Vision Systems

Lecture 7

Part 1

From Edges to Blobs and Corners

From Blobs to Corners

- In the following image, what are some interesting features to choose?

Credit: K Grauman, R Urtasun

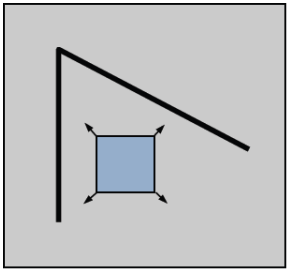
From Blobs to Corners

- Look for image regions that are unusual. How to define ”unusual”?
- Textureless patches are nearly impossible to localize.
- Patches with **large contrast changes** (gradients) are easier to localize.
- But straight line segments at a single orientation suffer from the **aperture problem** (we’ll see next slide), i.e., it is only possible to align the patches along the direction normal to the edge direction.
- Gradients in at least two (significantly) different orientations are the easiest, e.g., corners.

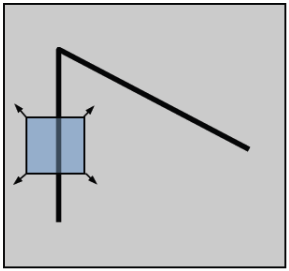
Credit: R Urtasun

From Blobs to Corners

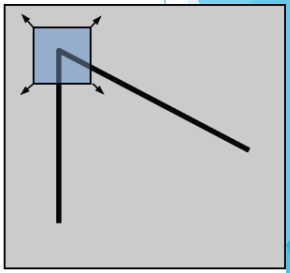
- Consider a small window of pixels. How does the window change when you shift it?



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

Credit: S Seitz, D Frolova, D Simakova, R Urtasun

Autocorrelation

- In the previous slide, how to quantify the ”significant” change of the window?

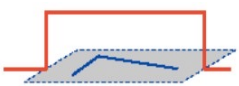
Autocorrelation

- In the previous slide, how to quantify the "significant" change of the window?
- Answer: **Autocorrelation function**. Compute the sum of squared differences between pixel intensities with respect to small variations in the image patch position.

$$E_{AC}(\Delta \mathbf{u}) = \sum_{x,y} w(\mathbf{p}_i)[I(\mathbf{p}_i + \delta u) - I(\mathbf{p}_i)]^2$$

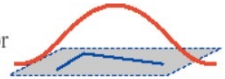
where $\mathbf{p}_i = (x, y)$, a particular position on the image.

Window function $w(x,y)$ =

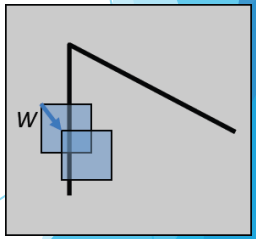


1 in window, 0 outside

or



Gaussian



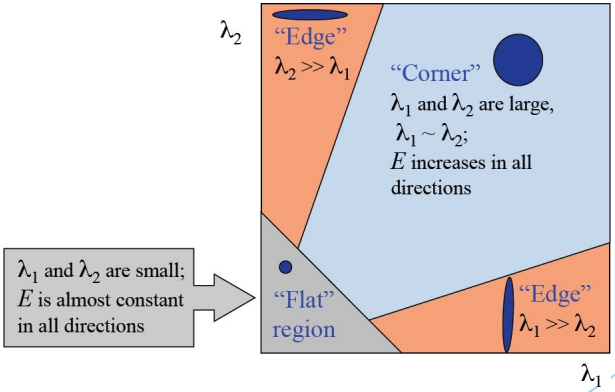
Credit: R Urtasun

Computing Autocorrelation

- How do the eigenvalues determine if an image point is a corner?

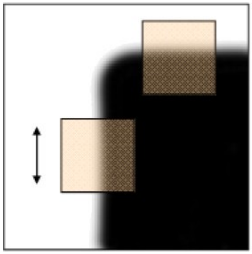
Computing Autocorrelation

- How do the eigenvalues determine if an image point is a corner?



Credit: N Snavely, R Urtasun

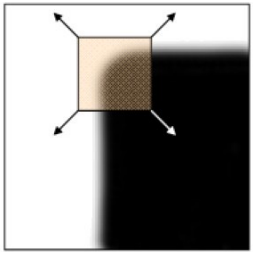
Computing Autocorrelation



"edge":

$\lambda_1 >> \lambda_2$

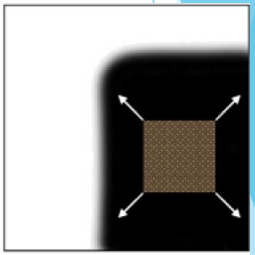
$\lambda_2 >> \lambda_1$



"corner":

λ_1 and λ_2 are large,

$\lambda_1 \sim \lambda_2$;



"flat" region

λ_1 and λ_2 are small;

Credit: K Grauman, R Urtasun

Harris Corner Detector

- Compute gradients at each point in the image.
- Compute **A** for each image window to get its cornerness scores.
- Compute the eigenvalues/compute the following function M_c

$$M_c = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = det(A) - \kappa\, trace^2(A)$$

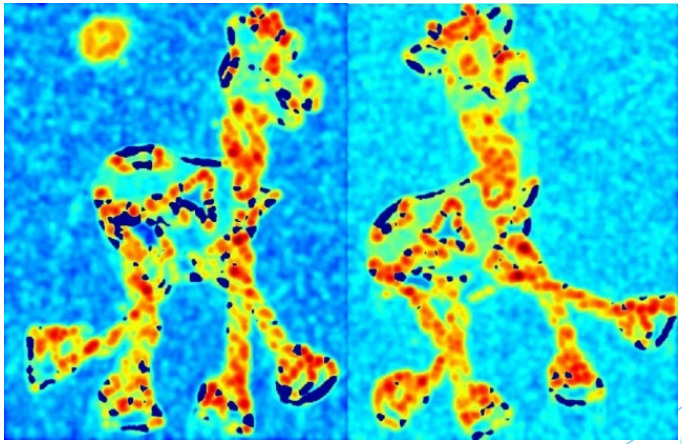
- Find points whose surrounding window gave larger cornerness response (M_c >threshold)
- Take points of local maxima, perform non-maximum suppression.

