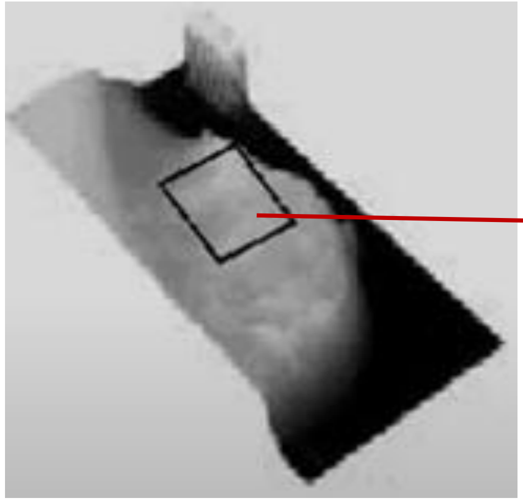
Computer Vision

Feature Extraction
(Local texture representation)

What is Texture?

- Provides information of the spatial arrangement of colors/ intensities in an image
- Characterised by spatial distribution of intensity levels in neighbourhood



Image

Patch with texture

Image texture

- A group of pixels having similar properties is called Texel
- If Texel is repeated spatially, we get a particular texture
- Human can distinguish depth of objects based on texture

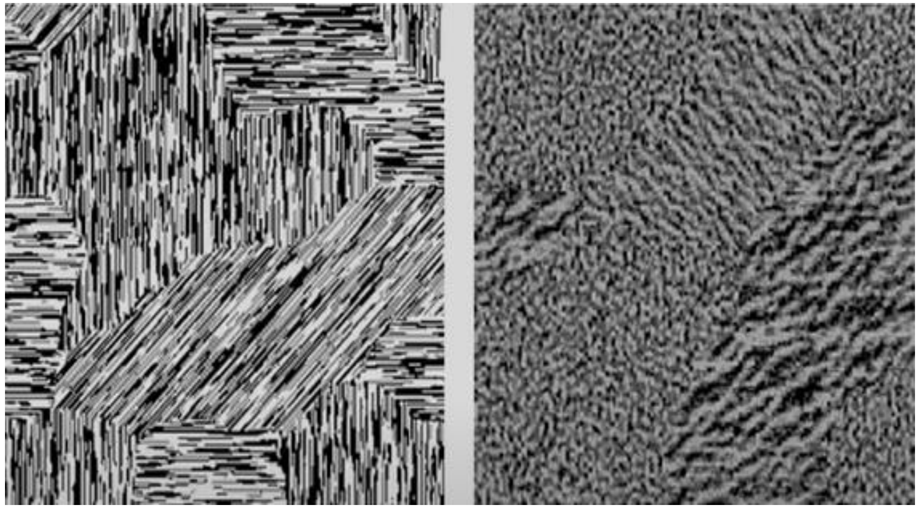


Image Texture

- Images can have same number of white and black pixels but different textures (distribution)
- Texture can be regular, smooth and coarse

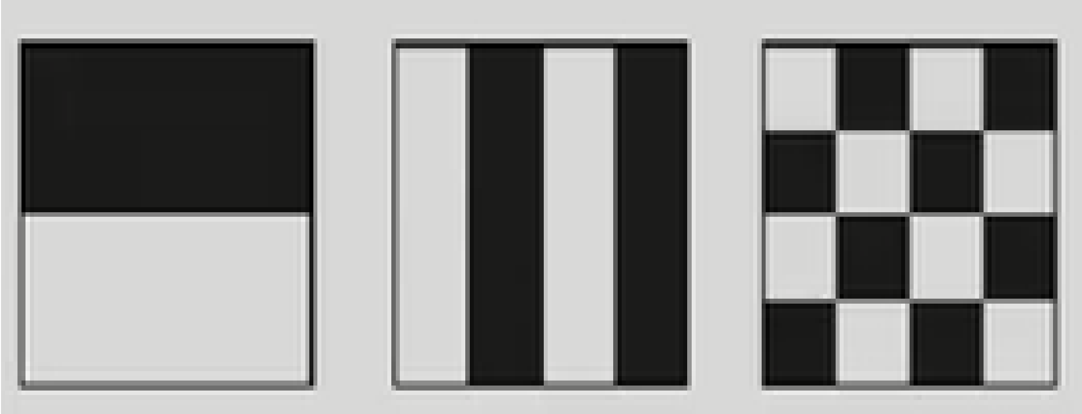


Image Texture

- Image with
  - Rough/ coarse texture has a large difference between neighboring pixel intensities
  - smooth texture has little difference between pixel intensities
- Textures in images quantify intensity differences in neighboring pixels
- First order texture measures are statistics of pixel values, like variance for entire image

