

Deep Learning for Computer Vision

# Feature Detectors: SIFT and Variants

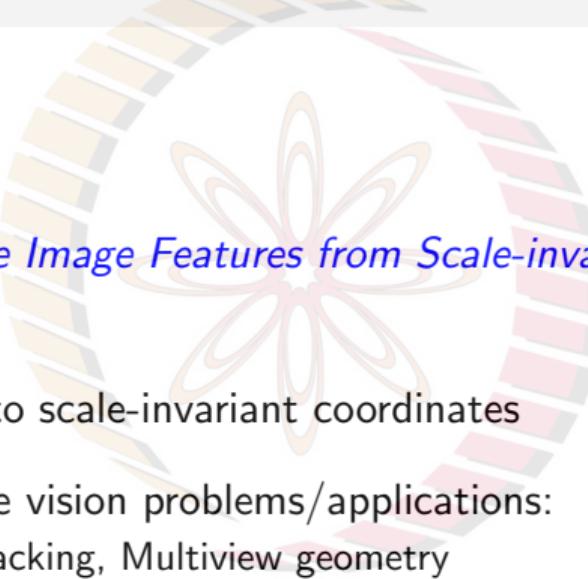
Vineeth N Balasubramanian

Department of Computer Science and Engineering  
Indian Institute of Technology, Hyderabad



NPTEL

# SIFT: Scale Invariant Feature Transform

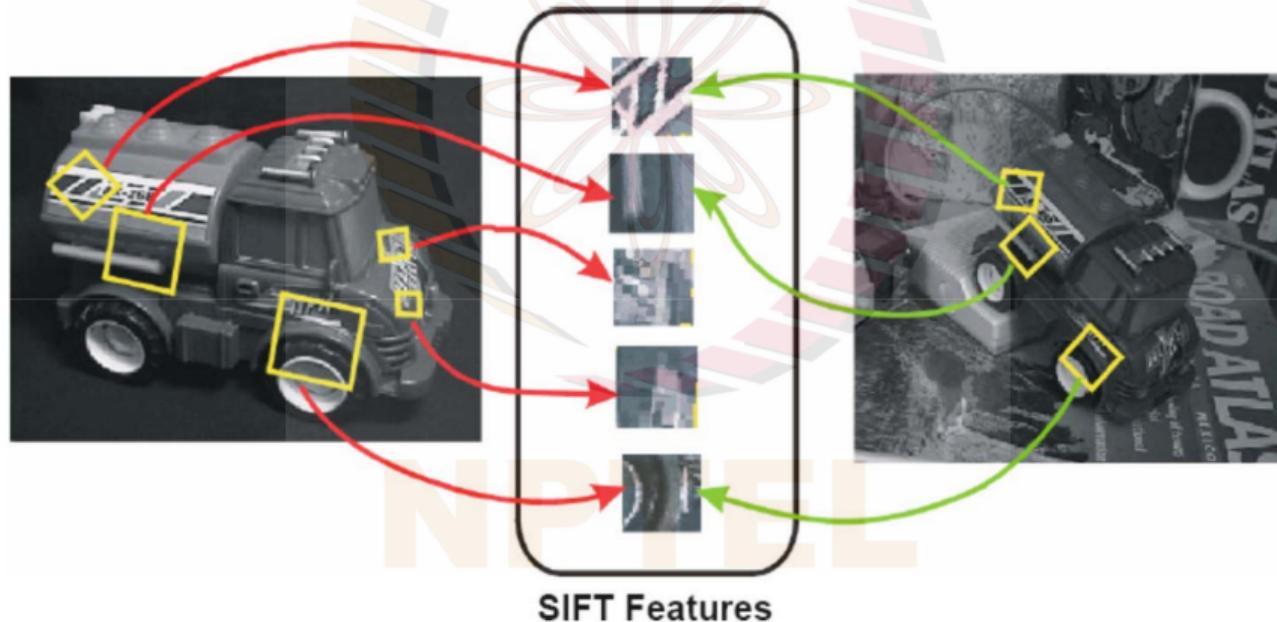


- David G. Lowe, *Distinctive Image Features from Scale-invariant Keypoints*, IJCV 2004
  - Over 50000 citations
- Transforms image data into scale-invariant coordinates
- Fundamental to many core vision problems/applications:
  - Recognition, Motion tracking, Multiview geometry

NPTEL

# SIFT: Invariant Local Features

Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, shear.

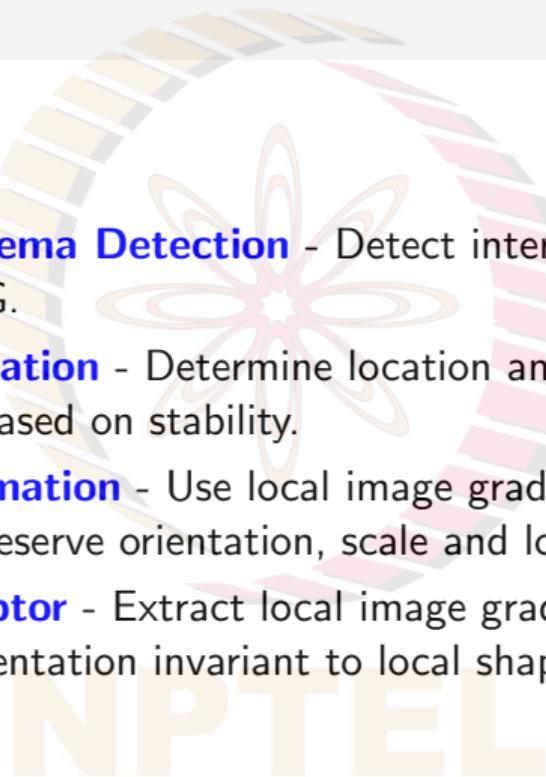


Credit: Mubarak Shah, University of Central Florida

Vineeth N B (IIT-H)

§2.4 Feature Detectors

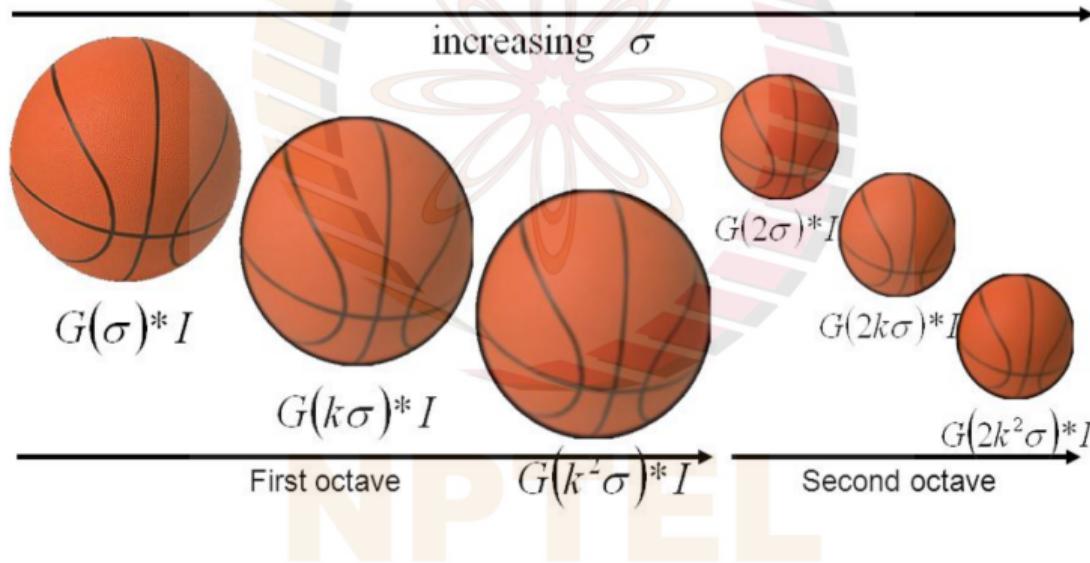
3 / 28



- **Step 1: Scale-space Extrema Detection** - Detect interesting points (invariant to scale and orientation) using DOG.
- **Step 2: Keypoint Localization** - Determine location and scale at each candidate location, and select them based on stability.
- **Step 3: Orientation Estimation** - Use local image gradients to assign orientation to each localized keypoint. Preserve orientation, scale and location for each feature.
- **Step 4: Keypoint Descriptor** - Extract local image gradients at selected scale around keypoint and form a representation invariant to local shape and illumination distortion.