Credit: R Urtasun

Harris Corner Detector: Example

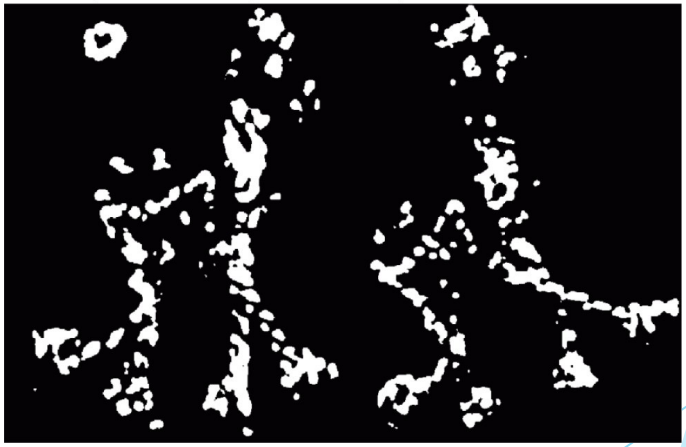
Credit: K Grauman, R Urtasun

Computing Cornerness



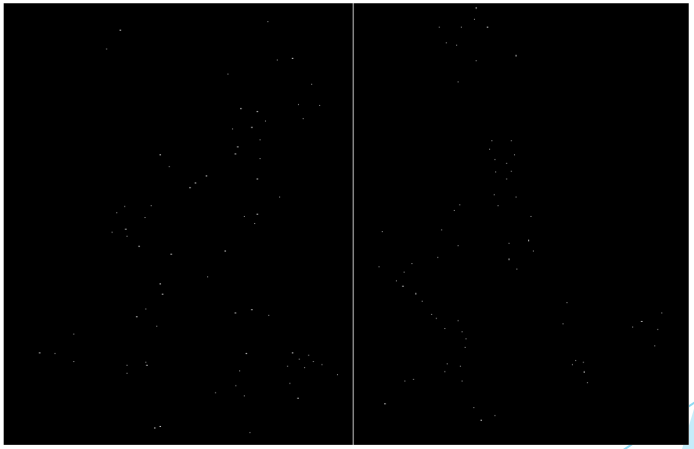
Credit: K Grauman, R Urtasun

Finding High Response



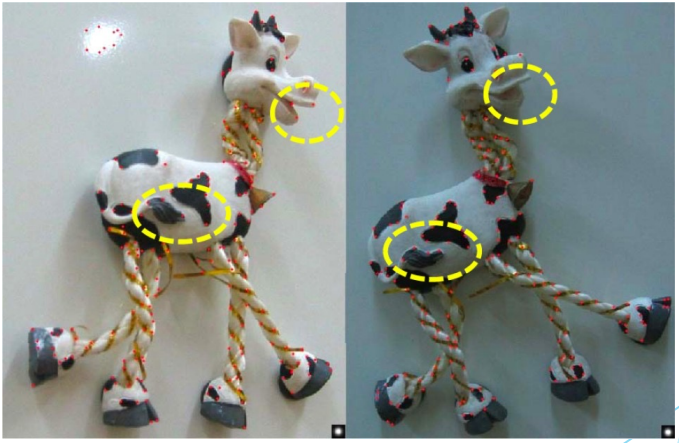
Credit: K Grauman, R Urtasun

Non-max Suppression



Credit: K Grauman, R Urtasun

Results



Credit: K Grauman, R Urtasun

Harris Corner Detector: Variants

- Harris and Stephens '88 is rotationally invariant and downweights edge-like features where $\lambda_1 >> \lambda_0$.

$$det(\mathbf{A}) - \alpha trace(\mathbf{A})^2 = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

- Triggs '04 suggested $\lambda_0 - \alpha\lambda_1$.
- Brown et al, '05 use harmonic mean:

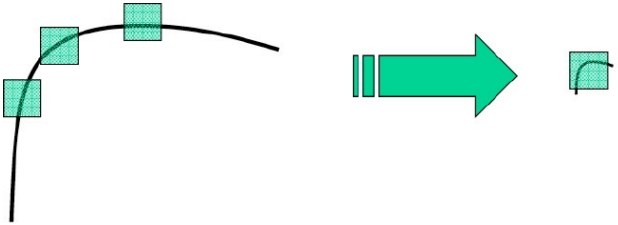
$$\frac{det(\mathbf{A})}{trace(\mathbf{A})} = \frac{\lambda_0\lambda_1}{\lambda_0 + \lambda_1}$$

which is smoother when $\lambda_0 \approx \lambda_1$

Credit: R Urtasun

Harris Corner Detector: Properties

- Scale-invariant?



All points will be classified as edges

Corner !

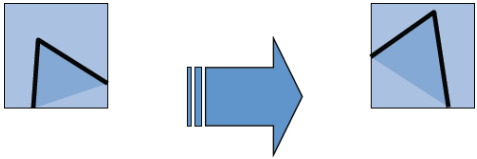
Credit: R Urtasun

Harris Corner Detector: Properties

- Rotation-invariant?

$$\mathbf{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} = \mathbf{U} \begin{bmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{bmatrix} \mathbf{U}^T \quad \text{with} \quad \mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

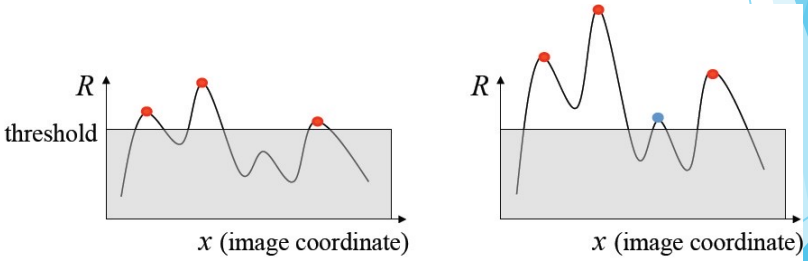
- Relative cornerness remains the same!



Credit: N Snavely, R Urtasun

Harris Corner Detector: Properties

- Photometric change: Affine intensity change $I = aI + b$?
- Only derivatives are used, so it's invariant to shift $I = I + b$.
- What about intensity scale?



