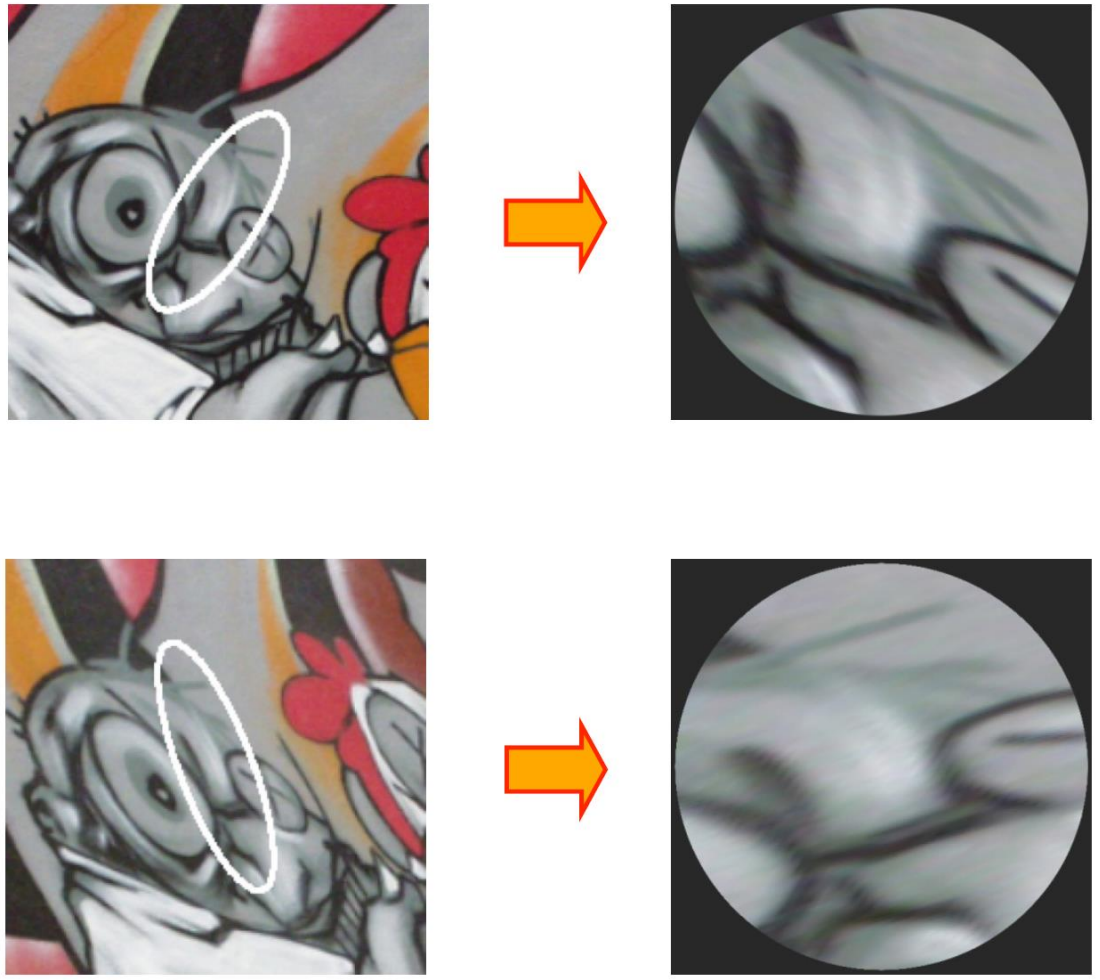
pokoknya equivariance dan invariance/covariance itu gak bisa dengan scaling. (contoh kalo scaling artinya corner location bisa aja berubah)

- Geometrically transformed versions of the same neighborhood will give rise to regions that are related by the same transformation.
  - **Normalization:** transform these regions into same size circles.

Slide credit: Vittorio Ferrari, Chris Williams

Affine Normalization

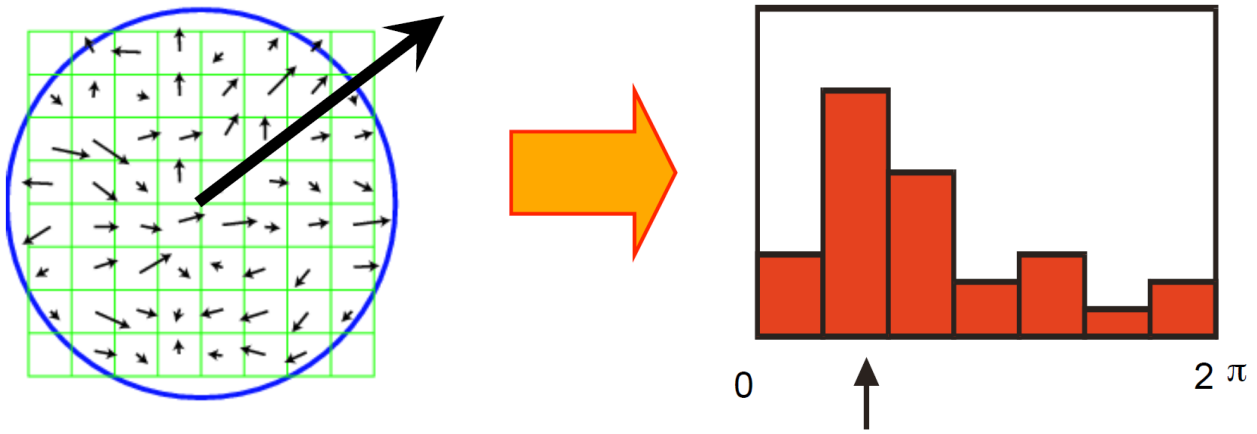
- **Problem:** There is no unique transformation from an ellipse to a unit circle.
  - We can rotate or flip a unit circle, and it still stays a unit circle.



Slide credit: Vittorio Ferrari, Chris Williams

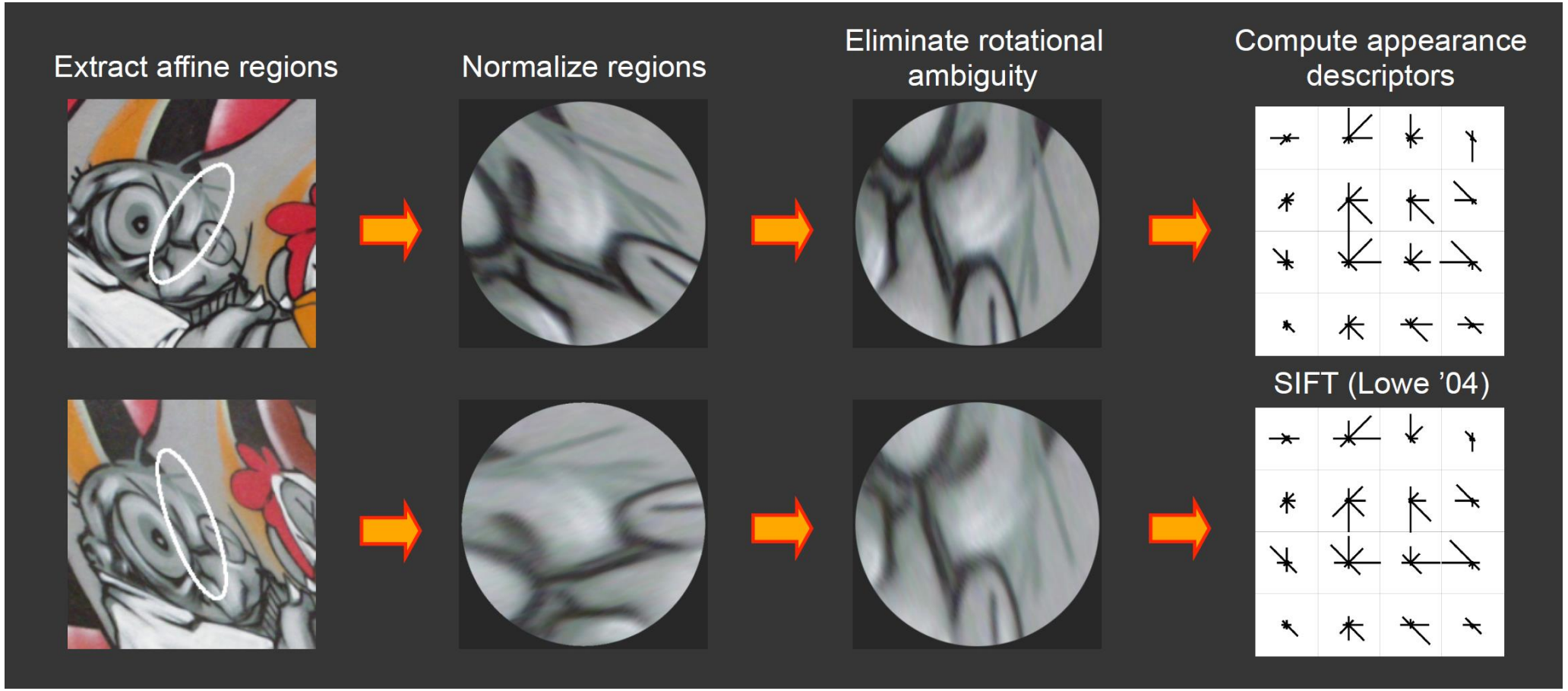
Eliminating Rotation Ambiguity

- To assign a unique orientation to circular image windows:
  1. Create histogram of local gradient directions in the patch.
  2. Assign canonical orientation at peak of smoothed histogram.