# SIFT: Scale-space Extrema Detection

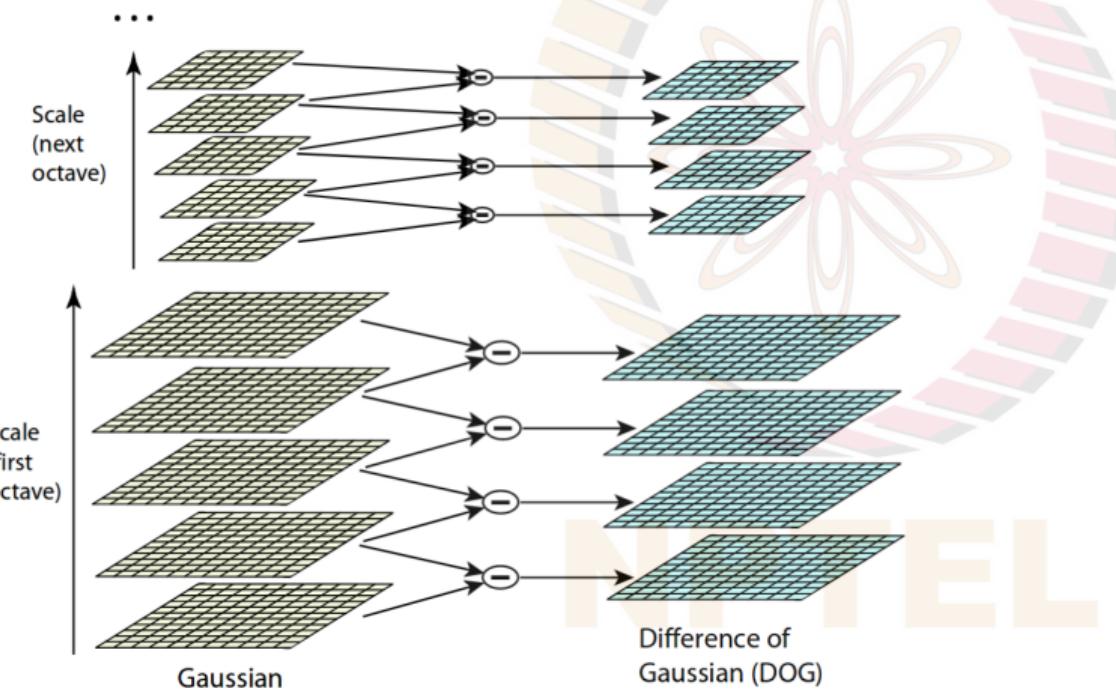
## Constructing Scale Space



Credit: Ofir Pele

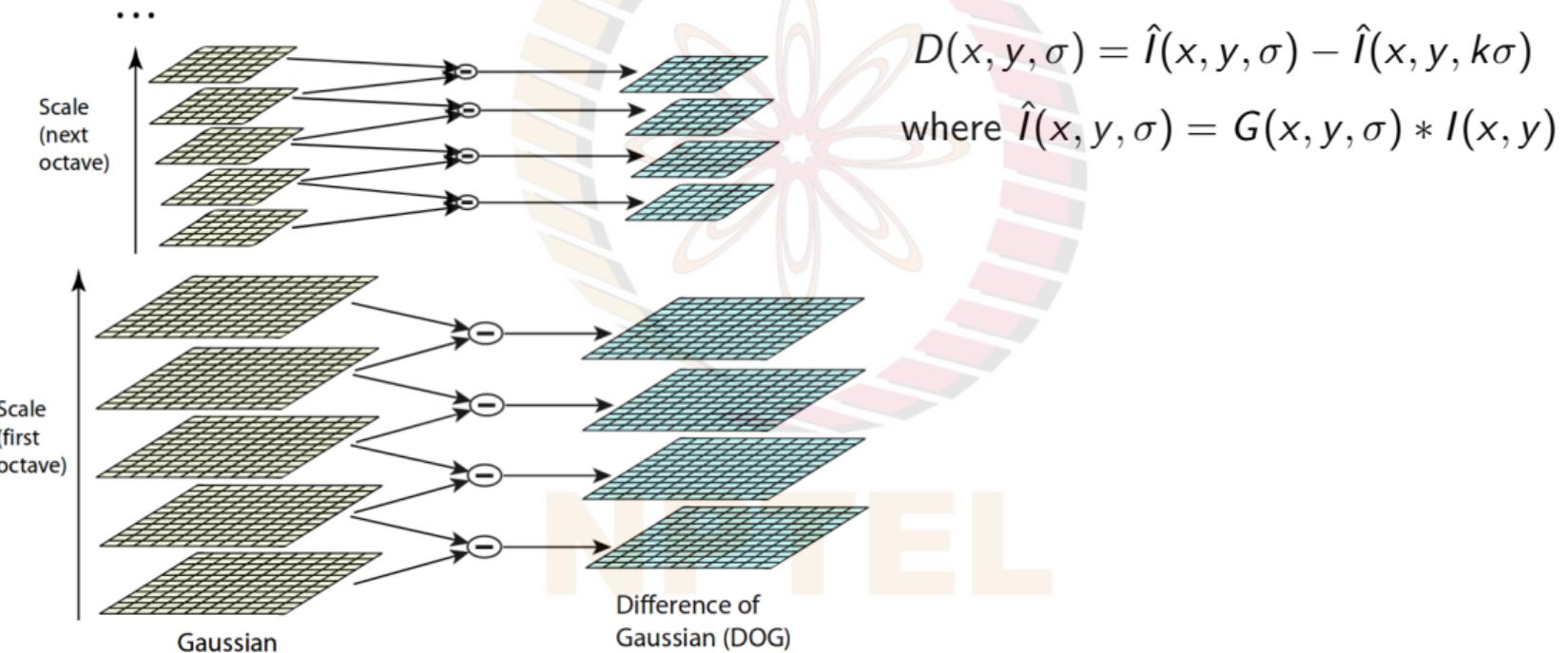
# SIFT: Scale-space Extrema Detection

## Difference of Gaussians



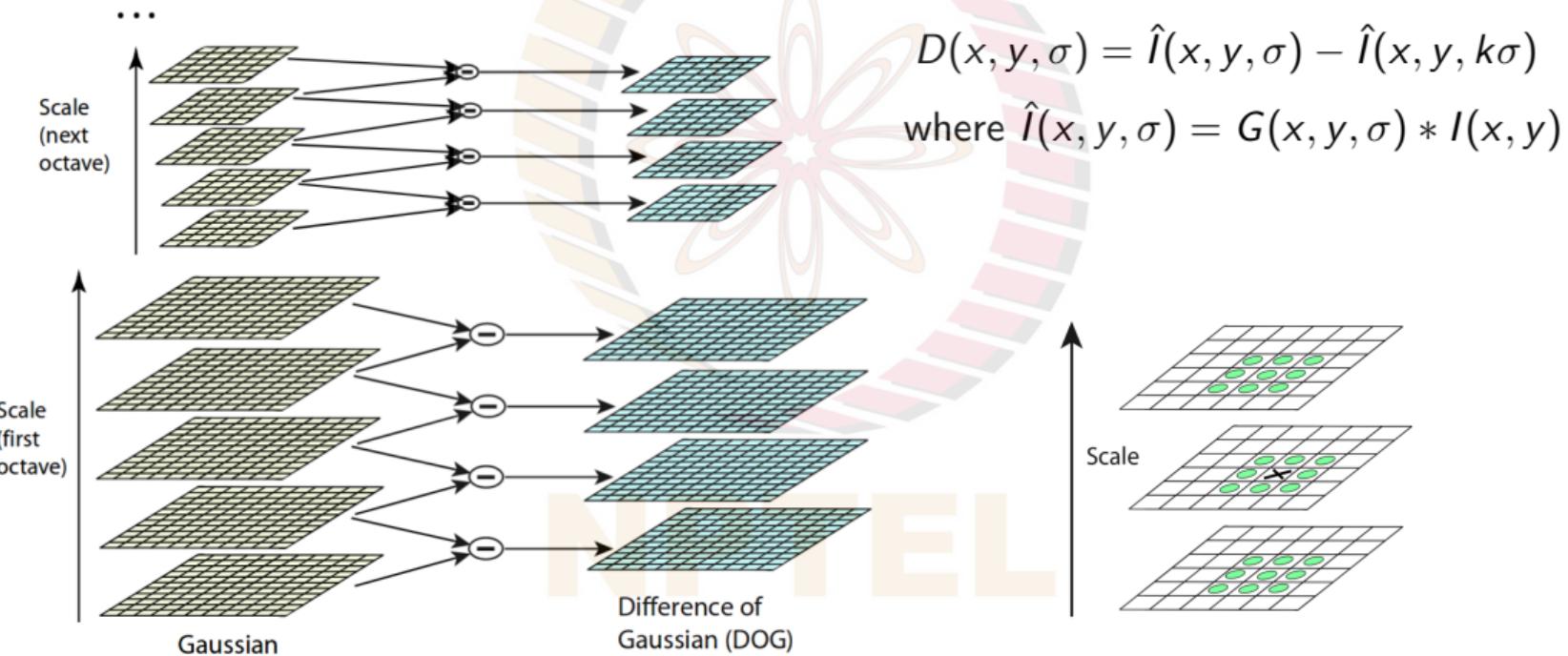
# SIFT: Scale-space Extrema Detection

## Difference of Gaussians



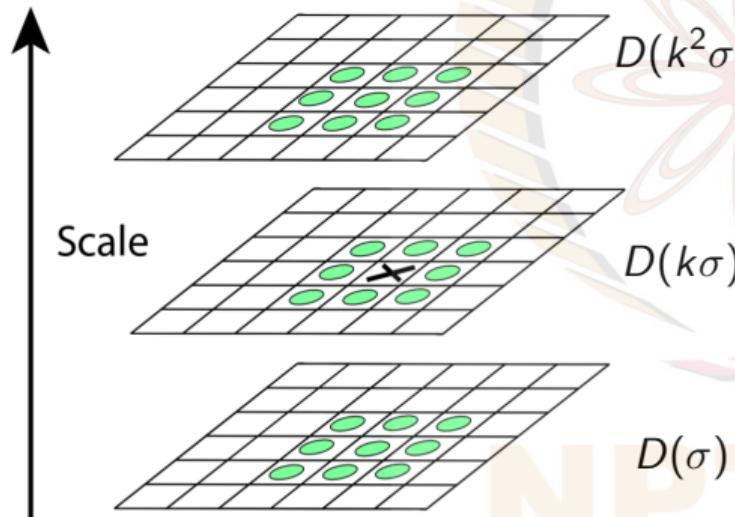
# SIFT: Scale-space Extrema Detection

Difference of Gaussians



Credit: "Distinctive Image Features from Scale-Invariant Points", IJCV 2004

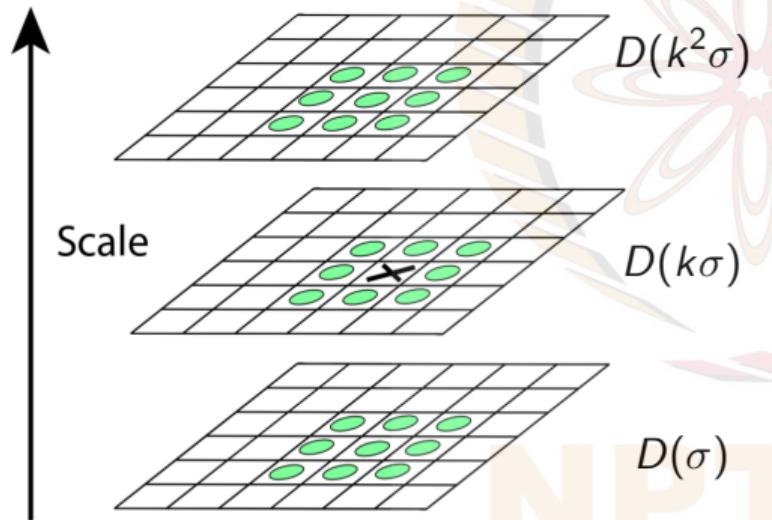
# SIFT: Scale-space Extrema Detection



- Compare a pixel (X) with 26 pixels in current and adjacent scales (Green Circles)
- Select a pixel (X) if it is larger/smaller than all 26 pixels