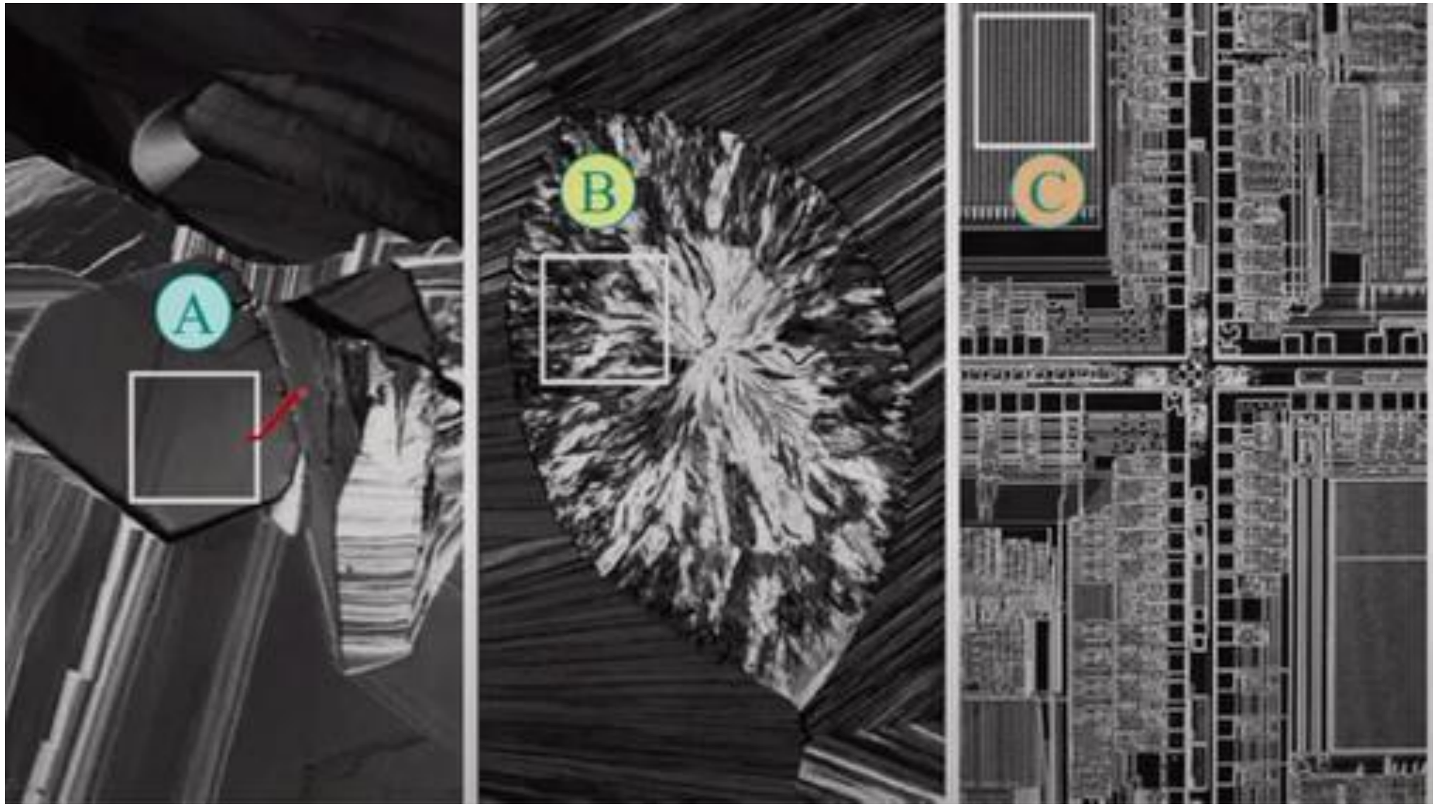
  and do not consider pixel relationships
- Second order texture features consider the relationship between two pixels in the original image. GLCM is an example