Slide credit: Svetlana Lazebnik, Vittorio Ferrari, Chris Williams

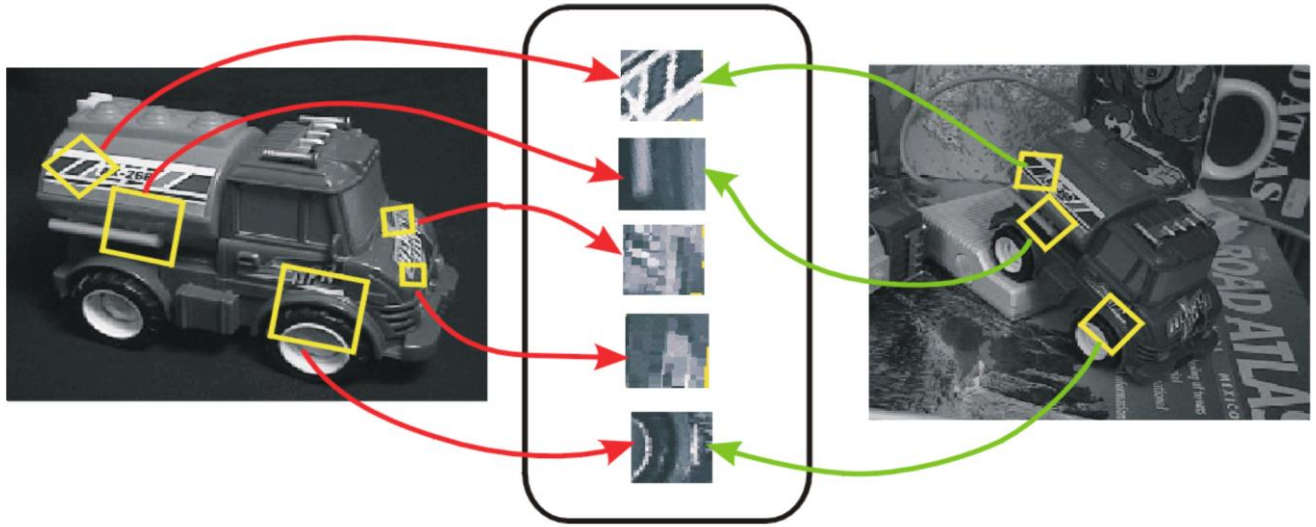
From Equivariant Regions to Invariant Features



Slide credit: Svetlana Lazebnik, Vittorio Ferrari, Chris Williams

Invariance vs. Equivariance of Features

- **Invariance Feature Detection:**
  - features(transform(image)) = features(image)
- **Equivariance Feature Detection :**
  - features(transform(image)) = transform(features(image))
- ***What we would like to have:***
  - Equivariant detection (e.g., corners, blobs) → Invariant description



Slide credit: Svetlana Lazebnik, David Lowe, Vittorio Ferrari, Chris Williams

Simplest Feature Descriptors

- **Simplest descriptors:** Vector of raw intensity values
- How to compare two such vectors?
  - Sum of squared differences (SSD):

$$SSD(u, v) = \sum_i (u_i - v_i)^2$$

    - Not invariant to intensity change
  - Normalized cross-correlation (NCC)

$$\rho(u, v) = \frac{\sum_i (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\left(\sum_j (u_j - \bar{u})^2\right) \quad \left(\sum_j (v_j - \bar{v})^2\right)}}$$

    - Invariant to affine intensity change

Slide credit: Vittorio Ferrari, Chris Williams

Disadvantage of Intensity Vectors as Descriptors

- Small deformations can affect the matching score a lot.

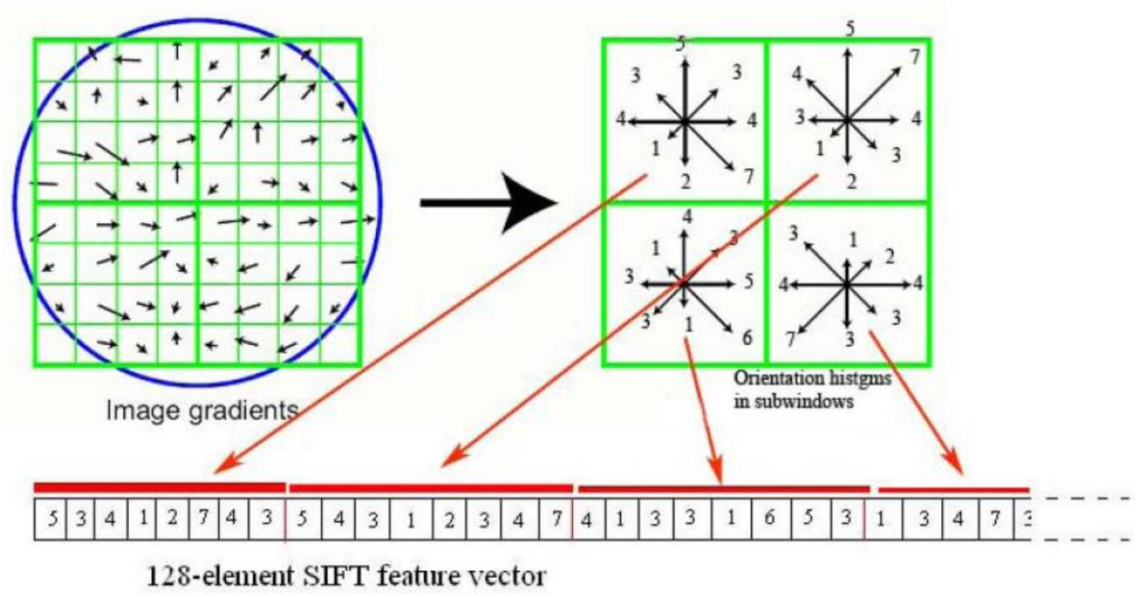
- Next: Scale-invariant Feature Transform (SIFT)

Slide credit: Svetlana Lazebnik, Vittorio Ferrari, Chris Williams

Scale Invariant Feature Transform (SIFT)

Scale Invariant Feature Transform (SIFT)

- Divide patch into 4×4 sub-patches: 16 cells
- Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch (Note: From normalized orientation image).
- Resulting descriptor: 4×4×8=128 dimensions