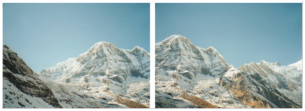
Partially invariant to affine intensity change

Credit: K Grauman, R Urtasun

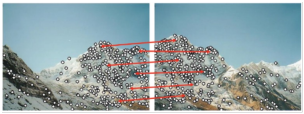
Part 2

Scale Space, Image Pyramids and Filter Banks

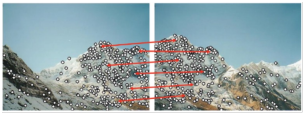
Review

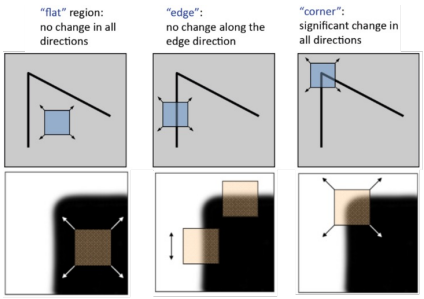


Review

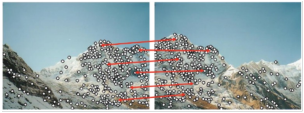


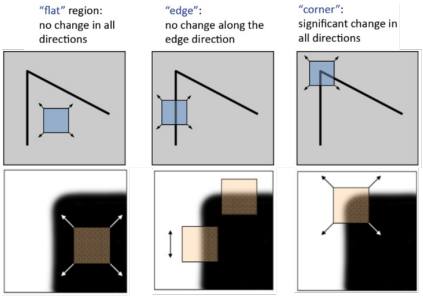
Review



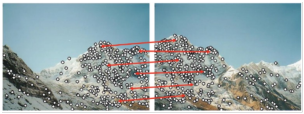

"flat" region:
no change in all directions
"edge":
no change along the edge direction
"corner":
significant change in all directions
"flat" region
λ_1 and λ_2 are small;
"edge":
$\lambda_1 >> \lambda_2$
$\lambda_2 >> \lambda_1$
"corner":
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;

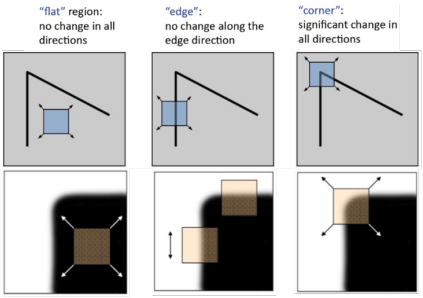
Review

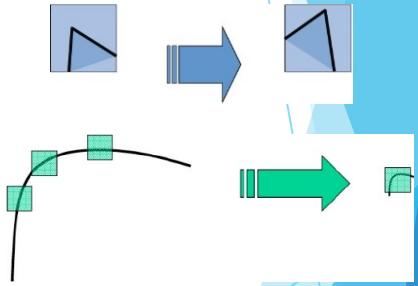



"flat" region:
no change in all directions
"edge":
no change along the edge direction
"corner":
significant change in all directions
"flat" region
λ_1 and λ_2 are small;
"edge":
$\lambda_1 >> \lambda_2$
$\lambda_2 >> \lambda_1$
"corner":
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;

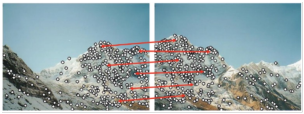
Review

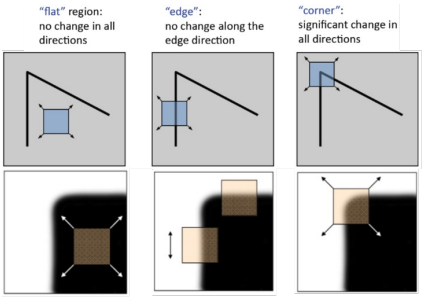


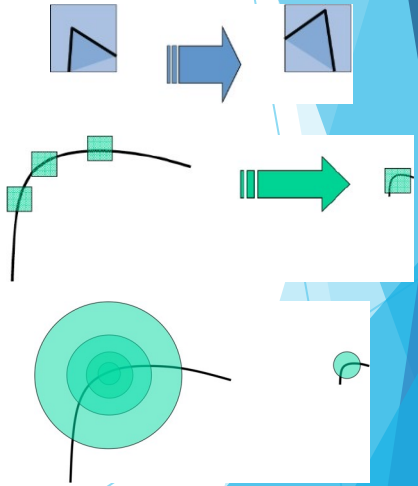

"flat" region:
no change in all directions
"edge":
no change along the edge direction
"corner":
significant change in all directions
"flat" region
λ_1 and λ_2 are small;
"edge":
$\lambda_1 >> \lambda_2$
$\lambda_2 >> \lambda_1$
"corner":
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;



Review




"flat" region:
no change in all directions
"edge":
no change along the edge direction
"corner":
significant change in all directions
"flat" region
λ_1 and λ_2 are small;
"edge":
$\lambda_1 >> \lambda_2$
$\lambda_2 >> \lambda_1$
"corner":
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;



Scale-Invariant Interest Point Detection

How can we independently select interest points in each image, such that detections are repeatable across different scales?

Scale-Invariant Interest Point Detection

- How can we independently select interest points in each image, such that detections are repeatable across different scales?
  - Extract features at a variety of scales, e.g., by using multiple resolutions in a pyramid, and then matching features at the same level.

Scale-Invariant Interest Point Detection

- How can we independently select interest points in each image, such that detections are repeatable across different scales?
  - Extract features at a variety of scales, e.g., by using multiple resolutions in a pyramid, and. then matching features at the same level.
  - When does this work?

Scale-Invariant Interest Point Detection

- How can we independently select interest points in each image, such that detections are repeatable across different scales?
  - Extract features at a variety of scales, e.g., by using multiple resolutions in a pyramid, and then matching features at the same level.
  - When does this work?
  - More efficient to extract features stable in both location and scale.