rough

smooth

Analysis of Texture

- Statistical features of image patch can be used to determine the texture



Smooth Coarse Regular

Texture	Standard Deviation
Smooth	11.79
Coarse	74.63
Regular	33.73

Image texture

- Broad applications are
  - Texture based segmentation
  - Texture synthesis
  - Texture transfer
  - Shape from texture
  - Spectral property

Ex: Texture based segmentation

- Extract texture features using Gabor filter, GLCM etc
- GLCM provides statistical features
- Gabor filter provides edge and texture features
- Based on features, classify different textures
- Partition into different regions where texture is homogeneous

Ex: Texture based segmentation

- Extract texture features using Gabor filter, GLCM etc
- GLCM provides statistical features
- Gabor filter provides edge and texture features
- Based on features, classify different textures
- Partition into regions where texture is homogeneous

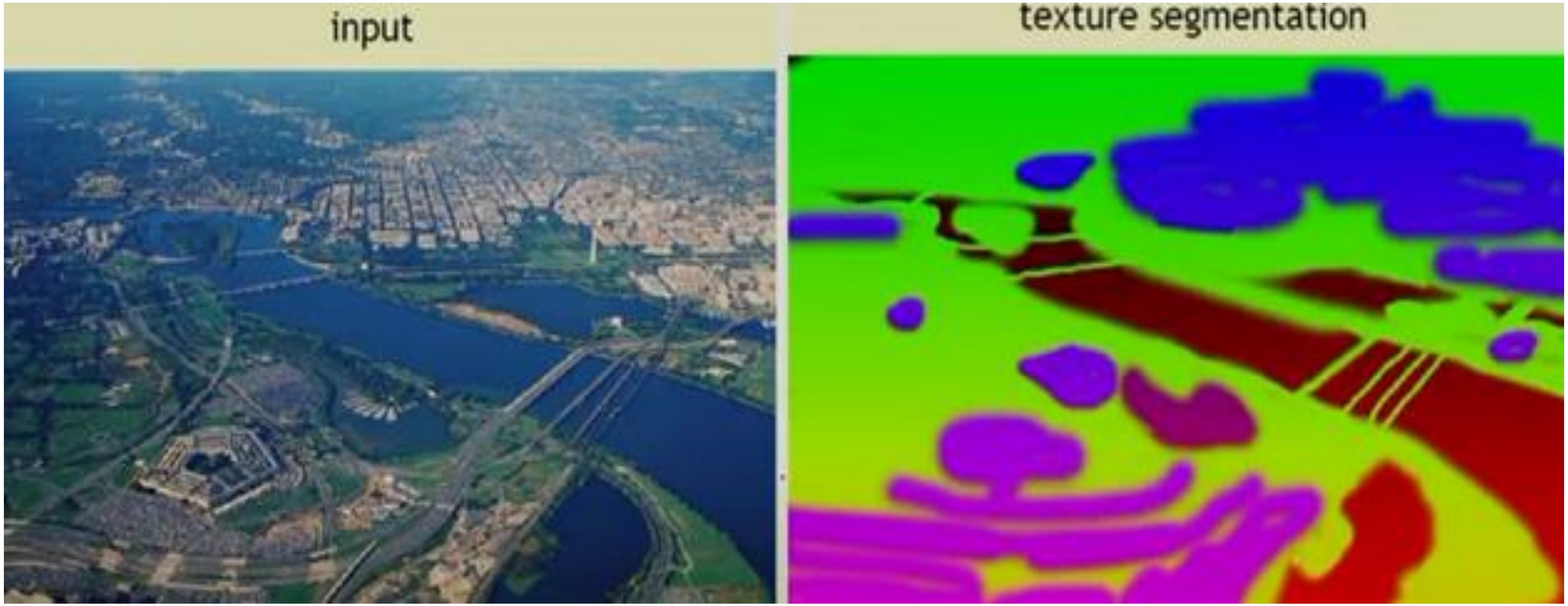


Image texture (Texture based Synthesis)