Credit: "Distinctive Image Features from Scale-Invariant Points", IJCV 2004

# SIFT: Scale-space Extrema Detection



- Compare a pixel (X) with 26 pixels in current and adjacent scales (Green Circles)
- Select a pixel (X) if it is larger/smaller than all 26 pixels

Credit: "Distinctive Image Features from Scale-Invariant Points", IJCV 2004

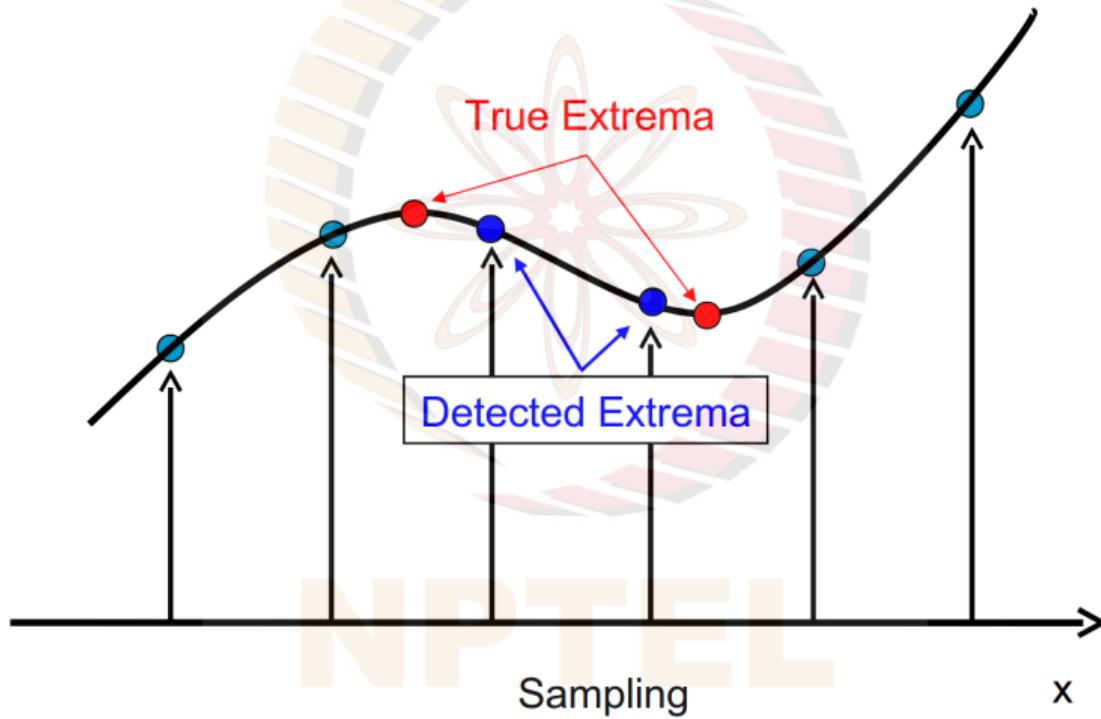
# SIFT Algorithm Stages

- **Step 1: Scale-space extrema Detection** - Detect interesting points (invariant to scale and orientation) using DOG.
- **Step 2: Keypoint Localization** - Determine location and scale at each candidate location, and select them based on stability.
- **Step 3: Orientation Estimation** - Use local image gradients to assign orientation to each localized keypoint. Preserve orientation, scale and location for each feature.
- **Step 4: Keypoint Descriptor** - Extract local image gradients at selected scale around keypoint and form a representation invariant to local shape and illumination distortion them.

NPTEL

# SIFT: Keypoint Localization

The Problem:



Credit: Ofir Pele

# SIFT: Keypoint Localization

The Solution:

- Use Taylor series expansion of the scale-space function:

$$D(\mathbf{s}_0 + \Delta\mathbf{s}) = D(\mathbf{s}_0) + \frac{\partial D}{\partial \mathbf{s}} \Big|_{\mathbf{s}_0} \Delta\mathbf{s} + \frac{1}{2} \Delta\mathbf{s}^T \frac{\partial^2 D}{\partial \mathbf{s}^2} \Big|_{\mathbf{s}_0} \Delta\mathbf{s}$$

where  $\mathbf{s}_0 = (x_0, y_0, \sigma_0)^T$  and  $\Delta\mathbf{s} = (\delta x, \delta y, \delta \sigma)^T$

NPTEL

# SIFT: Keypoint Localization

## The Solution:

- Use Taylor series expansion of the scale-space function:

$$D(\mathbf{s}_0 + \Delta\mathbf{s}) = D(\mathbf{s}_0) + \frac{\partial D}{\partial \mathbf{s}} \Big|_{\mathbf{s}_0} \Delta\mathbf{s} + \frac{1}{2} \Delta\mathbf{s}^T \frac{\partial^2 D}{\partial \mathbf{s}^2} \Big|_{\mathbf{s}_0} \Delta\mathbf{s}$$

where  $\mathbf{s}_0 = (x_0, y_0, \sigma_0)^T$  and  $\Delta\mathbf{s} = (\delta x, \delta y, \delta \sigma)^T$

- The location of the extremum,  $\hat{\mathbf{s}}$ , is determined by taking the derivative of this function with respect to  $\mathbf{s}$  and setting it to zero:

$$\hat{\mathbf{s}} = - \left( \frac{\partial^2 D}{\partial \mathbf{s}^2} \Big|_{\mathbf{s}_0} \right)^{-1} \frac{\partial D}{\partial \mathbf{s}} \Big|_{\mathbf{s}_0}$$

# SIFT: Keypoint Localization

## The Solution:

- Use Taylor series expansion of the scale-space function:

$$D(\mathbf{s}_0 + \Delta\mathbf{s}) = D(\mathbf{s}_0) + \frac{\partial D}{\partial \mathbf{s}} \Big|_{\mathbf{s}_0} \Delta\mathbf{s} + \frac{1}{2} \Delta\mathbf{s}^T \frac{\partial^2 D}{\partial \mathbf{s}^2} \Big|_{\mathbf{s}_0} \Delta\mathbf{s}$$

where  $\mathbf{s}_0 = (x_0, y_0, \sigma_0)^T$  and  $\Delta\mathbf{s} = (\delta x, \delta y, \delta \sigma)^T$

- The location of the extremum,  $\hat{\mathbf{s}}$ , is determined by taking the derivative of this function with respect to  $\mathbf{s}$  and setting it to zero:

$$\hat{\mathbf{s}} = - \left( \frac{\partial^2 D}{\partial \mathbf{s}^2} \Big|_{\mathbf{s}_0} \right)^{-1} \frac{\partial D}{\partial \mathbf{s}} \Big|_{\mathbf{s}_0}$$

- Next, reject low contrast points and points that lie on the edge

# SIFT: Keypoint Localization

## The Solution:

- Use Taylor series expansion of the scale-space function:

$$D(\mathbf{s}_0 + \Delta\mathbf{s}) = D(\mathbf{s}_0) + \frac{\partial D}{\partial \mathbf{s}} \Big|_{\mathbf{s}_0} \Delta\mathbf{s} + \frac{1}{2} \Delta\mathbf{s}^T \frac{\partial^2 D}{\partial \mathbf{s}^2} \Big|_{\mathbf{s}_0} \Delta\mathbf{s}$$

where  $\mathbf{s}_0 = (x_0, y_0, \sigma_0)^T$  and  $\Delta\mathbf{s} = (\delta x, \delta y, \delta \sigma)^T$

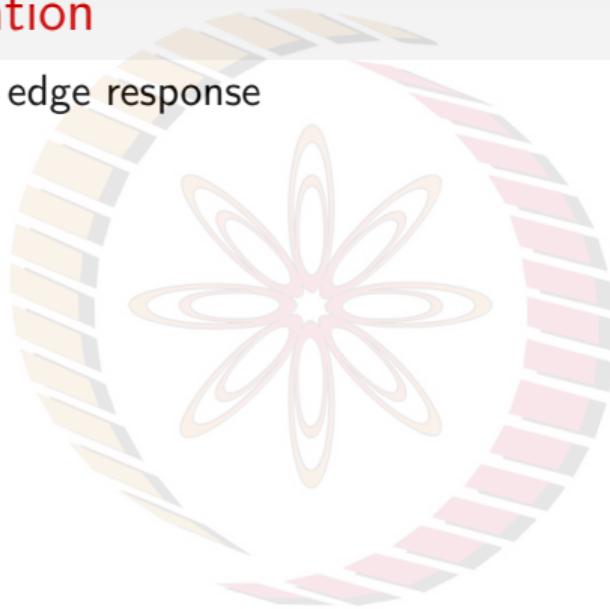
- The location of the extremum,  $\hat{\mathbf{s}}$ , is determined by taking the derivative of this function with respect to  $\mathbf{s}$  and setting it to zero:

$$\hat{\mathbf{s}} = - \left( \frac{\partial^2 D}{\partial \mathbf{s}^2} \Big|_{\mathbf{s}_0} \right)^{-1} \frac{\partial D}{\partial \mathbf{s}} \Big|_{\mathbf{s}_0}$$

- Next, reject low contrast points and points that lie on the edge
- **Low contrast points elimination:**
  - Reject keypoint if  $D(\hat{\mathbf{s}})$  is smaller than 0.03 (assuming image values are normalized in [0,1])

## SIFT: Keypoint Localization

- Reject points with strong edge response in one direction only
- Edge Elimination - How?