Scale-Invariant Interest Point Detection

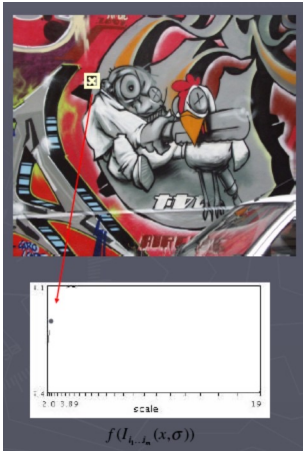
- How can we independently select interest points in each image, such that detections are repeatable across different scales?
  - Extract features at a variety of scales, e.g., by using multiple resolutions in a pyramid, and then matching features at the same level.
  - When does this work?
  - More efficient to extract features stable in both location and scale.
  - Find scale that gives local maxima of a function f in both position and scale.

$$f(I_{i_1 \ldots i_m}(x,\sigma)) \;=\; f(I_{i_1 \ldots i_m}(x',\sigma'))$$

Credit: R Urtasun

Automatic Scale Selection

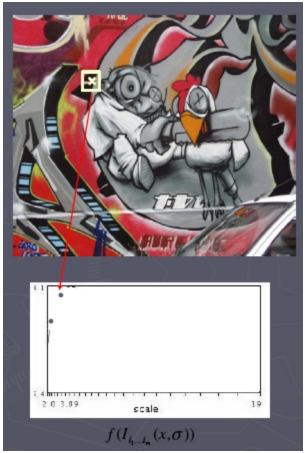
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

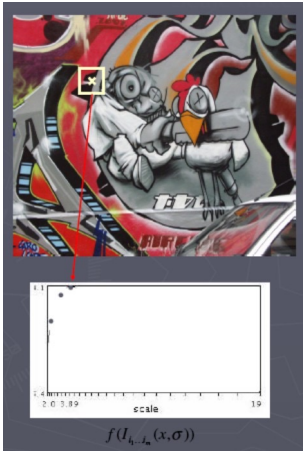
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

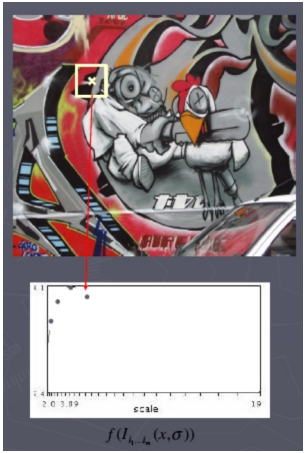
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

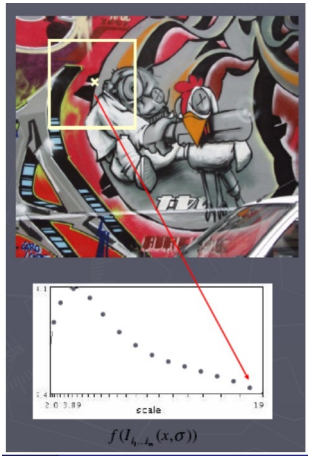
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

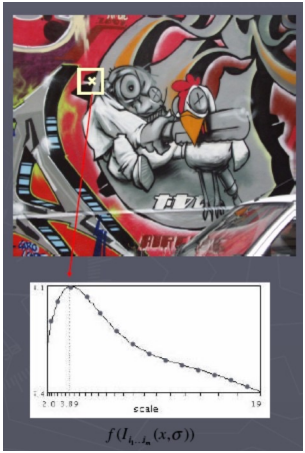
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

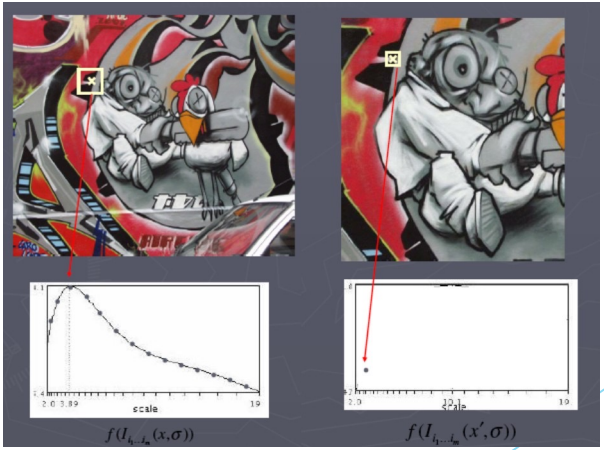
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

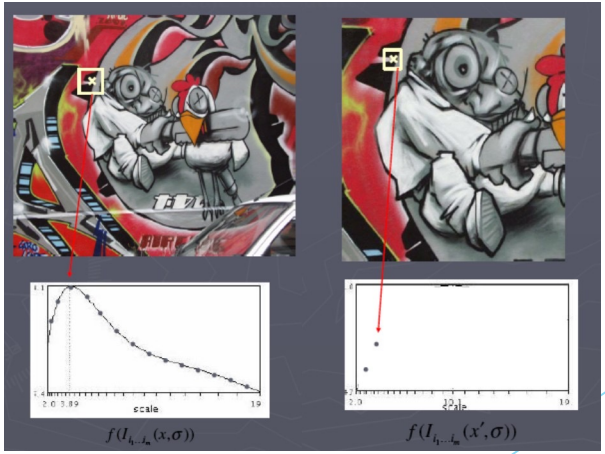
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

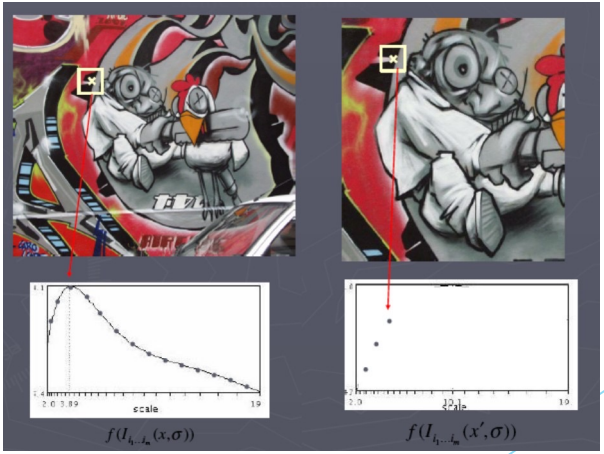
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