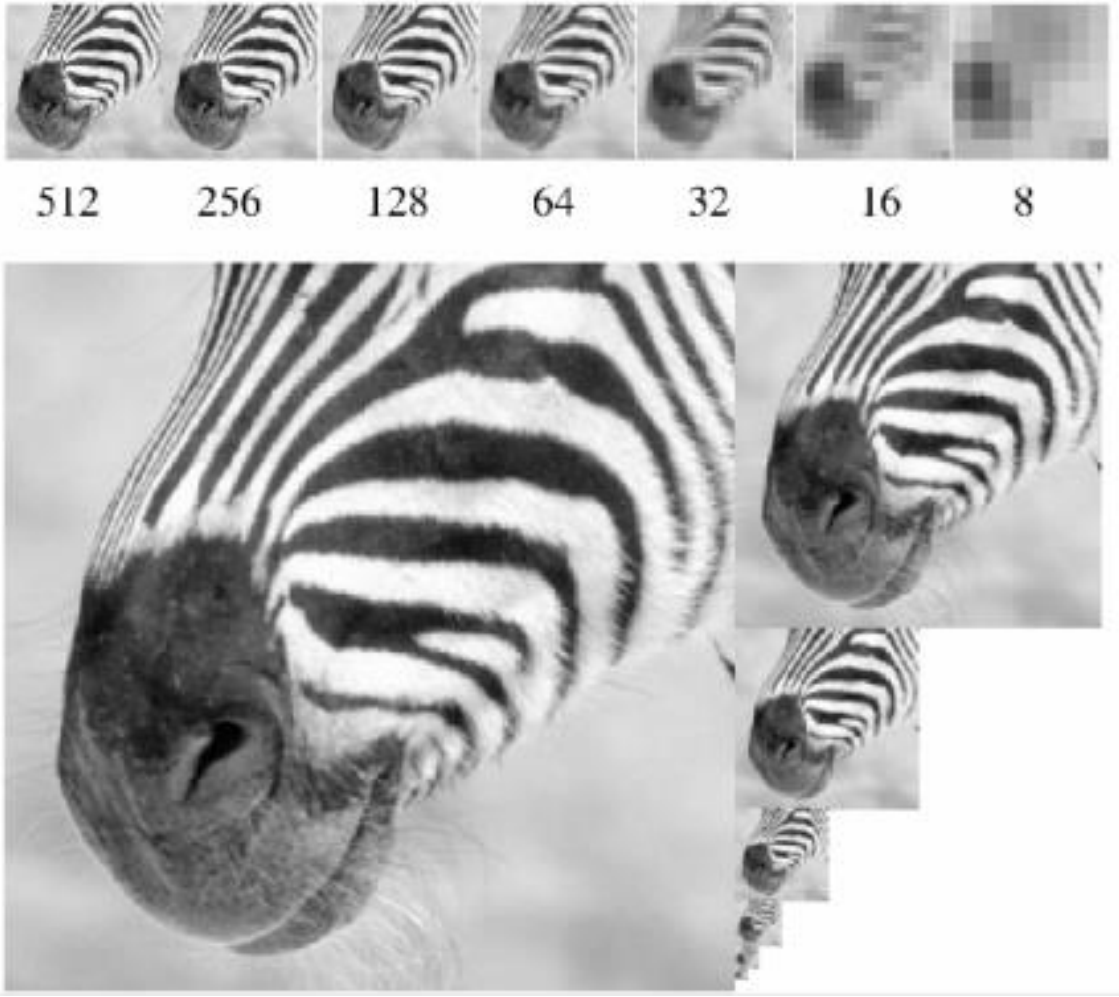


Some Important Concepts

Scale Space

gambar kalo scalenya diubah, bentuk yang ada digambar bakal tetep sama. Jadi deskripsi gambar harusnya gak berubah walaupun scalenya berubah.

- Different scales are appropriate for describing different objects in the image, and we may not know the correct scale/size ahead of time.



Some Important Concepts (2)

sift yang ada beda di corner.

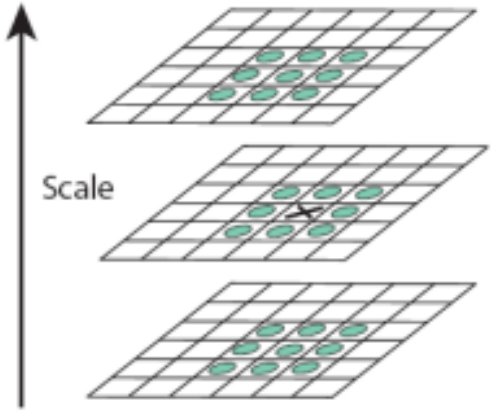
Karena beda dengan teman-teman (scale2 yang lain) membuat SIFT itu scale covariant.

Difference of Gaussians

- subtracting one Gaussian blurred version of image from another, less blurred version

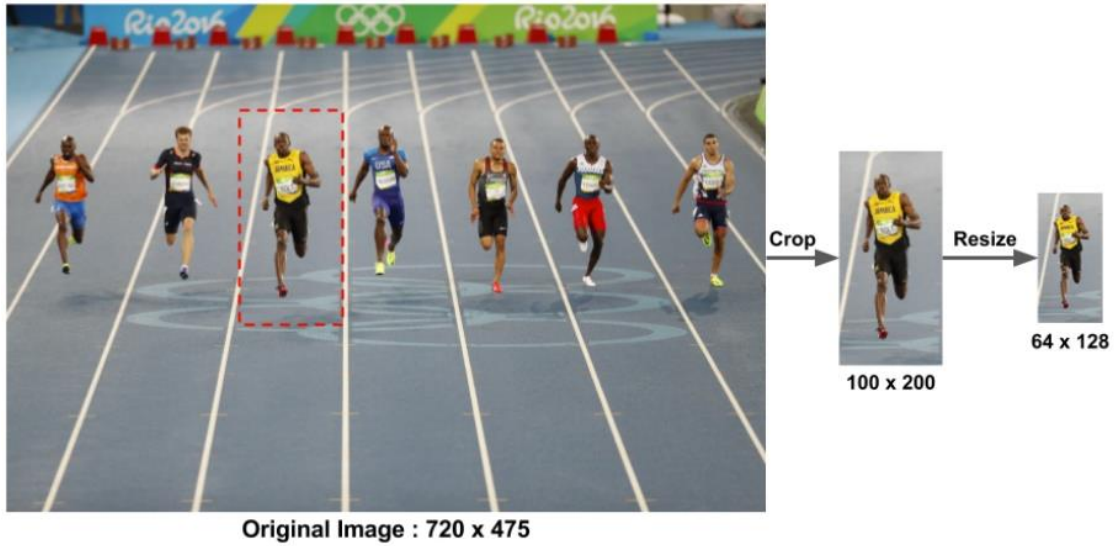
Scale-space Extrema

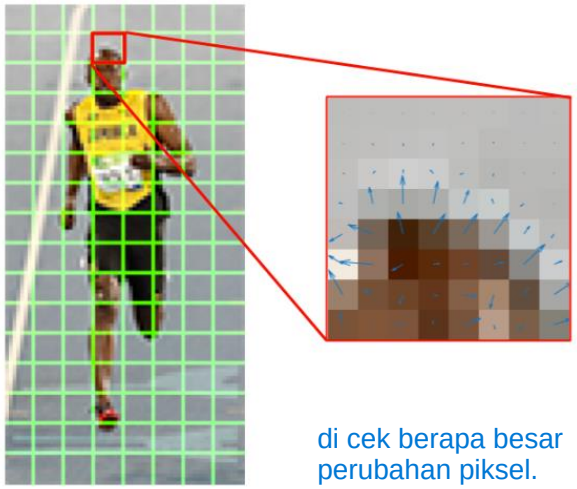
- Find points which are extrema within surrounding 3x3 cube (26 neighbors)



Histogram of Gradients (HOG)

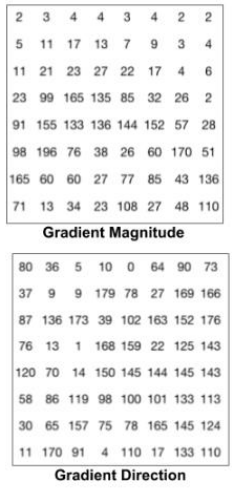
https://learnopencv.com/histogram-of-oriented-gradients/

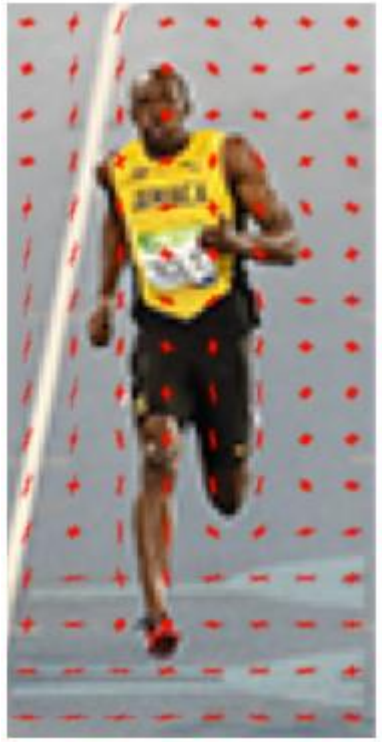




di cek berapa besar perubahan piksel.

Dari nilai-nilai yang ada, kita masuk ke histogram of gradient.

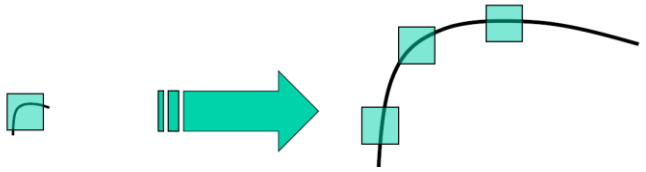




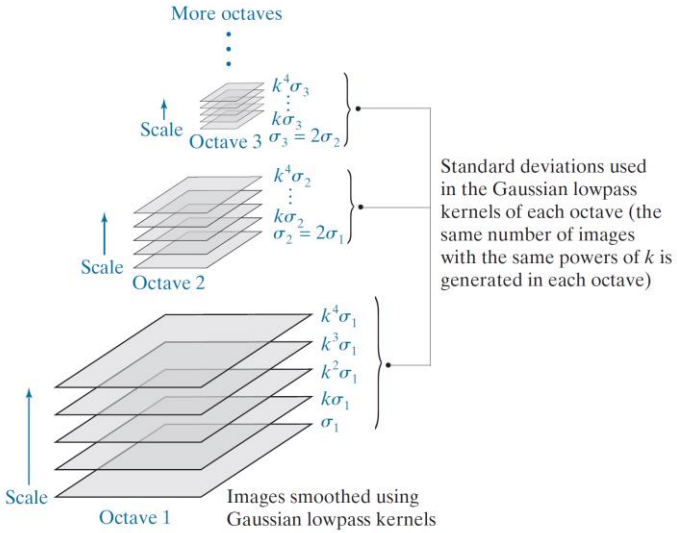
ini hasil ilustrasi dari histogram of gradien