NPTEL

# SIFT: Keypoint Localization

- Reject points with strong edge response in one direction only
- Edge Elimination - How?
  - Similar to Harris corner detector!
  - SIFT instead uses Hessian
- Compute Hessian of D (principal curvature)

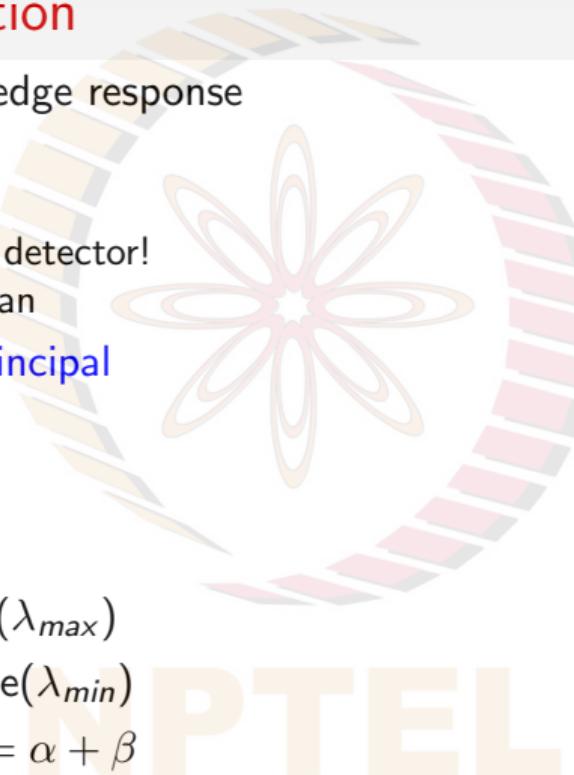
$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$\alpha$  : largest eigenvalue( $\lambda_{max}$ )

$\beta$  : smallest eigenvalue( $\lambda_{min}$ )

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta$$

$$Det(H) = D_{xx}D_{yy} - D_{xy}^2 = \alpha\beta$$



# SIFT: Keypoint Localization

- Reject points with strong edge response in one direction only
- Edge Elimination - How?
  - Similar to Harris corner detector!
  - SIFT instead uses Hessian
- Compute Hessian of D (principal curvature)

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$\alpha$  : largest eigenvalue( $\lambda_{max}$ )

$\beta$  : smallest eigenvalue( $\lambda_{min}$ )

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta$$

$$Det(H) = D_{xx}D_{yy} - D_{xy}^2 = \alpha\beta$$

- Evaluate ratio

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2}$$

$$\frac{Tr(H)^2}{Det(H)} = \frac{(r + 1)^2}{r} \text{ where } r = \frac{\alpha}{\beta}$$

# SIFT: Keypoint Localization

- Reject points with strong edge response in one direction only

- Edge Elimination - How?

- Similar to Harris corner detector!
  - SIFT instead uses Hessian

- Compute Hessian of D (principal curvature)

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$\alpha$  : largest eigenvalue( $\lambda_{max}$ )

$\beta$  : smallest eigenvalue( $\lambda_{min}$ )

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta$$

$$Det(H) = D_{xx}D_{yy} - D_{xy}^2 = \alpha\beta$$

- Evaluate ratio

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2}$$

$$\frac{Tr(H)^2}{Det(H)} = \frac{(r + 1)^2}{r} \text{ where } r = \frac{\alpha}{\beta}$$

- This quantity is minimum when  $r = 1$  (eigenvalues are equal)
- Reject keypoint if:  $\frac{Tr(H)^2}{Det(H)} >$  a threshold  
(Original SIFT uses  $r = 10$ )

# SIFT Algorithm Stages

- **Step 1: Scale-space extrema Detection** - Detect interesting points (invariant to scale and orientation) using DOG.
- **Step 2: Keypoint Localization** - Determine location and scale at each candidate location, and select them based on stability.
- **Step 3: Orientation Estimation** - Use local image gradients to assign orientation to each localized keypoint. Preserve orientation, scale and location for each feature.
- **Step 4: Keypoint Descriptor** - Extract local image gradients at selected scale around keypoint and form a representation invariant to local shape and illumination distortion them.

NPTEL

# SIFT: Orientation Estimation

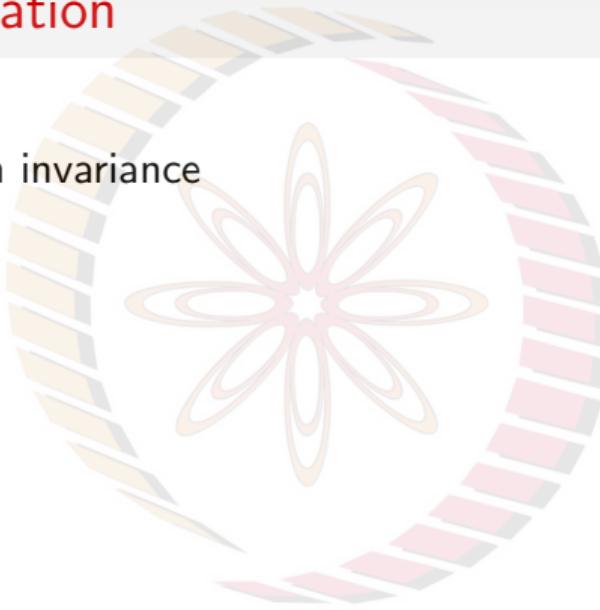
- Why?



NPTEL

## SIFT: Orientation Estimation

- Why? To achieve rotation invariance



NPTEL

## SIFT: Orientation Estimation

- Why? To achieve rotation invariance
- Use scale of point to choose correct image:

$$\hat{I}(x, y) = G(x, y, \sigma) * I(x, y)$$

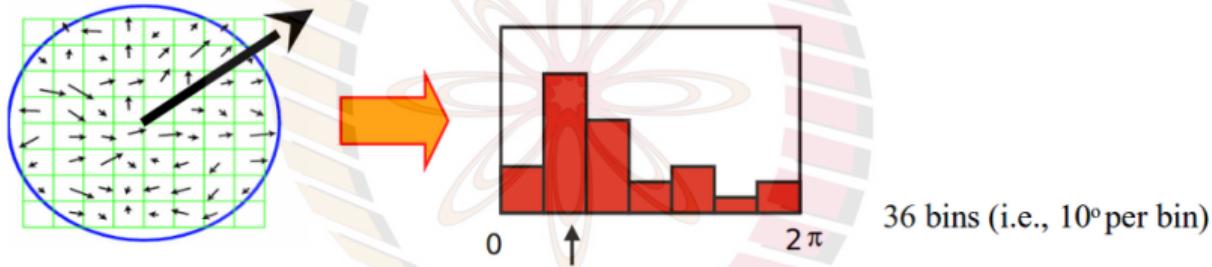
- Compute gradient magnitude and orientation using finite differences:

$$m(x, y) = \sqrt{(\hat{I}(x + 1, y) - \hat{I}(x - 1, y))^2 + (\hat{I}(x, y + 1) - \hat{I}(x, y - 1))^2}$$

$$\theta(x, y) = \tan^{-1} \left( \frac{(\hat{I}(x, y + 1) - \hat{I}(x, y - 1))}{(\hat{I}(x + 1, y) - \hat{I}(x - 1, y))} \right)$$

# SIFT: Orientation Estimation

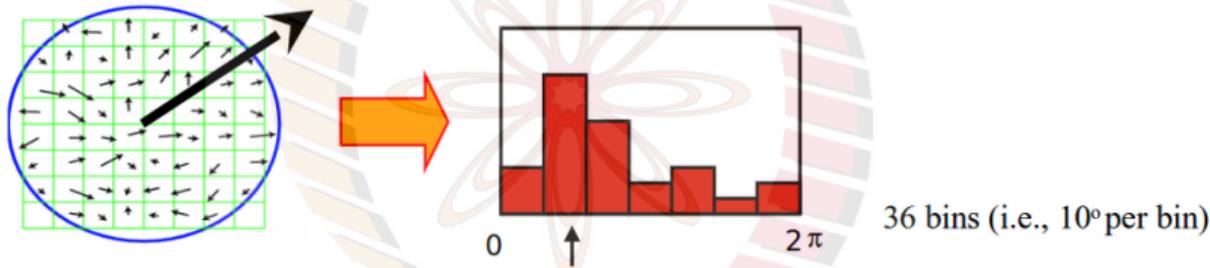
- Create histogram of gradient directions, within a region around the keypoint, at selected scale:



NPTEL

## SIFT: Orientation Estimation

- Create histogram of gradient directions, within a region around the keypoint, at selected scale:

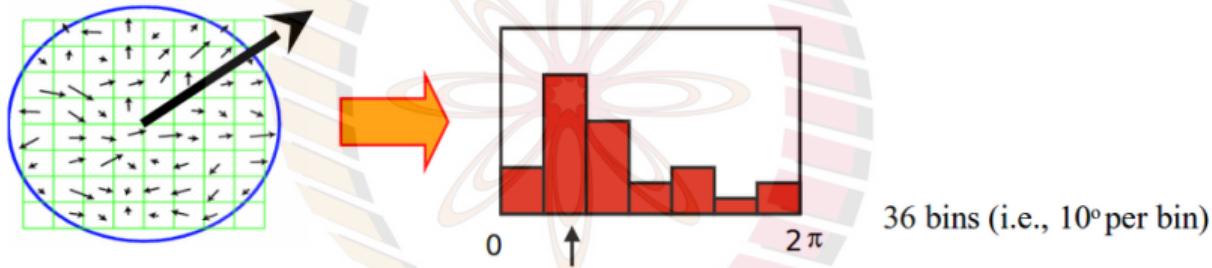


- Histogram entries are weighted by:
  - gradient magnitude, and
  - a Gaussian function with  $\sigma$  equal to 1.5 times scale of the keypoint