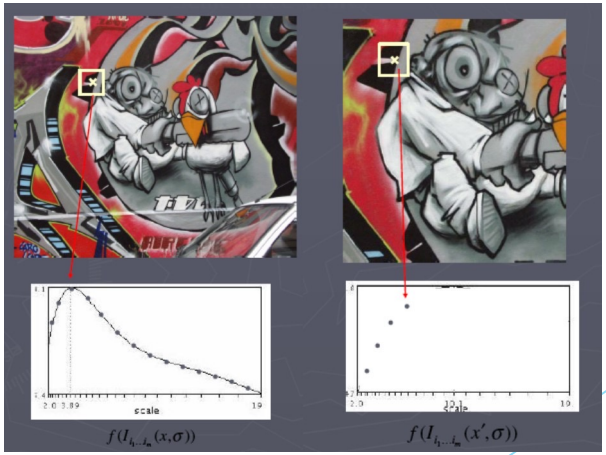
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

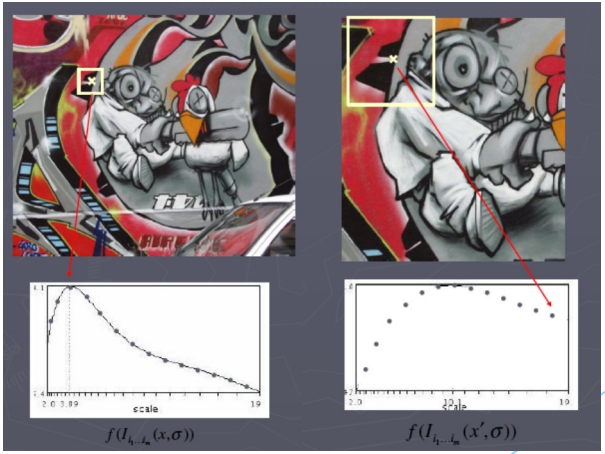
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

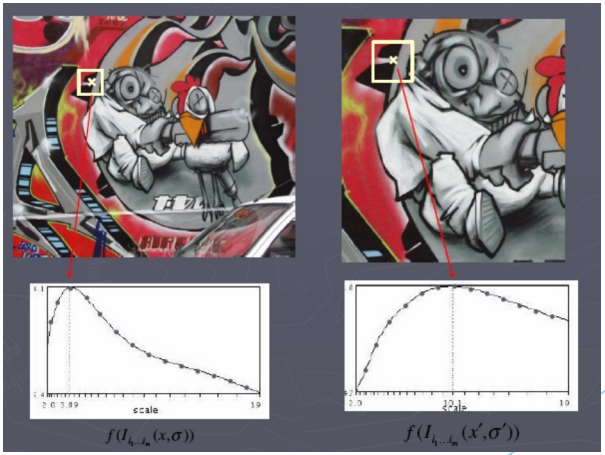
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

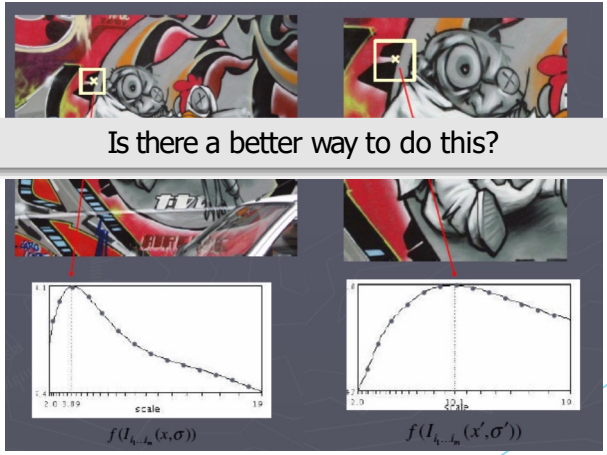
Function responses for increasing scale (scale signature).



Credit: R Urtasun

Automatic Scale Selection

Function responses for increasing scale (scale signature).



Is there a better way to do this?

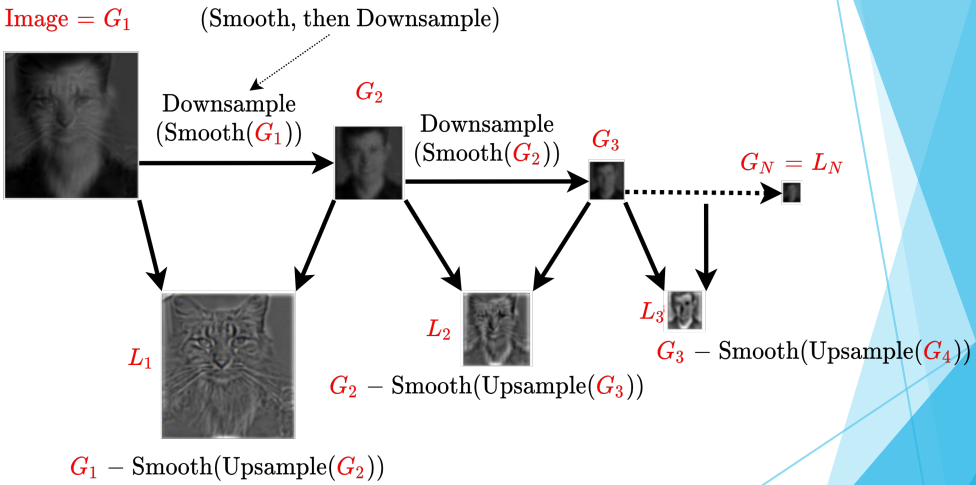
Automatic Scale Selection: Implementation

Instead of computing f for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid.

Sometimes need to create in-between levels, e.g., a $\frac{3}{4}$ size image.