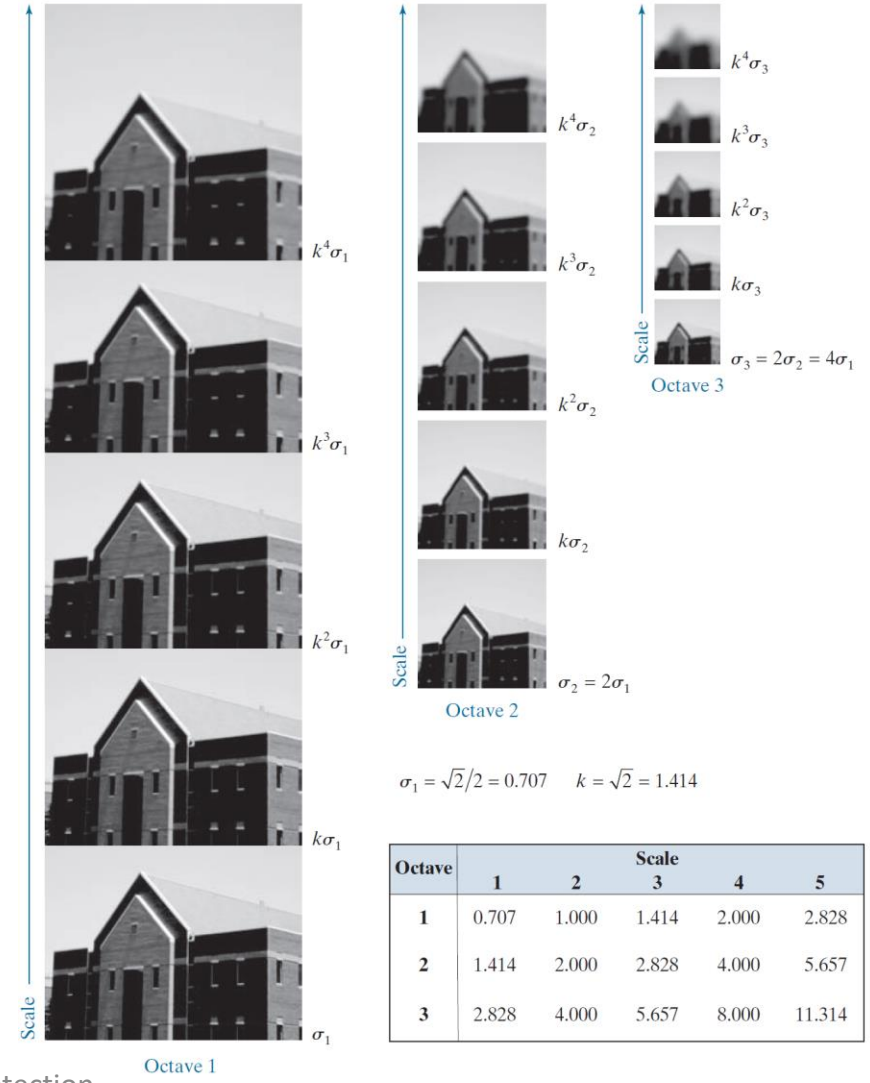
SIFT Algorithm (1/5) – Scale Space

- **Recall:** Scale invariant criteria



- **Solution used in SIFT:**
  - Gaussian kernels (for smoothing) and sub-samplings



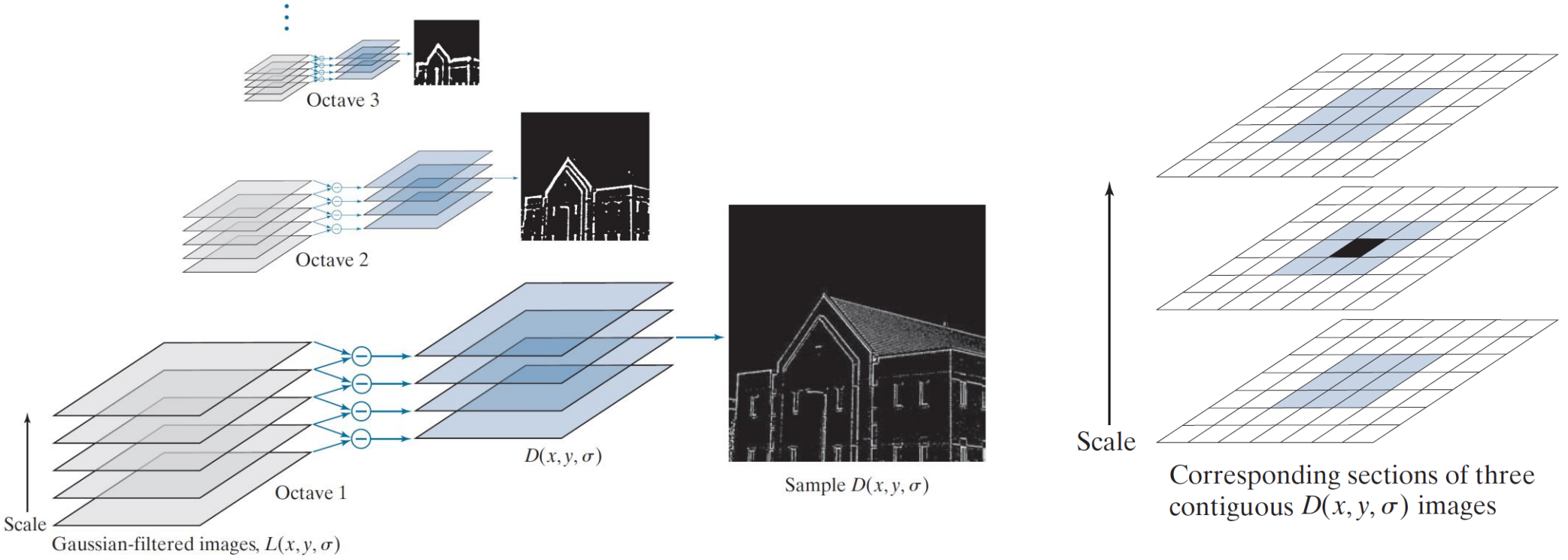


$\sigma_1 = \sqrt{2}/2 = 0.707 \quad k = \sqrt{2} = 1.414$

Octave	Scale 1	2	3	4	5
1	0.707	1.000	1.414	2.000	2.828
2	1.414	2.000	2.828	4.000	5.657
3	2.828	4.000	5.657	8.000	11.314

SIFT Algorithm (2/5) – Initial Keypoints

- Create initial keypoint locations by computing difference of Gaussians (DoG) $D(x, y, \sigma)$ (left figure) and find the extrema in each $D(x, y, \sigma)$ (right figure).



SIFT Algorithm (3/5) – Improve Keypoints

- **Improve keypoints**
  - Via a Taylor expansion (independent reading)
- **Delete unsuitable keypoints**
  - Eliminate keypoints that have low contrast and/or poorly localized
- The cleaned keypoints are scale independence as produced from multiple scales