NPTEL

## SIFT: Orientation Estimation

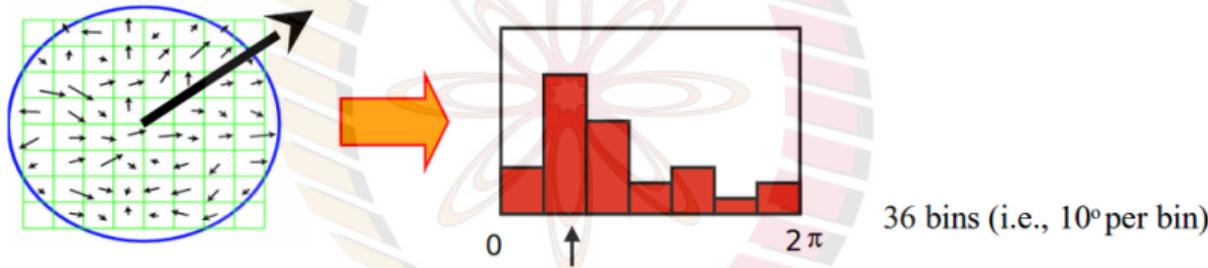
- Create histogram of gradient directions, within a region around the keypoint, at selected scale:



- Histogram entries are weighted by:
  - gradient magnitude, and
  - a Gaussian function with  $\sigma$  equal to 1.5 times scale of the keypoint
- Select **the peak as direction of keypoint**

# SIFT: Orientation Estimation

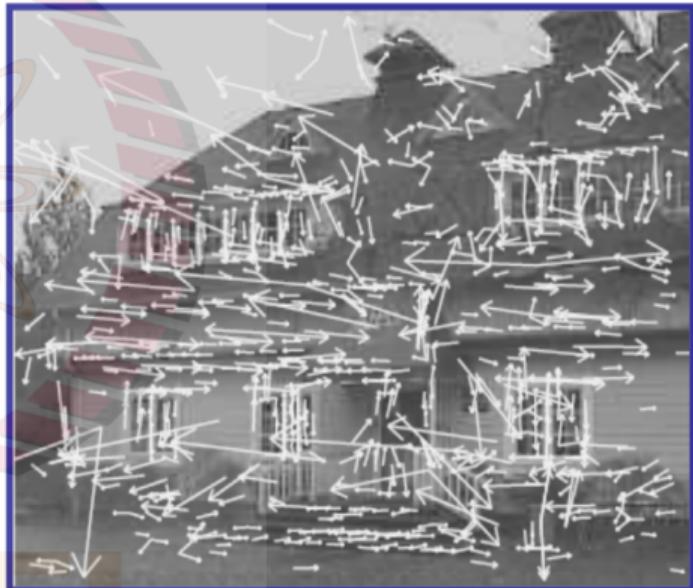
- Create histogram of gradient directions, within a region around the keypoint, at selected scale:



- Histogram entries are weighted by:
  - gradient magnitude, and
  - a Gaussian function with  $\sigma$  equal to 1.5 times scale of the keypoint
- Select **the peak as direction of keypoint**
- Introduce additional key points at same location if another peak is within 80% of max peak of histogram with different direction

Credit: Svetlana Lazebnik, UIUC

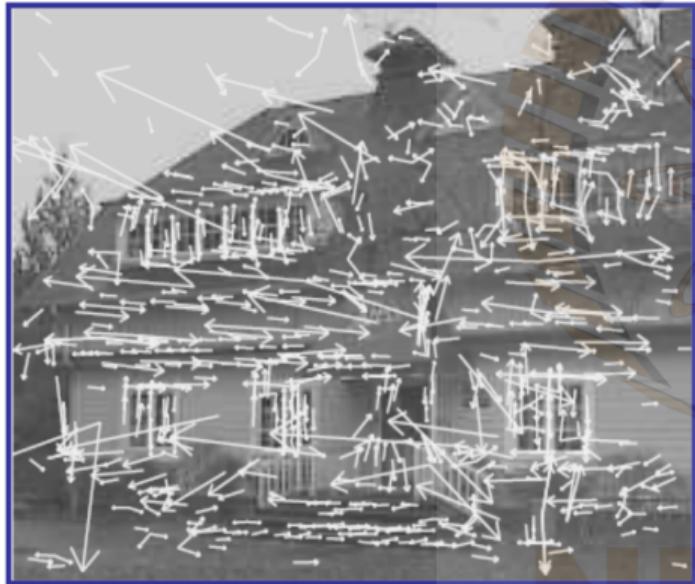
# SIFT



From 233x189 original image to 832 DoG Extrema

Credit: Mubarak Shah, University of Central Florida

# SIFT



From 832 DoG Extrema to 729 keypoints after low contrast threshold

Credit: Mubarak Shah, University of Central Florida

# SIFT



From 729 keypoints to 536 keypoints after testing ratio based on Hessian

Credit: Mubarak Shah, University of Central Florida

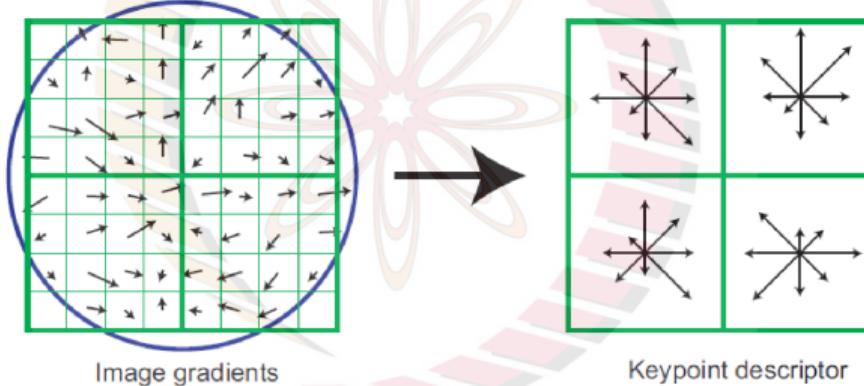
# SIFT Algorithm Stages

- **Step 1: Scale-space extrema Detection** - Detect interesting points (invariant to scale and orientation) using DOG.
- **Step 2: Keypoint Localization** - Determine location and scale at each candidate location, and select them based on stability.
- **Step 3: Orientation Estimation** - Use local image gradients to assign orientation to each localized keypoint. Preserve orientation, scale and location for each feature.
- **Step 4: Keypoint Descriptor** - Extract local image gradients at selected scale around keypoint and form a representation invariant to local shape and illumination distortion.

NPTEL

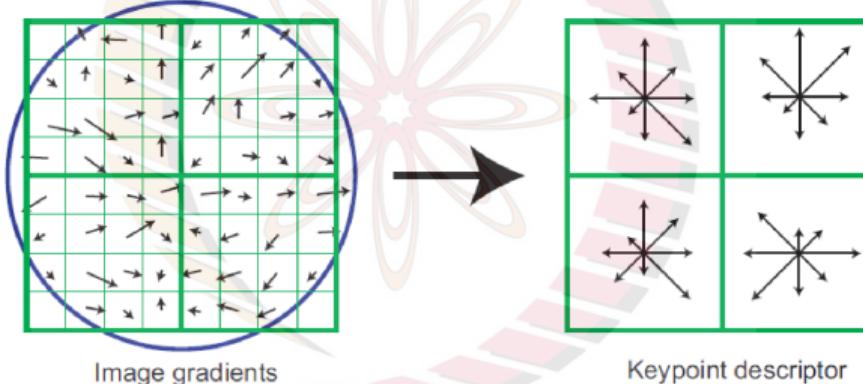
# SIFT: Keypoint Descriptor

- Compute gradient at each pixel in a  $16 \times 16$  window around the detected keypoint, using the appropriate level as detected.



## SIFT: Keypoint Descriptor

- Compute gradient at each pixel in a  $16 \times 16$  window around the detected keypoint, using the appropriate level as detected.

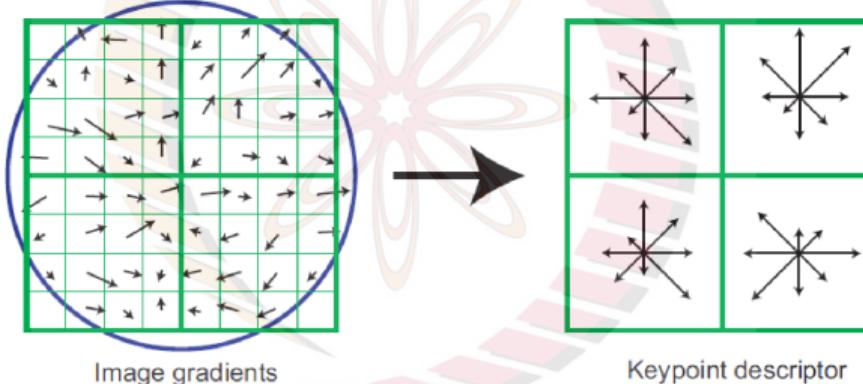


- Downweight gradients by a Gaussian fall-off function (blue circle) to reduce the influence of gradients far from the center.

NPTEL

# SIFT: Keypoint Descriptor

- Compute gradient at each pixel in a  $16 \times 16$  window around the detected keypoint, using the appropriate level as detected.