Credit: R Urtasun

Gaussian and Laplacian Pyramid



Credit: Derek Hoiem

Image Pyramids: Uses

- Compression

Image Pyramids: Uses

- Compression
- Object detection

Image Pyramids: Uses

- Compression
- Object detection
  - Scale search

Image Pyramids: Uses

- Compression
- Object detection
  - Scale search
  - Features

Image Pyramids: Uses

- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points

Image Pyramids: Uses

- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points
- Registration

Image Pyramids: Uses

- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points
- Registration
  - Coarse-to-fine Image Registration

Image Pyramids: Uses

- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points
- Registration
  - Coarse-to-fine Image Registration

Coarse-to-fine Image Registration:

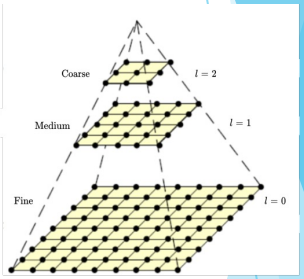
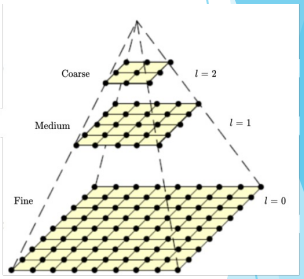


Image Pyramids: Uses

- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points
- Registration
  - Coarse-to-fine Image Registration

Coarse-to-fine Image Registration:

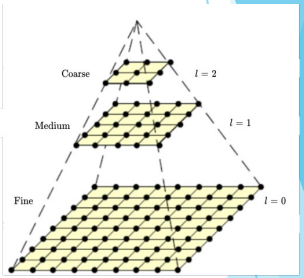


- Compute Gaussian pyramid.

Image Pyramids: Uses

- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points
- Registration
  - Coarse-to-fine Image Registration

Coarse-to-fine Image Registration:

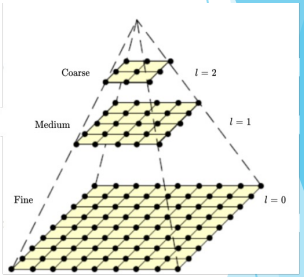


- Compute Gaussian pyramid.
- Align with coarse pyramid.

Image Pyramids: Uses

- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points
- Registration
  - Coarse-to-fine Image Registration

Coarse-to-fine Image Registration:



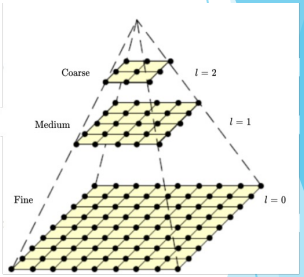
- Compute Gaussian pyramid.
- Align with coarse pyramid.
- Successively align with finer pyramids.

Image Pyramids: Uses

- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points
- Registration
  - Coarse-to-fine Image Registration

Credit: Derek Hoiem

Coarse-to-fine Image Registration:



- Compute Gaussian pyramid.
- Align with coarse pyramid.
- Successively align with finer pyramids.
  - Search smaller range.

Texture in Images

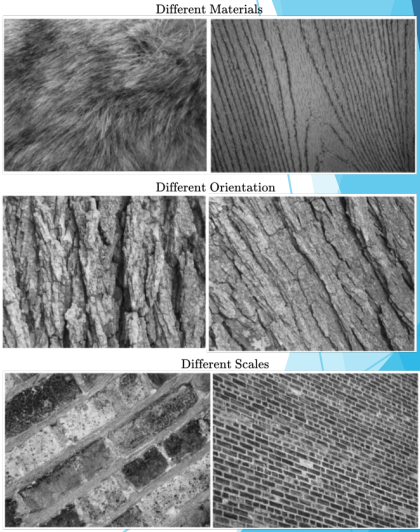
Textures:

- Regular or stochastic patterns caused by bumps, grooves and/or markings.

Texture in Images

Textures:

- Regular or stochastic patterns caused by bumps, grooves and/or markings.
- Gives us information about spatial arrangement of colors or intensities in an image.

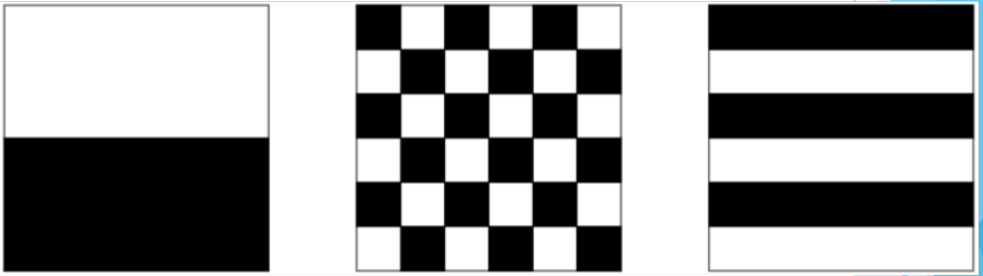


Credit: Derek Hoiem

Texture in Images

Conveys more information that can be exploited to match regions of interest in images.

Histogram conveys 50% white pixels and 50% black pixels



(Block Pattern) (Checkerboard Pattern) (Striped Pattern)

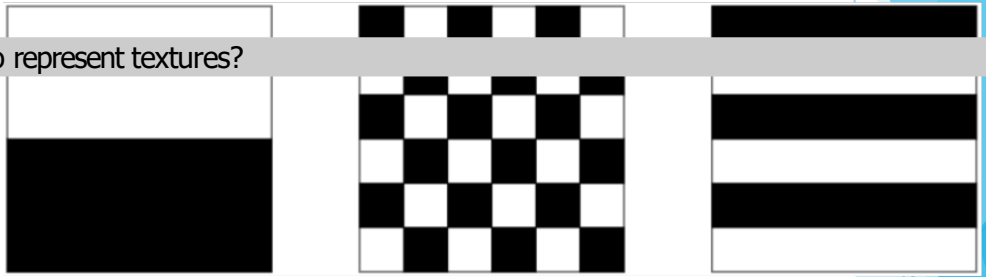
Drastically different textures

Credit: Linda G Shapiro

Texture in Images

Conveys more information that can be exploited to match regions of interest in images.

Histogram conveys 50% white pixels and 50% black pixels



(Block Pattern) (Checkerboard Pattern) (Striped Pattern)

Drastically different textures

How to represent textures?

Texture in Images

Conveys more information that can be exploited to match regions of interest in images.

Histogram conveys 50% white pixels and 50% black pixels

(Block Pattern) (Checkerboard Pattern) (Striped Pattern)

Drastically different textures

How to represent textures?