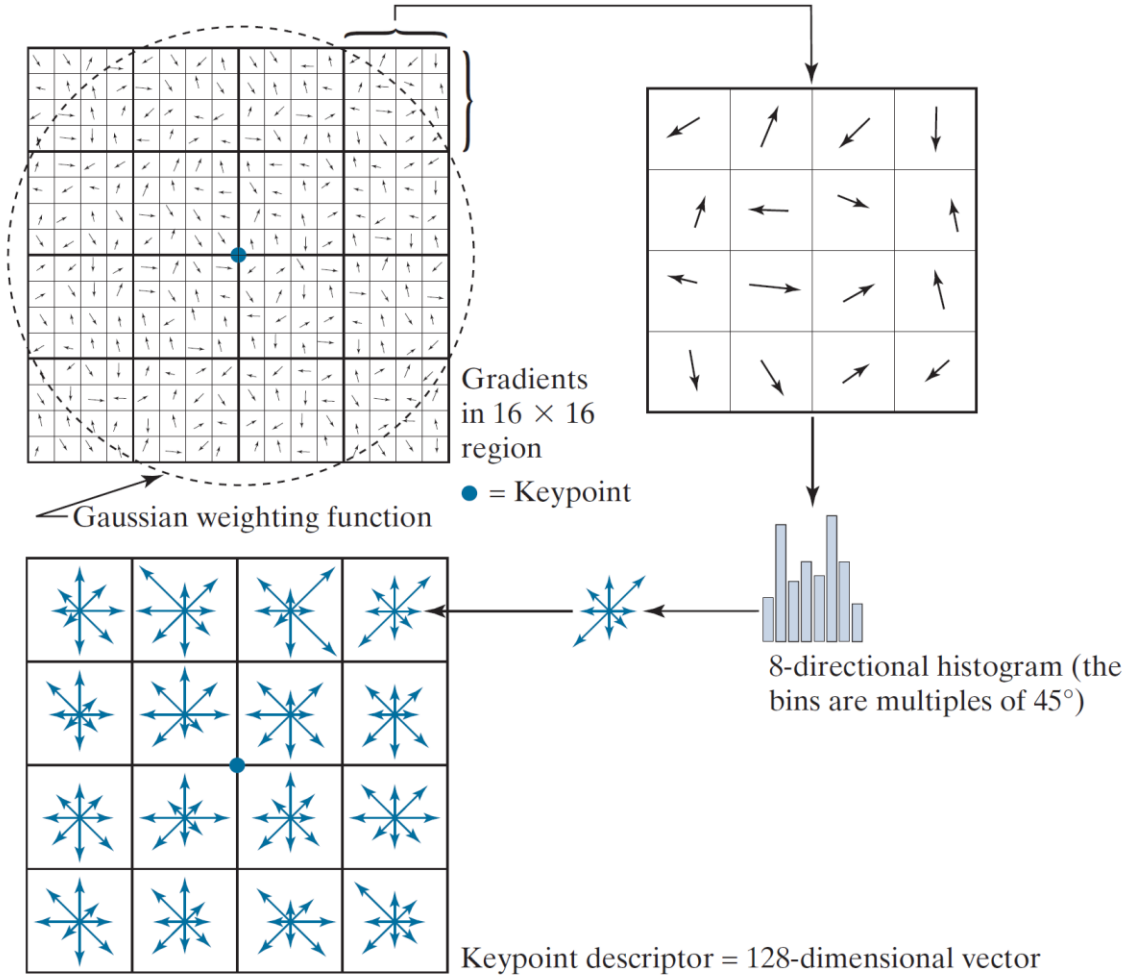
SIFT Algorithm (4/5) – Magnitudes and Orientations of Keypoints

- Magnitude and orientations of keypoints are computed based on the keypoints’s scale space.
  - In this way, all orientation computations are performed in a *scale-invariant manner*.

SIFT Algorithm (5/5) – Keypoint Descriptors

- SIFT produces a descriptor with $4 \times 4 \times 8 = 128$ dimensions

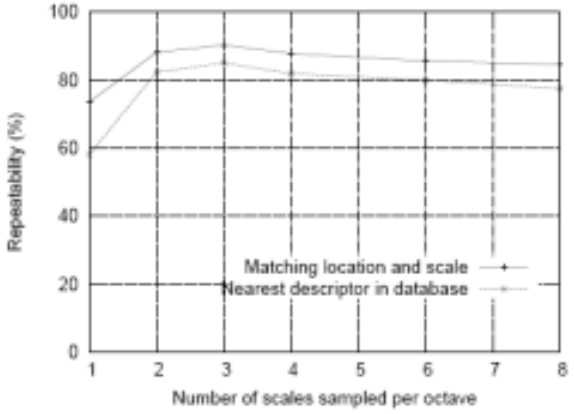
FIGURE 11.62
Approach used to compute a keypoint descriptor.

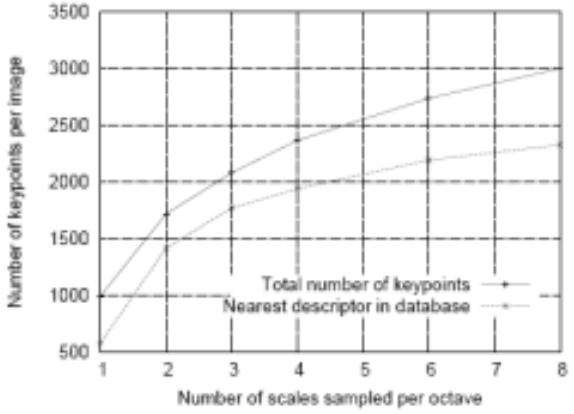


More on SIFT

Why 3 Octaves?

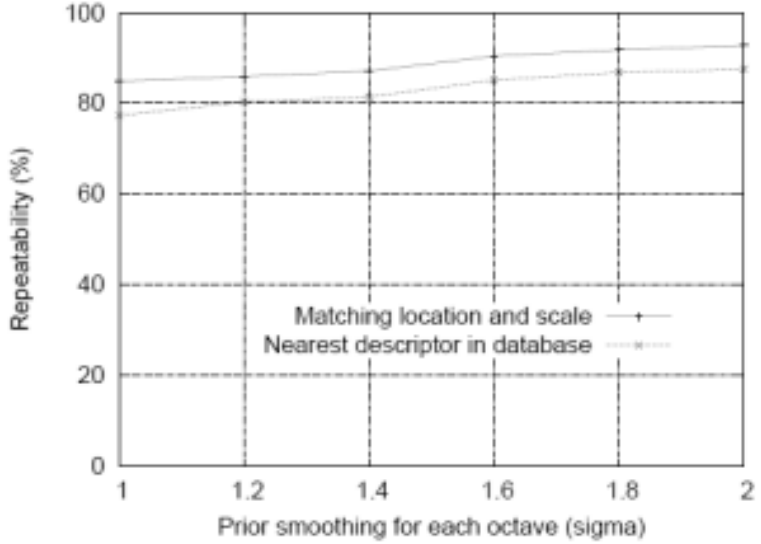
- 3 scales/octave empirically chosen





Gaussian σ

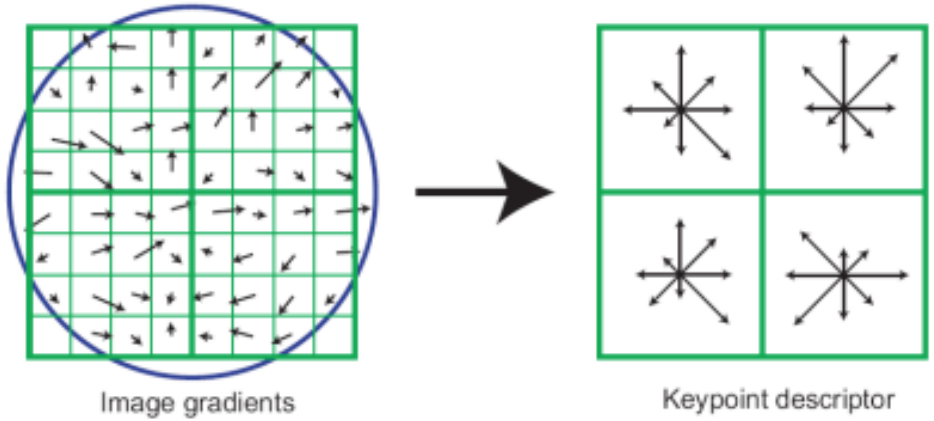
- σ = 1.6 empirically chosen



More on SIFT (2)

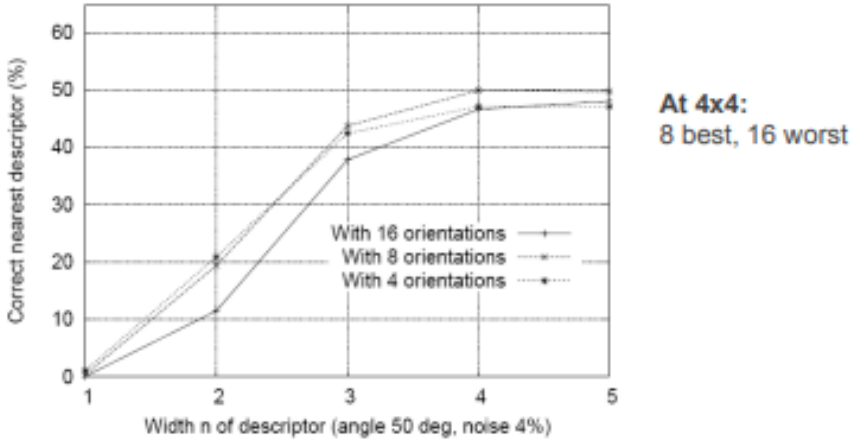
Gradient Histograms

- Sample gradient magnitude orientation (relative to keypoint) in 16x16 window
- Arrange into 4x4 histograms (8 bins)