- Downweight gradients by a Gaussian fall-off function (blue circle) to reduce the influence of gradients far from the center.
- In each  $4 \times 4$  quadrant, compute a gradient orientation histogram using 8 orientation histogram bins.

Credit: Raquel Urtasun, Szeliski

## SIFT: Keypoint Descriptor

- The resulting 128 non-negative values form a raw version of the SIFT descriptor vector.
- To reduce the effects of contrast or gain (additive variations are already removed by the gradient), the 128-D vector is normalized to unit length.
- To further make the descriptor robust to other photometric variations, values are clipped to 0.2 and the resulting vector is once again renormalized to unit length.

Credit: Raquel Urtasun, Szeliski

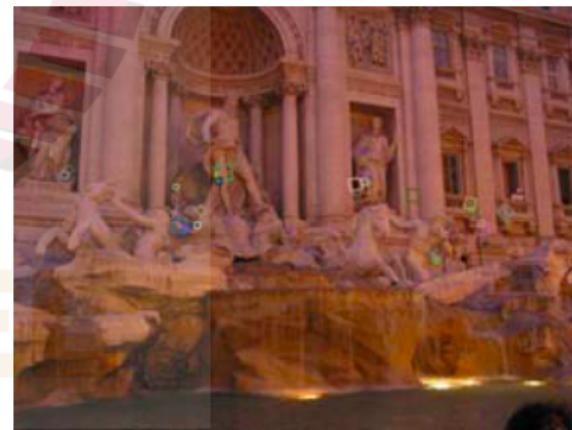


# SIFT

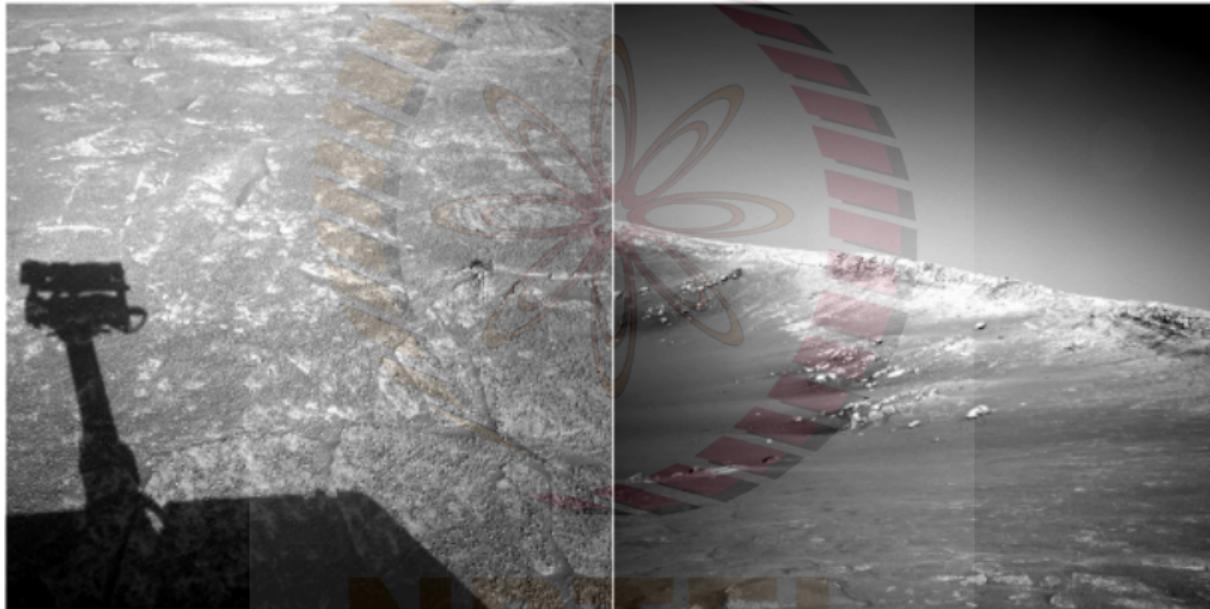
- Extraordinarily robust feature detection
- Changes in viewpoint: up to about 60 degree out of plane rotation
- Changes in illumination: sometimes even day vs night (below)
- Fast and efficient – can run in real-time



Credit: Raquel Urtasun, Szeliski



## SIFT: Example

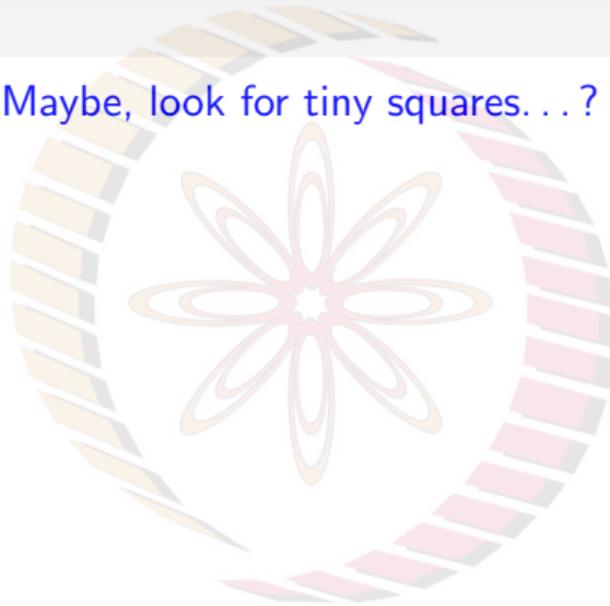


Mars Rover images

Credit: Raquel Urtasun, N Snavely

## SIFT: Example

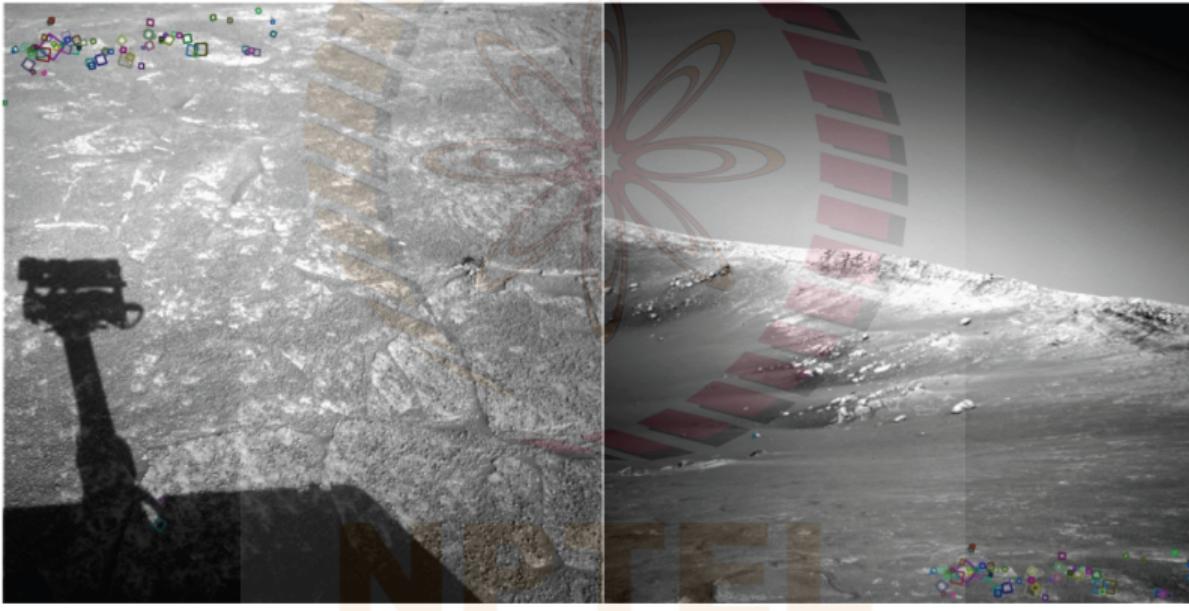
Maybe, look for tiny squares...?



NPTEL

# SIFT: Example

Maybe, look for tiny squares...?



Mars Rover images with SIFT feature matches

Credit: Raquel Urtasun, N Snavely

## SIFT: Invariances(Geometric Transformations)



Credit: Raquel Urtasun, Tinne Tuytelaars

## SIFT: Invariances(Photometric Transformations)



Credit: Raquel Urtasun, Tinne Tuytelaars

Vineeth N B (IIT-H)