- Construct a large image from a small sample image by repeating texture

Image texture (Texture based Synthesis)

- Construct a large image from a small sample image by repeating texture

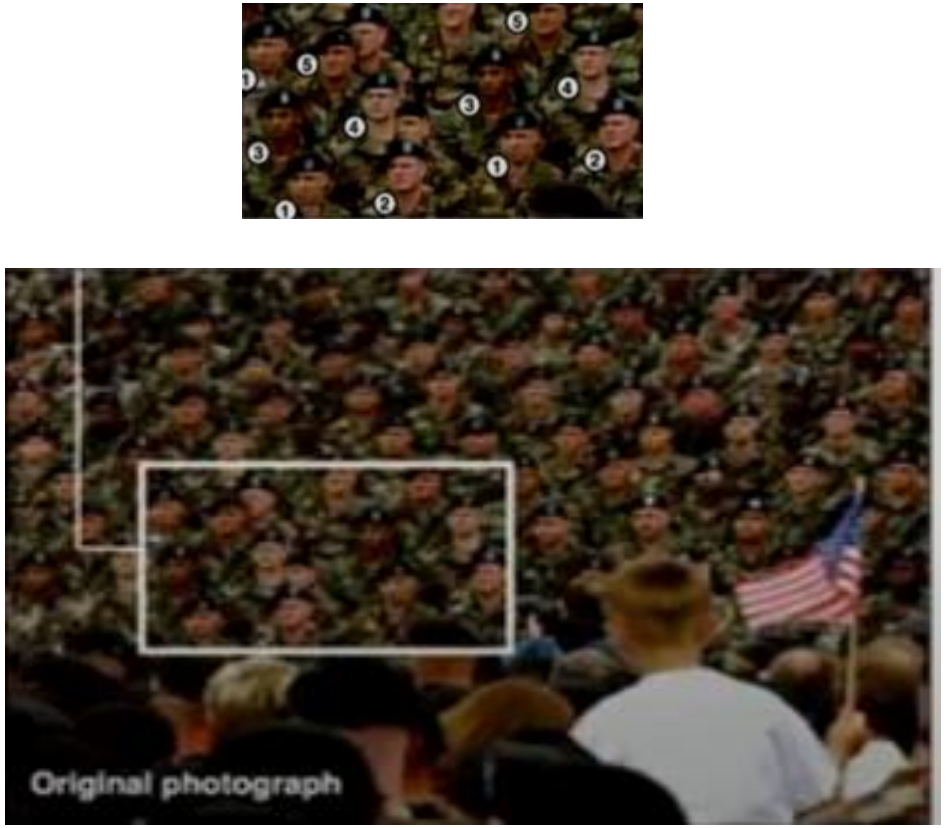


Image texture (Texture Transfer)