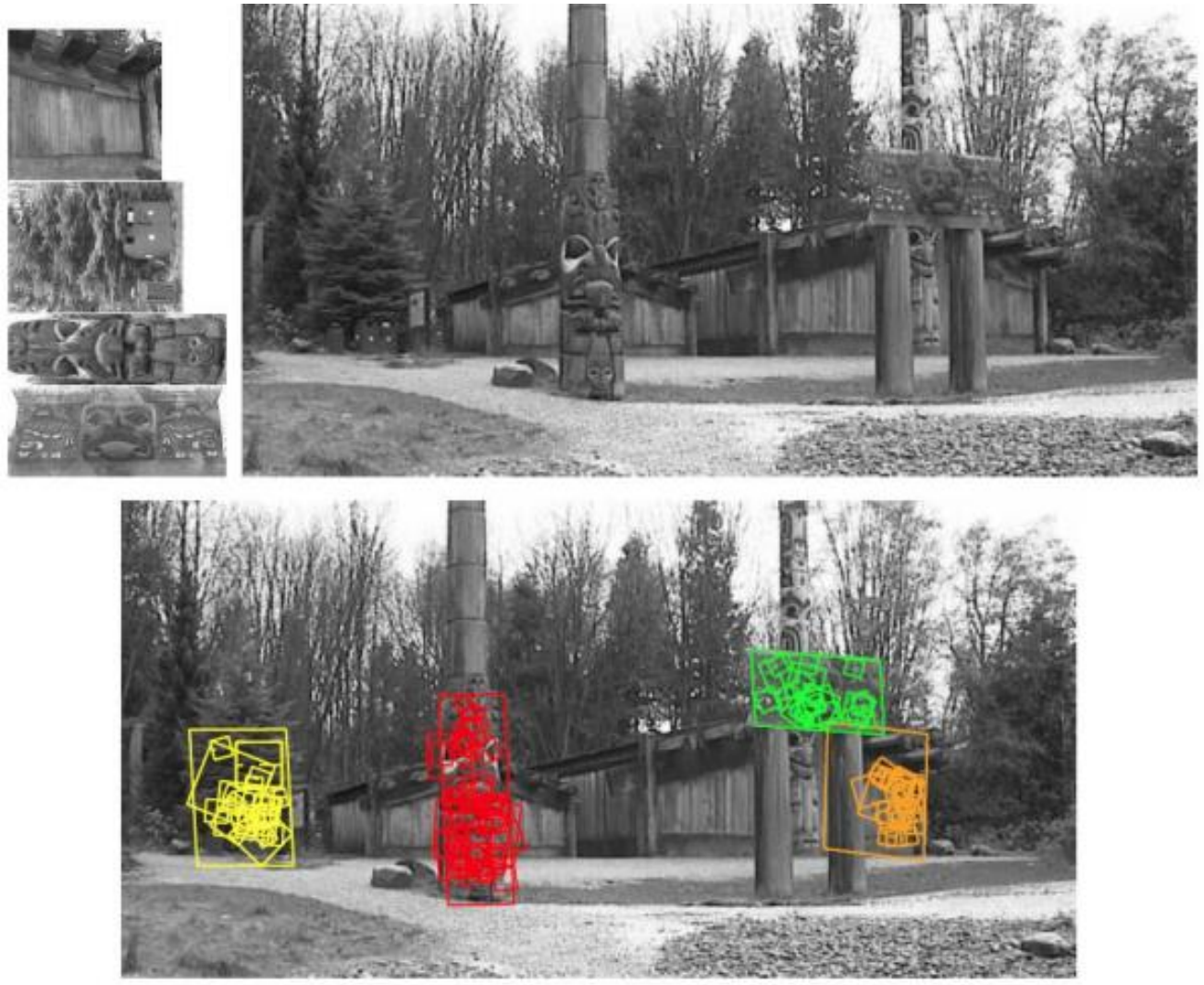
Descriptor Size

- R bins * N^2 sample grid results in R*N 2 vector
- Used 4x4 grid, 8 orientation bins: 128 element vector



SIFT is Robust - Occlusion

SIFT is Robust – Complex Scenes



Summary: SIFT Advantages

- Extraordinarily robust matching technique:
  - Can handle changes in viewpoint up to ~60 deg. out-of-plane rotation
  - Can handle significant changes in illumination
- Fast and efficient - can run in real time
- Lots of code available
  http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT

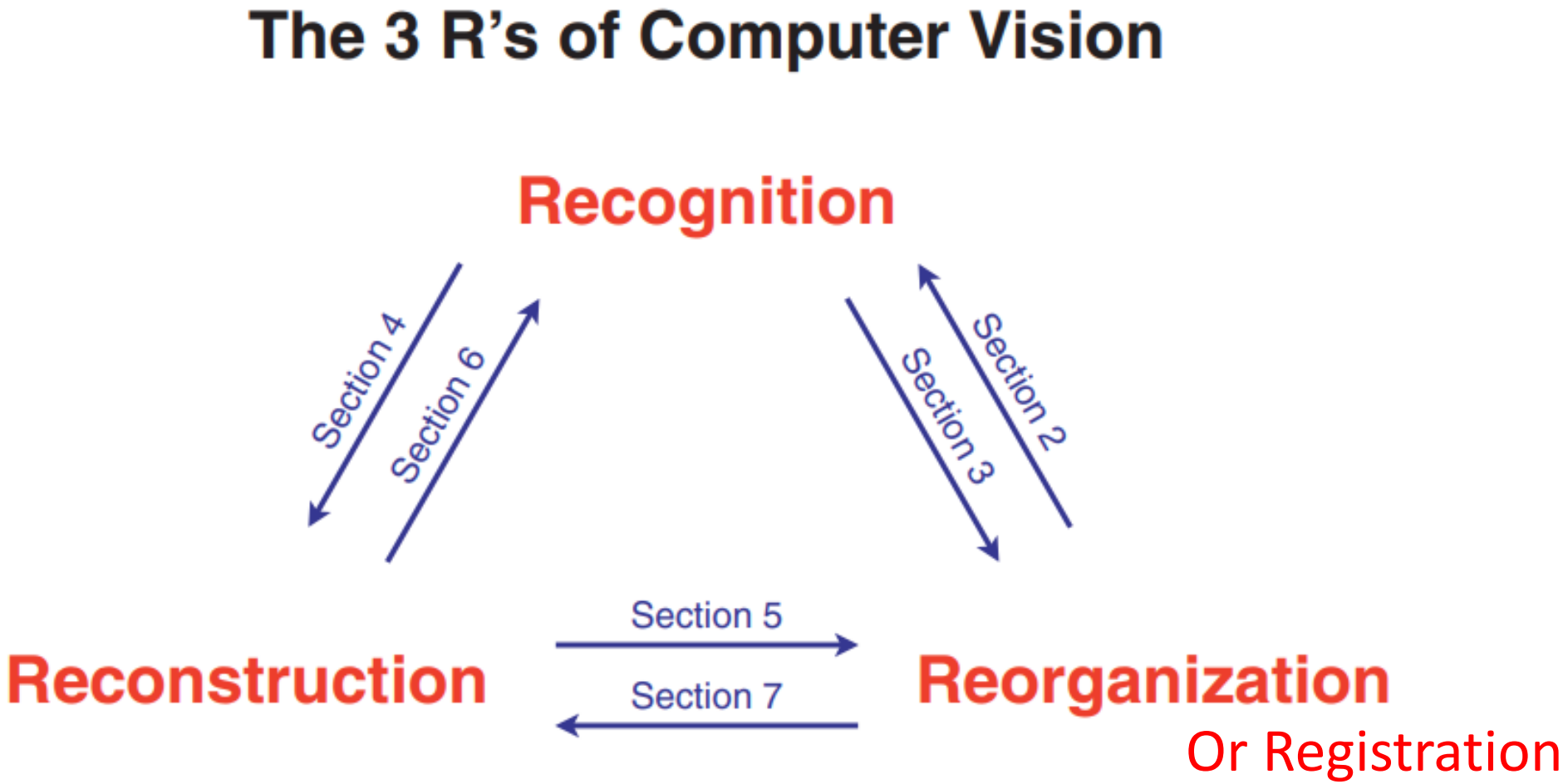
Other Invariant Features

- SURF (Speeded-Up Robust Features)
  - https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html
- FAST (Features from Accelerated Segment Test)
  - https://docs.opencv.org/master/df/d0c/tutorial_py_fast.html
- KAZE Features
  - https://www.doc.ic.ac.uk/~ajd/Publications/alcantarilla_etal_eccv2012.pdf
  - Tutorial
- BRIEF (Binary Robust Independent Elementary Features)
  - https://www.cs.ubc.ca/~lowe/525/papers/calonder_eccv10.pdf
- And many more…

Applications

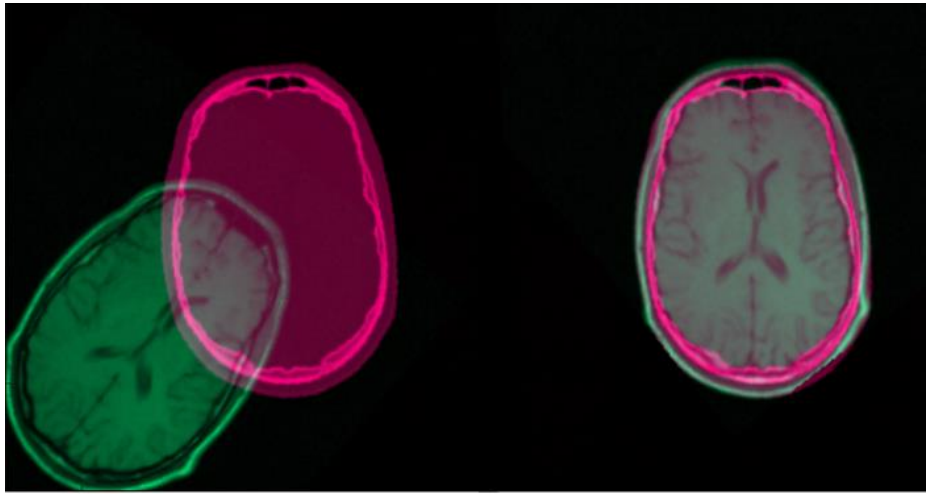
The 3 R’s of Computer Vision

Malik, Jitendra, et al. "The three R’s of computer vision: Recognition, reconstruction and reorganization." *Pattern Recognition Letters* 72 (2016): 4-14.