§2.4 Feature Detectors

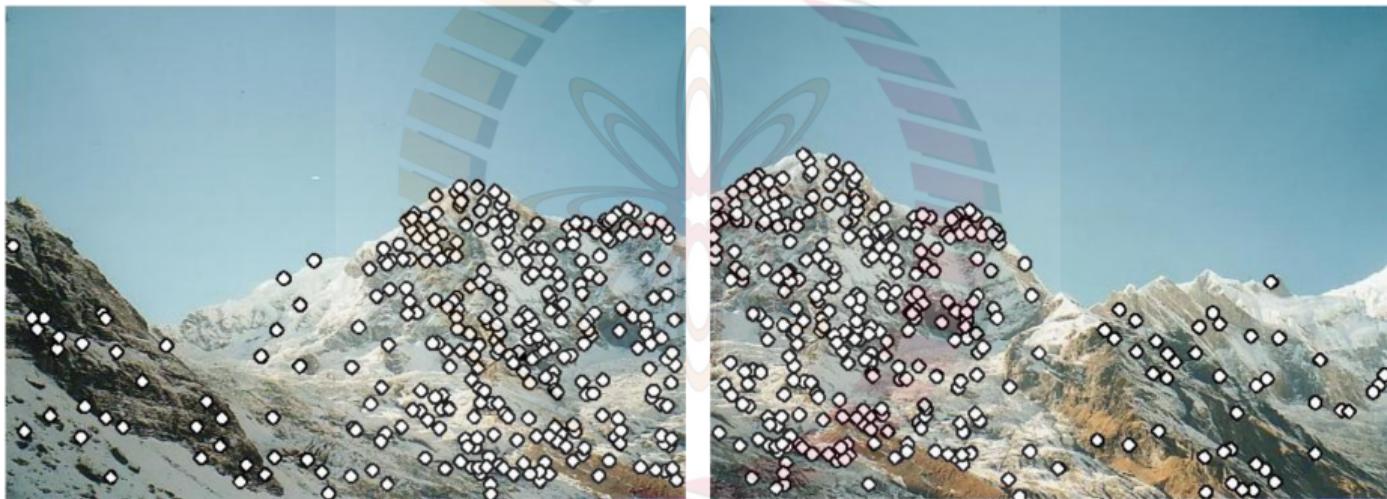
21 / 28

# SIFT Applications: Image Stitching



NPTEL

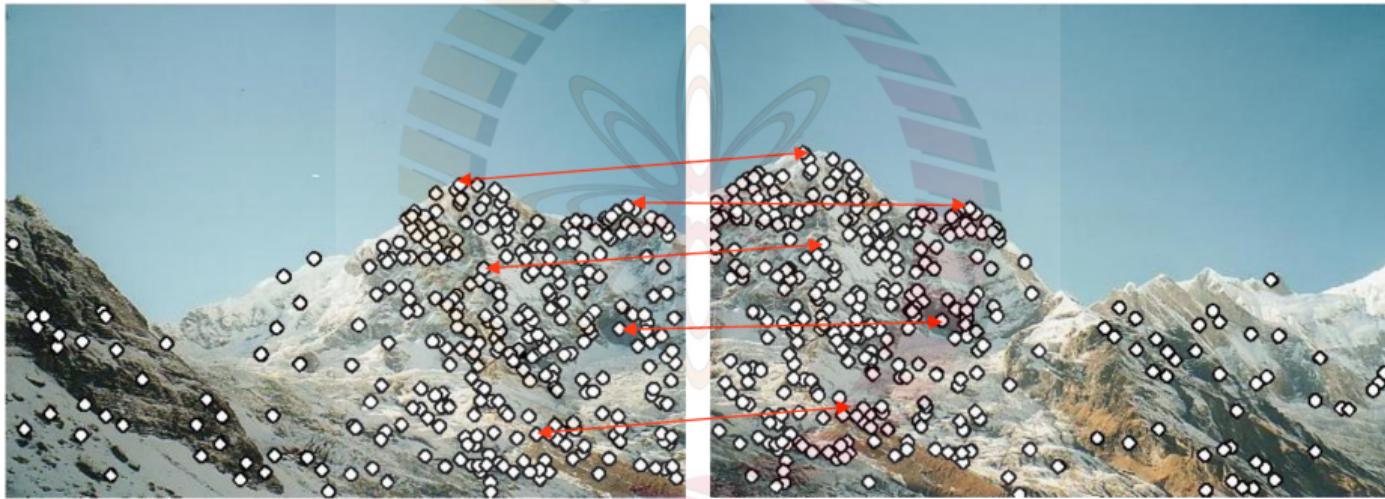
## SIFT Applications: Image Stitching



- Detect feature points in both images.

NPTEL

## SIFT Applications: Image Stitching



- Detect feature points in both images.
- Find corresponding pairs of feature points.

# SIFT Applications: Image Stitching



- Detect feature points in both images.
- Find corresponding pairs of feature points.
- Use the pairs to align the images.

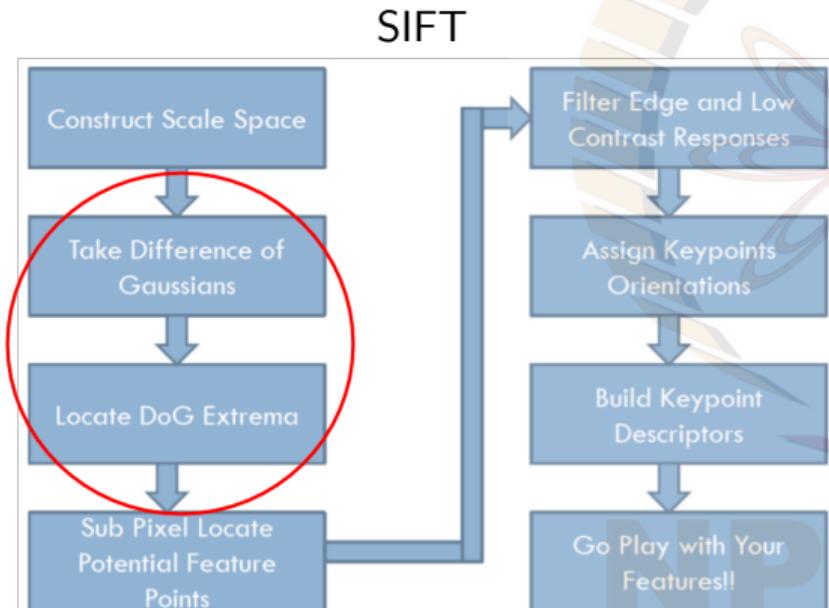
*Credit: Raquel Urtasun*

## More Resources

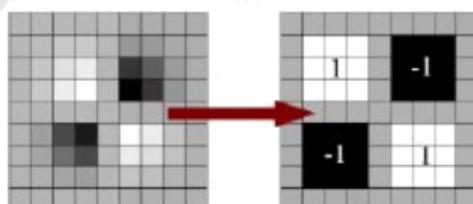
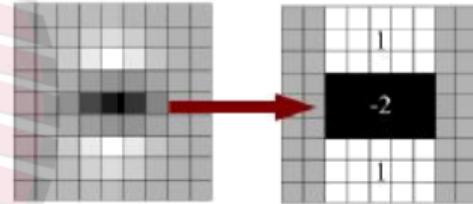
If you want to learn more on SIFT

- [The SIFT Keypoint Detector](#) by David Lowe
- Tutorial: [SIFT \(Scale-invariant feature transform\)](#)
- OpenCV-Python Tutorials: [Introduction to SIFT](#)
- Wikipedia: [Scale-invariant feature transform](#)
- [OpenSIFT: An Open-Source SIFT Library](#)

# SURF: Speeded Up Robust Features

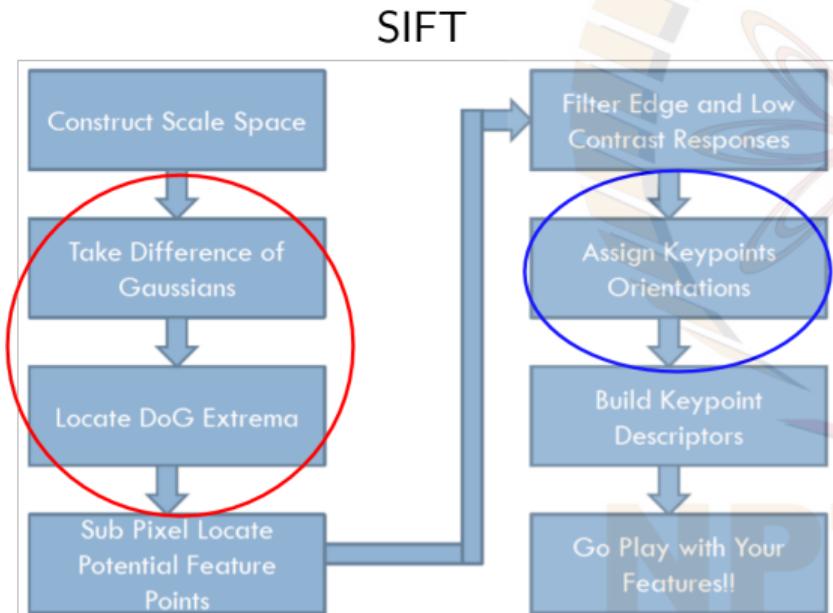


- Uses box filters instead of Gaussians to approximate Laplacians



- Uses Haar wavelets to get keypoint orientations

# SURF: Speeded Up Robust Features



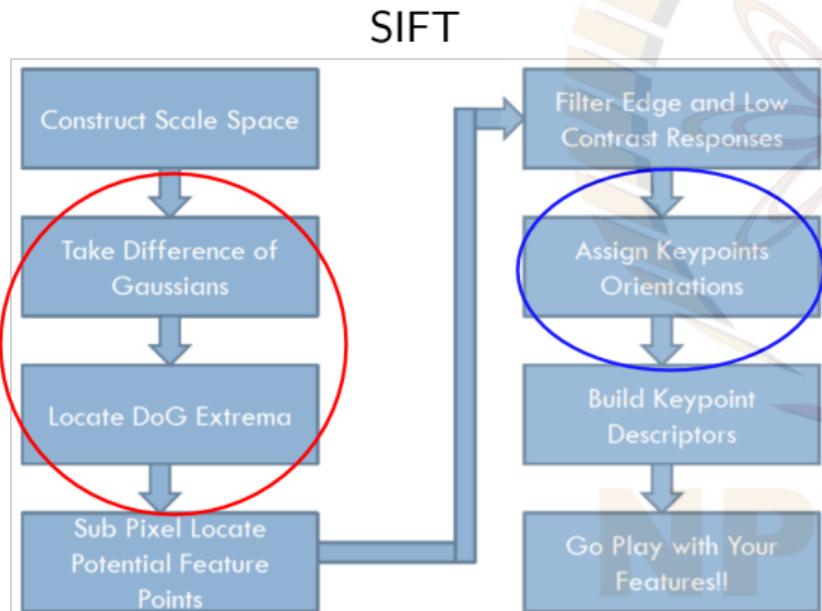
- Uses box filters instead of Gaussians to approximate Laplacians
- Uses Haar wavelets to get keypoint orientations

- Haar wavelets are simple filters which can be used to find gradients in the x and y directions