- Take a texture from one image and paint onto another image

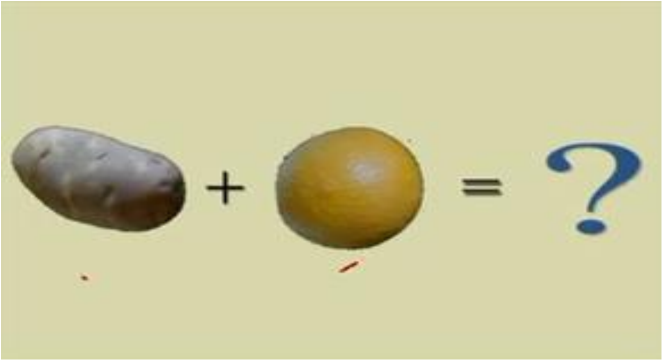
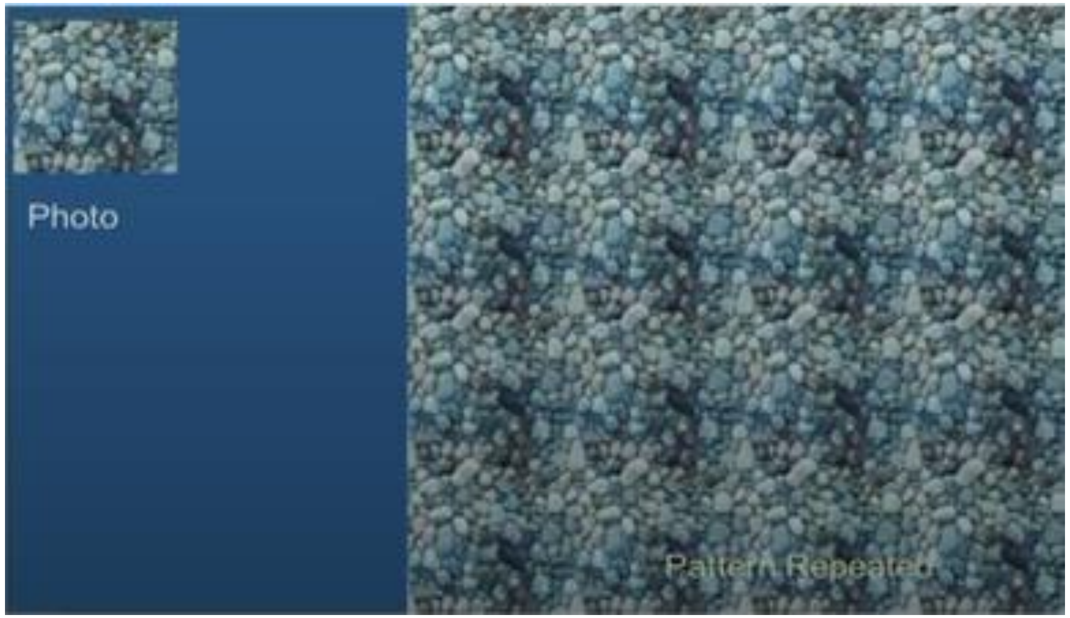


Image texture (Shape from Texture)

- Texture pattern variations can be used to estimate shape of a surface

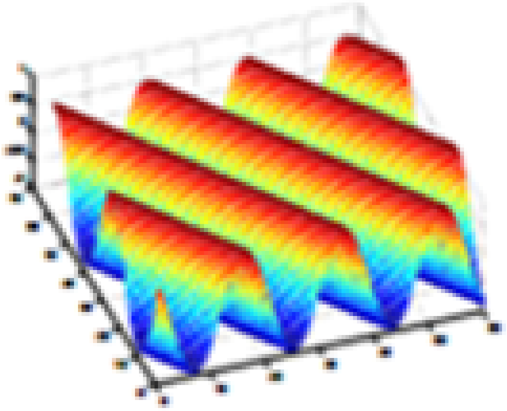


Spectral Property (Gabor Filter)

- Gabor filter is used to generate features that represent texture and edges
- Is a special class of bandpass filter
- Possess localization properties in both spatial and frequency domains
- Is a combination of Gaussian and Sinusoidal terms
- Sinusoidal signal of particular wavelength and orientation is modulated by a Gaussian wave
- Gaussian component provides weight and sine component provides directionality and repeatability of patterns
- Gabor kernel mimics human vision
- Mimics how human recognizes texture with eyes

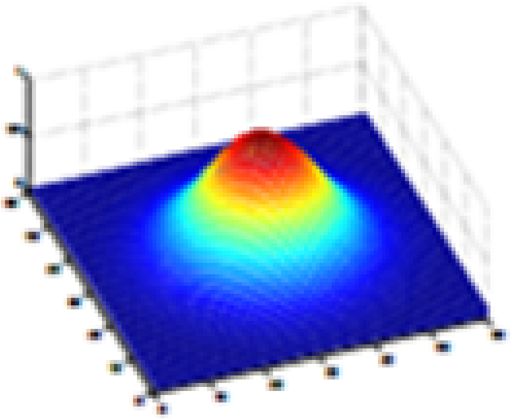
2D Gabor Filter

Visual color representation of Gabor filter (red is maximum and blue is minimum)