- Compute responses of blobs and edges at various orientations and scales.

Texture in Images

Conveys more information that can be exploited to match regions of interest in images.

Histogram conveys 50% white pixels and 50% black pixels

(Block Pattern) (Checkerboard Pattern) (Striped Pattern)

Drastically different textures

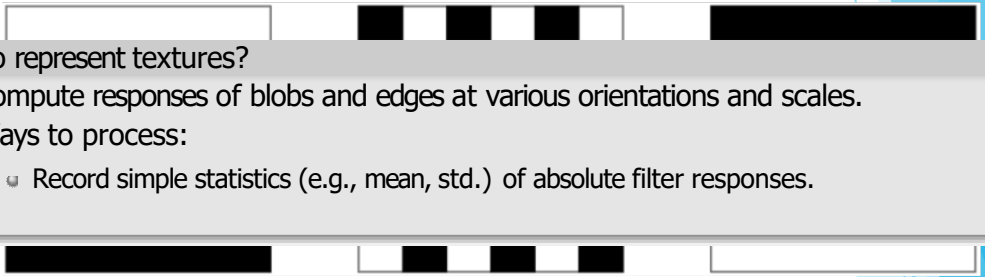
How to represent textures?

- Compute responses of blobs and edges at various orientations and scales.
- Ways to process:

Texture in Images

Conveys more information that can be exploited to match regions of interest in images.

Histogram conveys 50% white pixels and 50% black pixels



(Block Pattern) (Checkerboard Pattern) (Striped Pattern)

Drastically different textures

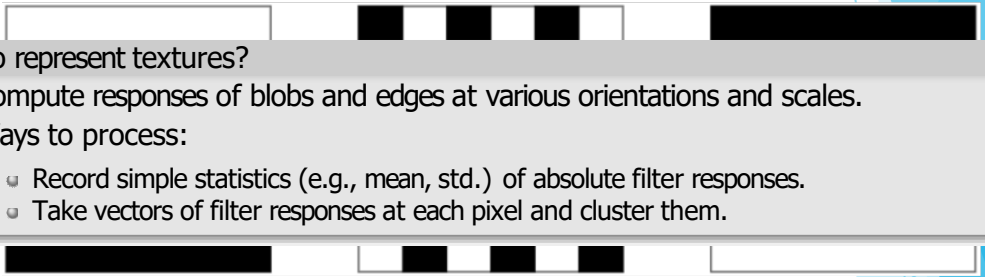
How to represent textures?

- Compute responses of blobs and edges at various orientations and scales.
- Ways to process:
  - Record simple statistics (e.g., mean, std.) of absolute filter responses.

Texture in Images

Conveys more information that can be exploited to match regions of interest in images.

Histogram conveys 50% white pixels and 50% black pixels



How to represent textures?

- Compute responses of blobs and edges at various orientations and scales.
- Ways to process:
  - Record simple statistics (e.g., mean, std.) of absolute filter responses.
  - Take vectors of filter responses at each pixel and cluster them.

(Block Pattern) (Checkerboard Pattern) (Striped Pattern)

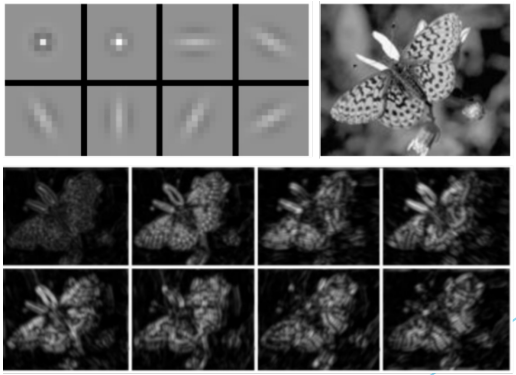
Drastically different textures

Filter Banks

- An array of bandpass filters that separates the input signal into multiple components, each one carrying a single frequency sub-band of the original signal.

Filter Banks

- An array of bandpass filters that separates the input signal into multiple components, each one carrying a single frequency sub-band of the original signal.
- Process image with each filter and keep responses (or squared/abs responses).



Credit: Derek Hoiem

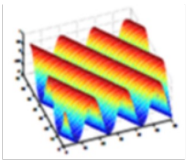
Gabor Filters

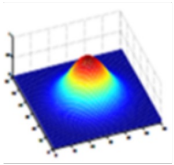
- Special classes of bandpass filters (i.e., they allow a certain ‘band’ of frequencies and reject the others).

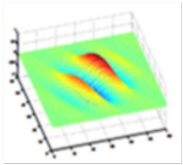
Gabor Filters

- Special classes of bandpass filters (i.e., they allow a certain 'band' of frequencies and reject the others).
- A Gabor filter can be viewed as a sinusoidal signal of particular frequency and orientation, modulated by a Gaussian wave.

A 2-D Gaussian







A sinusoid oriented 30° with x-axis

A corresponding 2-D Gabor Filter

A 2-D Gabor filter obtained by modulating the sine wave with a Gaussian

2-D Gabor Filter

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}} e^{i\left(2\pi \frac{x'}{\lambda} + \psi\right)}$$

- where:
- $x' = x\cos\theta + y\sin\theta$
- $y' = -x\sin\theta + y\cos\theta$
- θ – Orientation of the normal to the parallel stripes of Gabor function.

2-D Gabor Filter

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}} e^{i\left(2\pi \frac{x'}{\lambda} + \psi\right)}$$

- where:
- $x' = x\cos\theta + y\sin\theta$
- $y' = -x\sin\theta + y\cos\theta$
- θ – Orientation of the normal to the parallel stripes of Gabor function.
- λ – Wavelength of the sinusoidal component.

2-D Gabor Filter

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}} e^{i\left(2\pi \frac{x'}{\lambda} + \psi\right)}$$

- where:
- $x' = x\cos\theta + y\sin\theta$
- $y' = -x\sin\theta + y\cos\theta$
- θ – Orientation of the normal to the parallel stripes of Gabor function.
- λ – Wavelength of the sinusoidal component.
- ψ – Phase offset of the sinusoidal function.