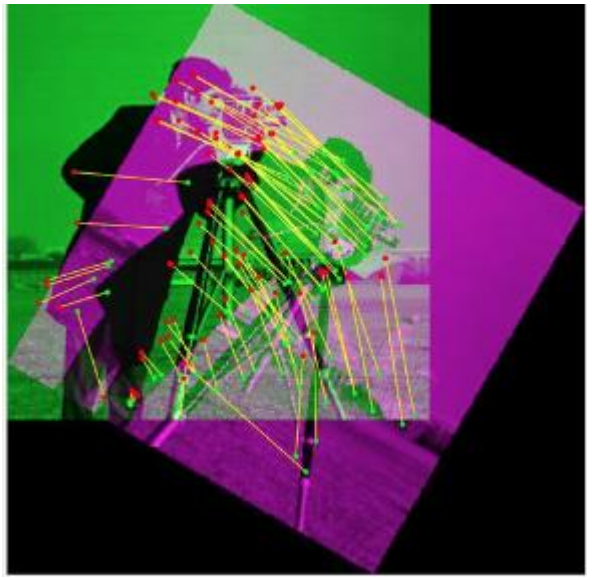
SIFT for Image Registration

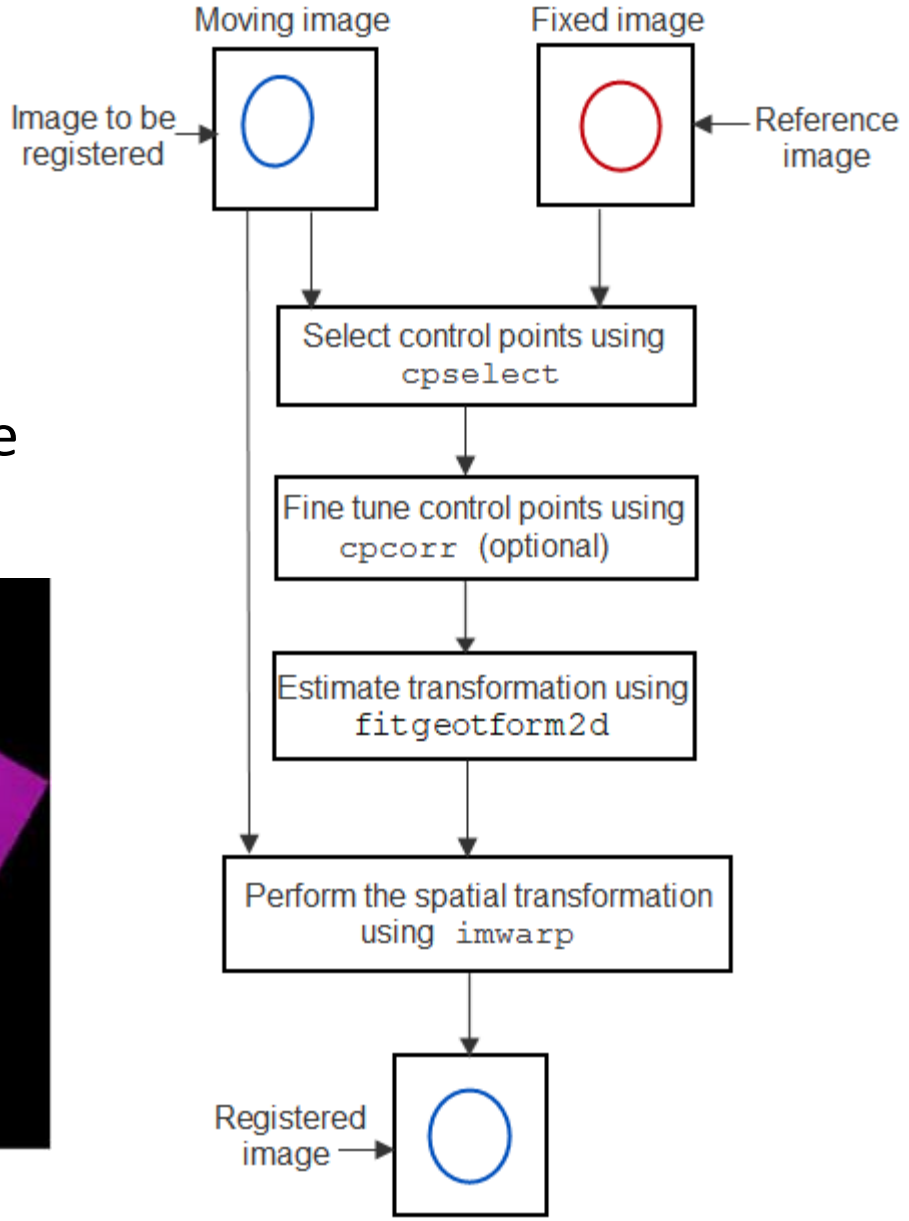
- Image registration is the process of mapping different images of one scene into the same coordinate system.
- These images can be taken at different times (multi-temporal registration), by different sensors (multi-modal registration), and/or from different viewpoints.



SIFT for Image Registration (2)

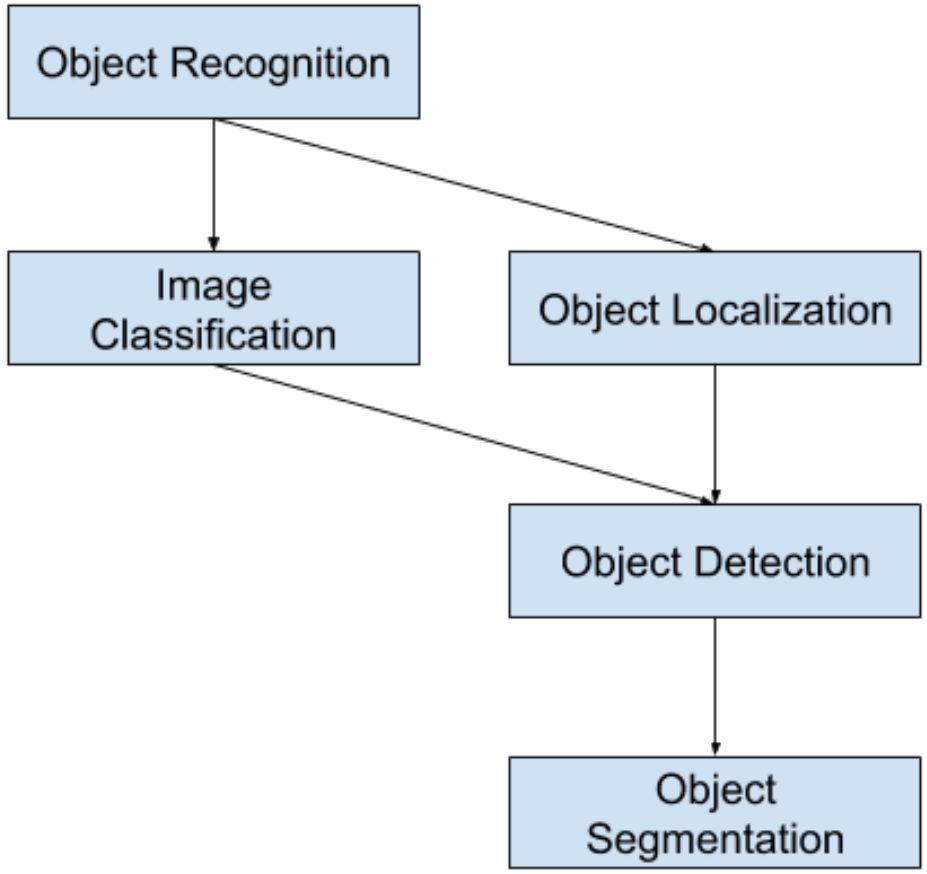
- Point-Mapping Method: registering 2 images with unknown misalignment – with 1 image as the reference (fixed image), the other the moving image
- Steps:
  - Detect and match keypoints
  - Estimate transformation T between points (Recall: RANSAC)
  - Transform the moving image according to T