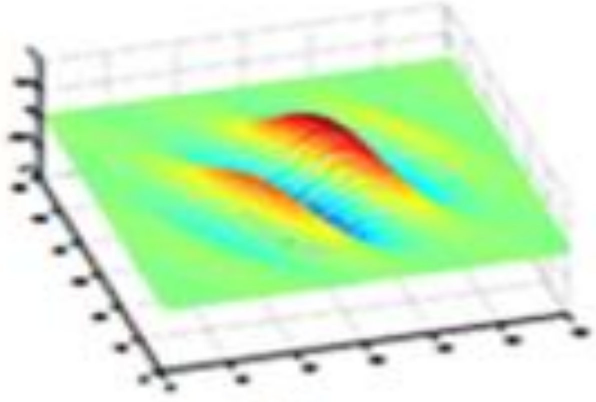
(a)

Sinusoid oriented 30^0 with X-axis



(b)

2D Gaussian



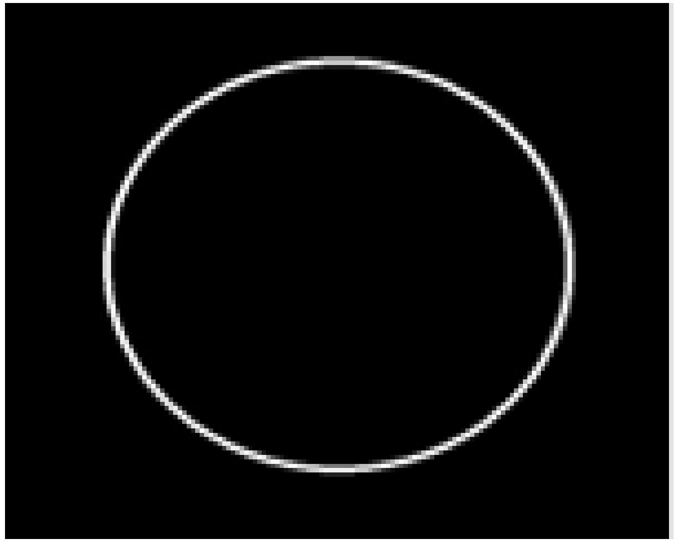
(c)

Gabor filter

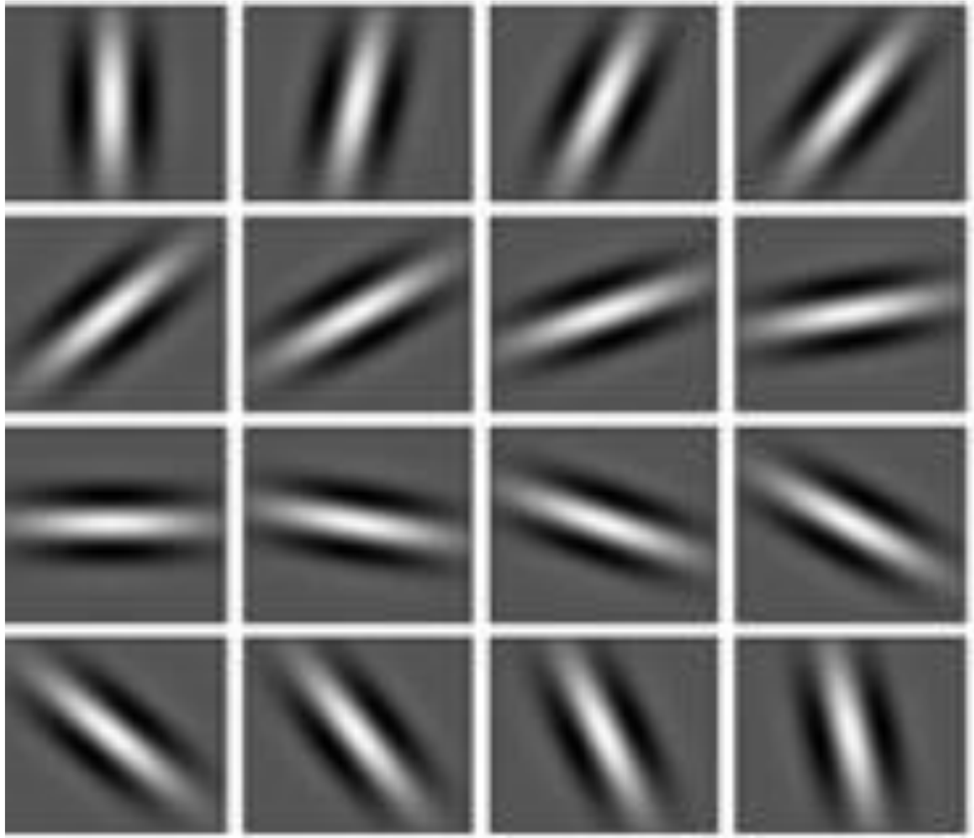
- Bank of Gabor filters are generated using number of different parameters of sinusoid and Gaussian
- Addition of filtered images is used to analyze texture feature of an image