2-D Gabor Filter

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}} e^{i\left(2\pi \frac{x'}{\lambda} + \psi\right)}$$

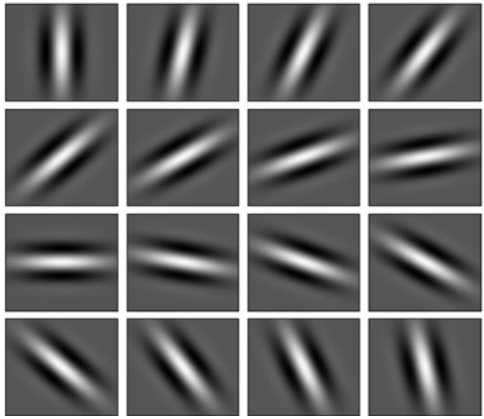
- where:
- $x' = x\cos\theta + y\sin\theta$
- $y' = -x\sin\theta + y\cos\theta$
- θ – Orientation of the normal to the parallel stripes of Gabor function.
- λ – Wavelength of the sinusoidal component.
- ψ – Phase offset of the sinusoidal function.
- σ – Standard deviation of the Gaussian envelope.

2-D Gabor Filter

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}} e^{i\left(2\pi \frac{x'}{\lambda} + \psi\right)}$$

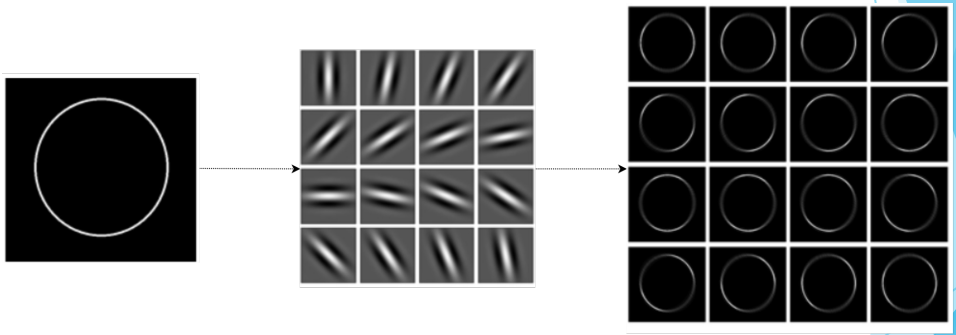
- where:
- $x' = x \cos\theta + y \sin\theta$
- $y' = -x \sin\theta + y \cos\theta$
- θ – Orientation of the normal to the parallel stripes of Gabor function.
- λ – Wavelength of the sinusoidal component.
- ψ – Phase offset of the sinusoidal function.
- σ – Standard deviation of the Gaussian envelope.
- γ – Spatial aspect ratio and specifies the ellipticity of the support of Gabor function.

Gabor Filter Banks



Bank of 16 Gabor filters at an orientation of 11.250 (i.e. if the first filter is at 00.00, then the second will be at 11.25, the third will be at 22.50, and so on.)

Gabor Filter Banks



Input Image of a Circle

A bank of 16 Gabor filters

Output Image of the circle after passed through indiviual Gabor filters

Steerable Filter Banks

Steerable Filters are a class of oriented filters that can be expressed as a linear combination of a set of basis filters.

Steerable Filter Banks

Steerable Filters are a class of oriented filters that can be expressed as a linear combination of a set of basis filters.

- For an isotropic Gaussian filter, $G(x, y) = e^{-(x^2+y^2)}$,

$$G_1^{\theta^\circ} = G_1^{0^\circ} \cos(\theta) + G_1^{90^\circ} \sin(\theta)$$

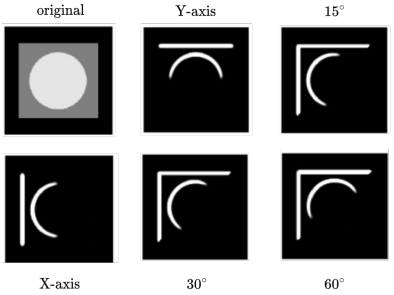
Steerable Filter Banks

Steerable Filters are a class of oriented filters that can be expressed as a linear combination of a set of basis filters.

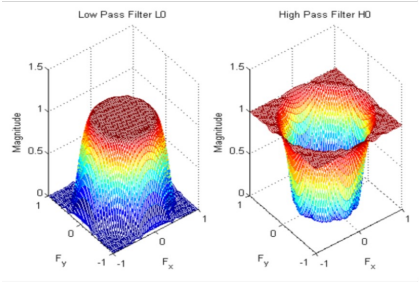
- For an isotropic Gaussian filter, $G(x, y) = e^{-(x^2+y^2)}$,

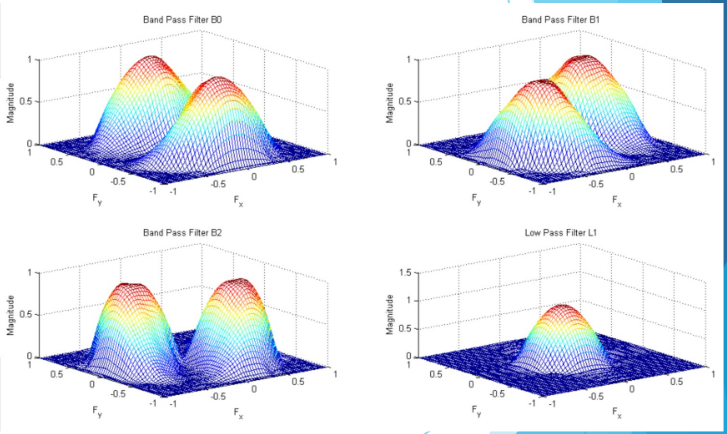
$$G_1^{\theta^\circ} = G_1^{0^\circ} \cos(\theta) + G_1^{90^\circ} \sin(\theta)$$

where $G_1^{\theta^\circ}$ is the first derivative of G at angle θ.



Steerable Filter Banks





Homework

Readings

Chapter 2, Szeliski, *Computer Vision: Algorithms and Applications*

Questions

Why is camouflage attire effective? How?

How is texture different from noise?

Will scale-invariant filters be effective in matching pictures containing Matryoshka (or Russian nesting) dolls?

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).