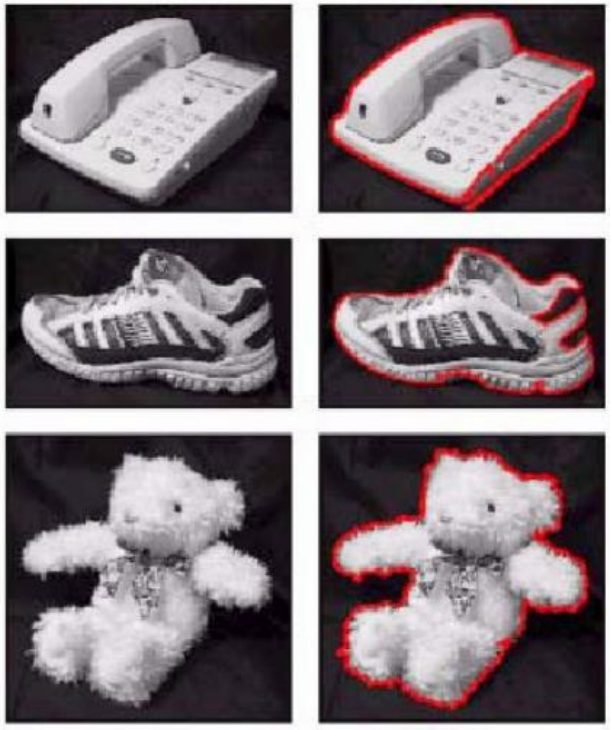
SIFT for Object Detection

- Object detection is a subset of object recognition, where the object is not only identified but also located in an image



SIFT for Object Detection (2)

- Compute SIFT points of models and store in a database



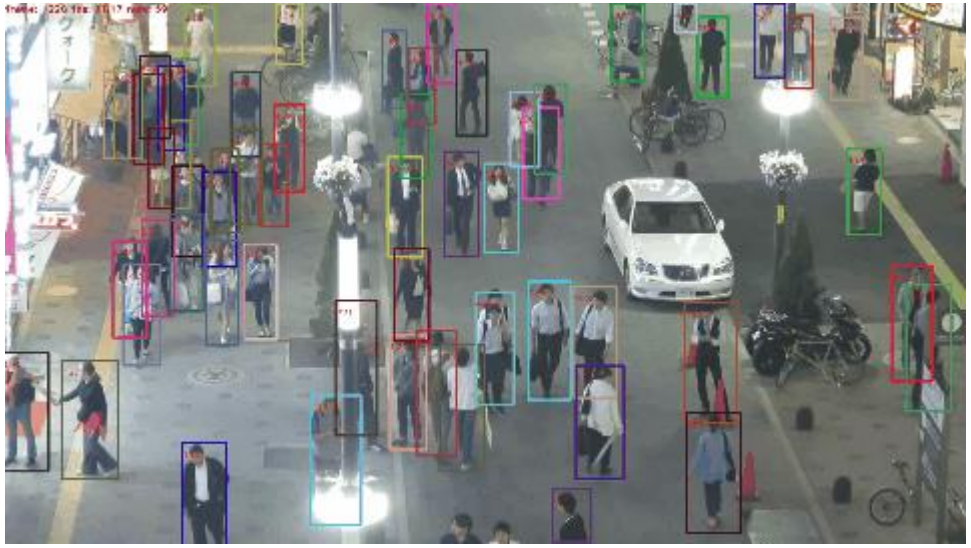
kita bisa buat database of image models

- We can find the matches from the database in images

SIFT for Object Tracking

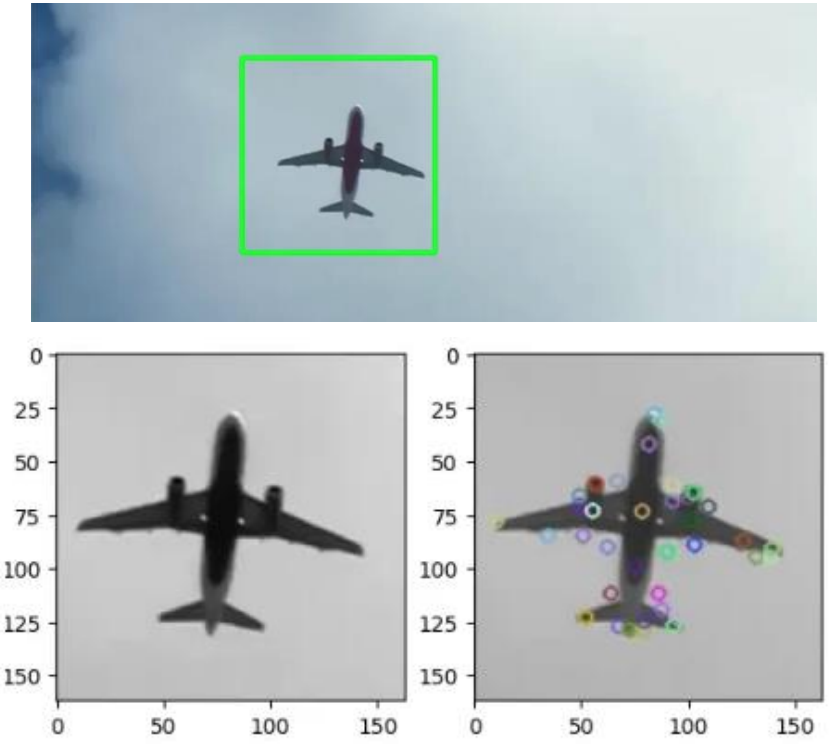
sift bagus kalo single object tracking. Tapi kalo banyak variant, SIFT itu sebenarnya agak kurang.

- After detection, the object is then tracked over space and time
  - Single Object Tracking
  - Multiple Object Tracking



SIFT for Object Tracking (2)

- Single Object: Create a target – and compute the SIFT points

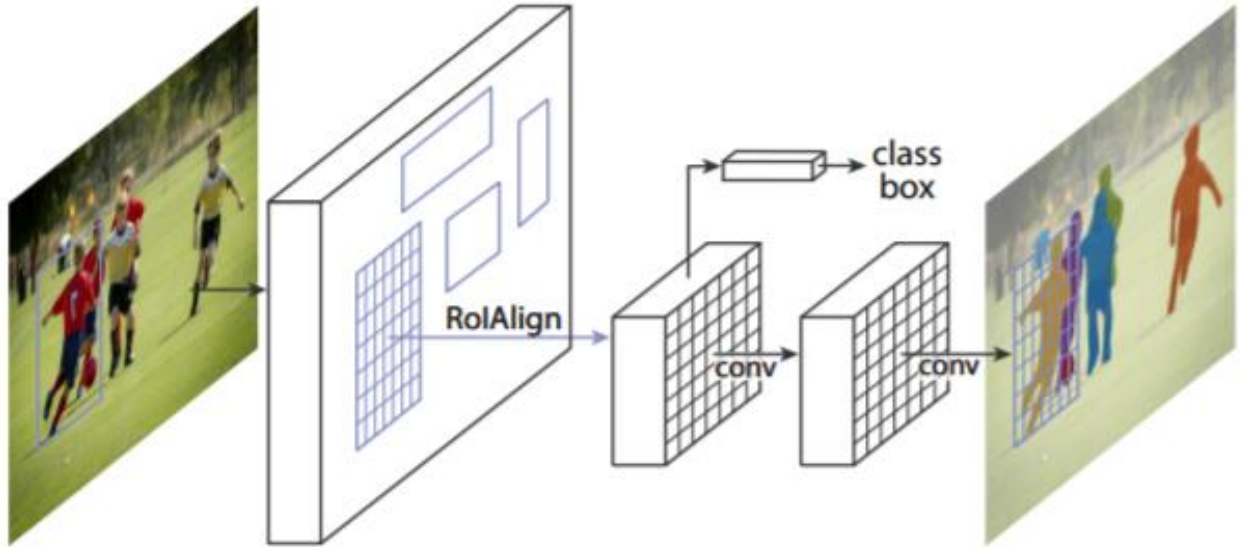


- Track the points in each frame

Multiple Object Detection and Tracking

- Corner and SIFT features are good for *one-to-one* object detection.
- However, they are not enough for object(s) with high variance (*one-to-many*).

Multiple Object Detection and Tracking (2)



- Deep learning models have been developed to model the detection and recognition of multiple objects in images – stay tuned!

Stay tuned for deep networks topic!