2D Gabor Filter

- Image is passed through each filter of a bank of filters
- Angle of detected edge is the angle at which the Gabor filter is oriented

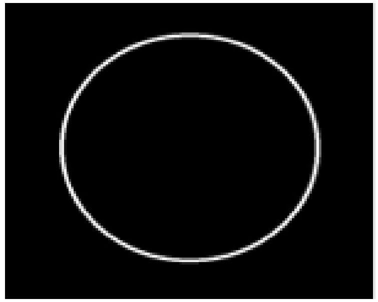


Image

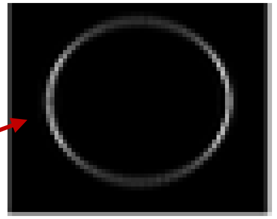


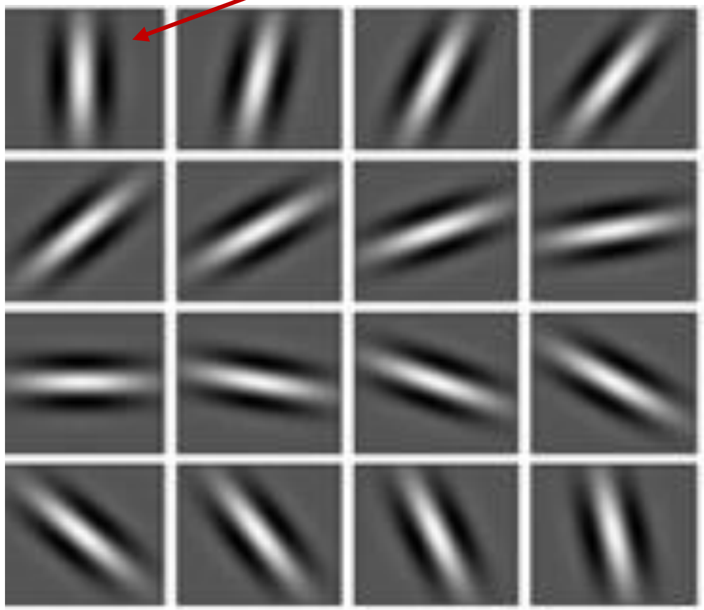
Visual representation of bank of 16 Gabor filters
white: maximum, Black: minimum