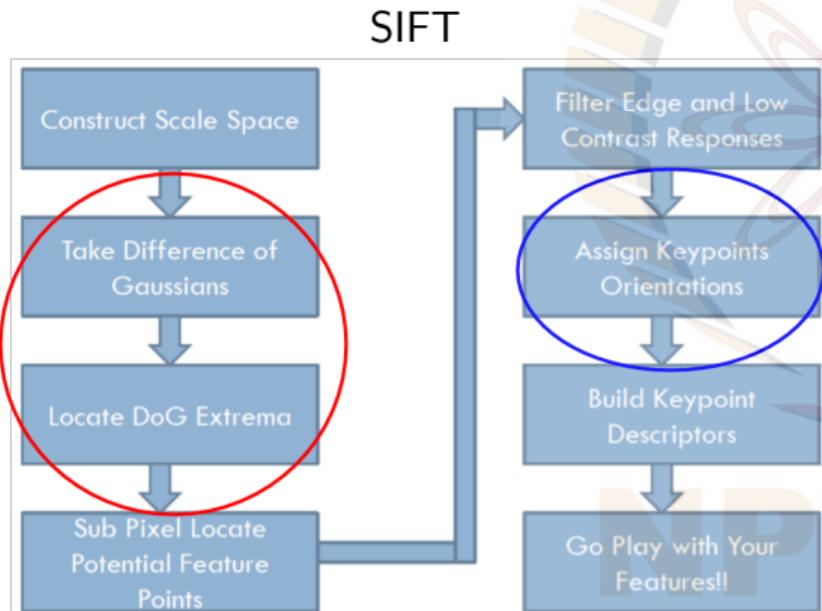
- SURF is good at handling blur and rotation variations
- SURF is not as good as SIFT on

# SURF: Speeded Up Robust Features



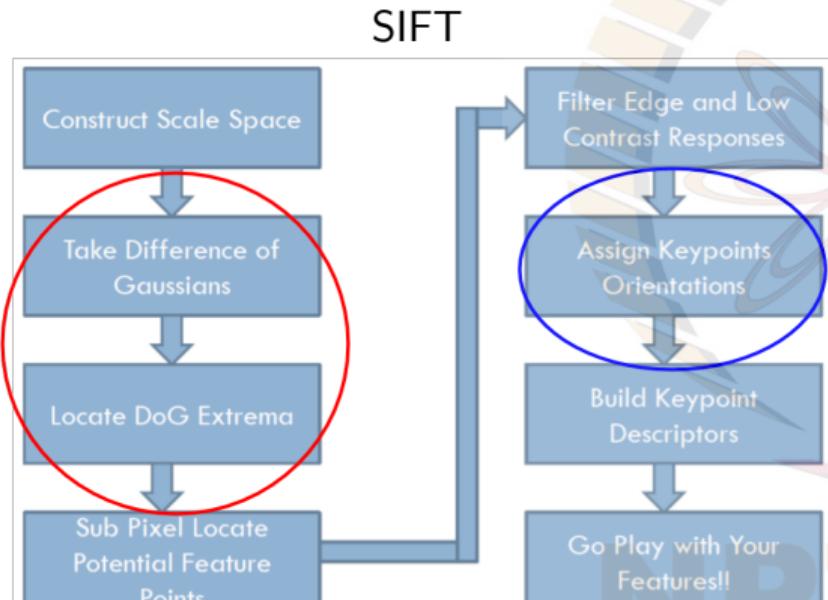
- Uses box filters instead of Gaussians to approximate Laplacians
- Uses Haar wavelets to get keypoint orientations
- SURF is good at handling blur and rotation variations
- SURF is not as good as SIFT on invariance to illumination and viewpoint changes
- SURF is  $\sim 3$  times faster than SIFT

# SURF: Speeded Up Robust Features



- Uses box filters instead of Gaussians to approximate Laplacians
- Uses Haar wavelets to get keypoint orientations
- SURF is good at handling blur and rotation variations
- SURF is not as good as SIFT on invariance to illumination and viewpoint changes
- SURF is  $\sim 3$  times faster than SIFT

# SURF: Speeded Up Robust Features



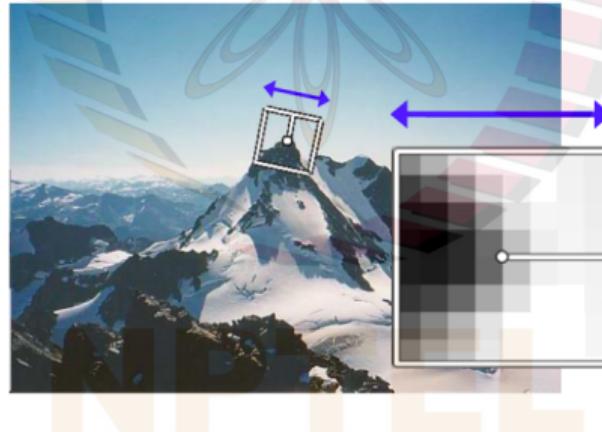
- Uses box filters instead of Gaussians to approximate Laplacians
- Uses Haar wavelets to get keypoint orientations
- SURF is good at handling blur and rotation variations
- SURF is not as good as SIFT on invariance to illumination and viewpoint changes
- SURF is  $\sim 3$  times faster than SIFT

For more information:

- <https://medium.com/data-breach/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e>
- <http://www.vision.ee.ethz.ch/~surf/>

# MOPS: Making Descriptor Rotation-invariant

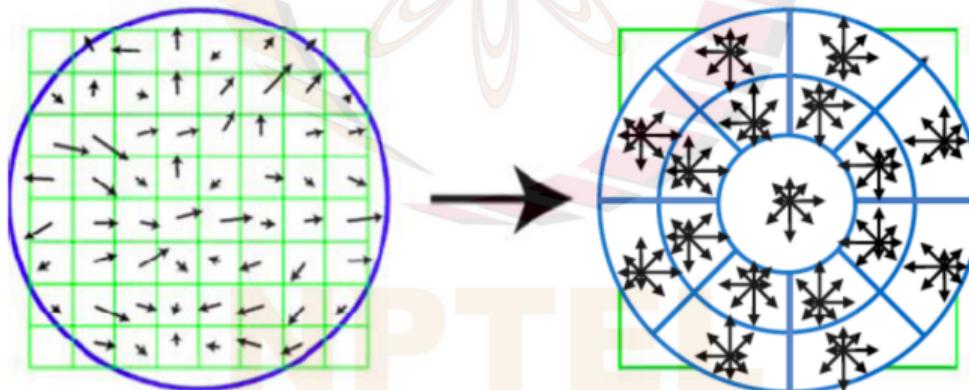
- Multiscale Oriented PatcheS descriptor
- Rotate patch according to its dominant gradient orientation.
- This puts the patches into a canonical orientation



Credit: Matthew Brown, Kristen Grauman, Raquel Urtasun

## Gradient Location-Orientation Histogram (GLOH)

- Variant of SIFT that uses a log-polar binning structure instead of four quadrants.
- Uses 17 spatial bins and 16 orientation bins.
- The 272D histogram is then projected onto a 128D descriptor using PCA trained on a large dataset.



Credit: Matthew Brown, Kristen Grauman, Raquel Urtasun

# Homework

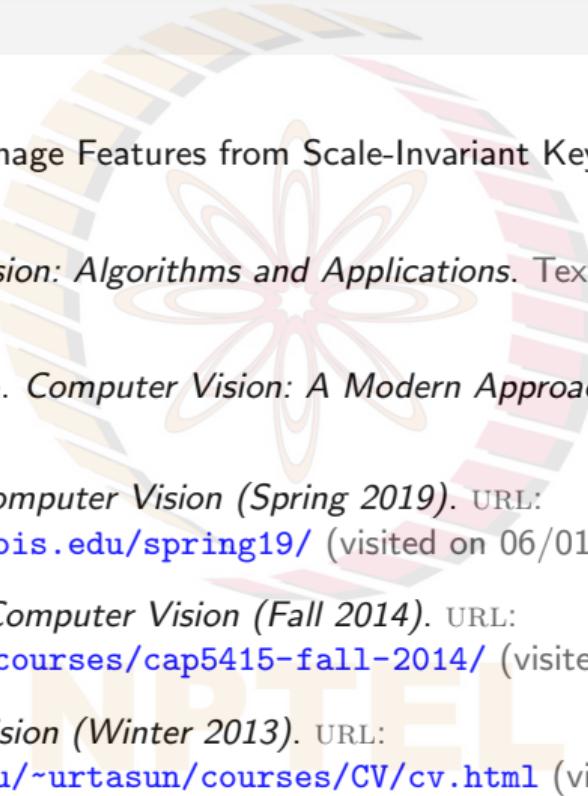
## Readings

- Section 3.5, Szeliski, *Computer Vision: Algorithms and Applications*
- For more information on SURF:
  - OpenCV-Python Tutorials : [Introduction to SURF](#)
  - Wikipedia: [Speeded up robust features](#)
- [Multi-Scale Oriented Patches](#)
- Other links provided on respective slides

## Questions

- Which descriptor performs better? SIFT or MOPS?
- Why is SIFT descriptor better than Harris Corner Detector?

# References

- 
-  David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), 91–110.
  -  Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer-Verlag, 2011.
  -  David Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. 2 edition. Boston: Pearson Education India, 2015.
  -  Lazebnik, Svetlana, CS 543 Computer Vision (Spring 2019). URL:  
<https://slazebni.cs.illinois.edu/spring19/> (visited on 06/01/2020).
  -  Shah, Mubarak, CAP 5415 - Computer Vision (Fall 2014). URL:  
<https://www.crcv.ucf.edu/courses/cap5415-fall-2014/> (visited on 06/01/2020).
  -  Urtasun, Raquel, Computer Vision (Winter 2013). URL:  
<https://www.cs.toronto.edu/~urtasun/courses/CV/cv.html> (visited on 06/01/2020).