2D Gabor Filter



Image

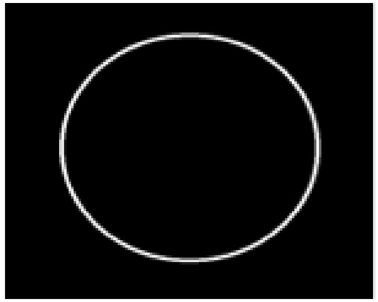




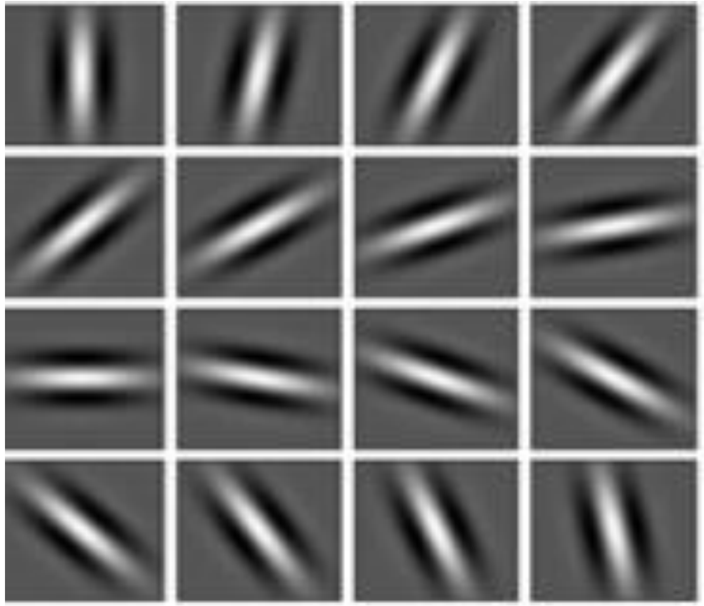
Bank of 16 Gabor filters
white: maximum, Black: minimum

Output of 16 Gabor filters

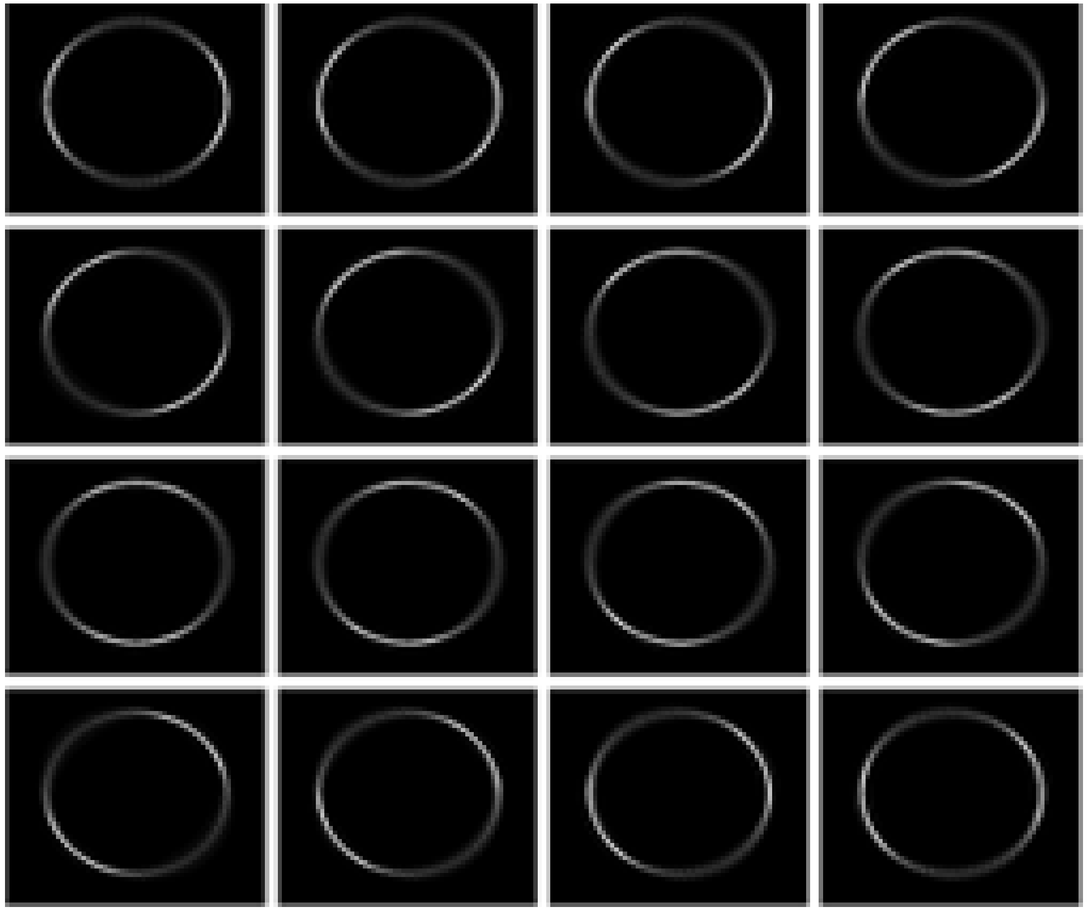
2D Gabor Filter



Image

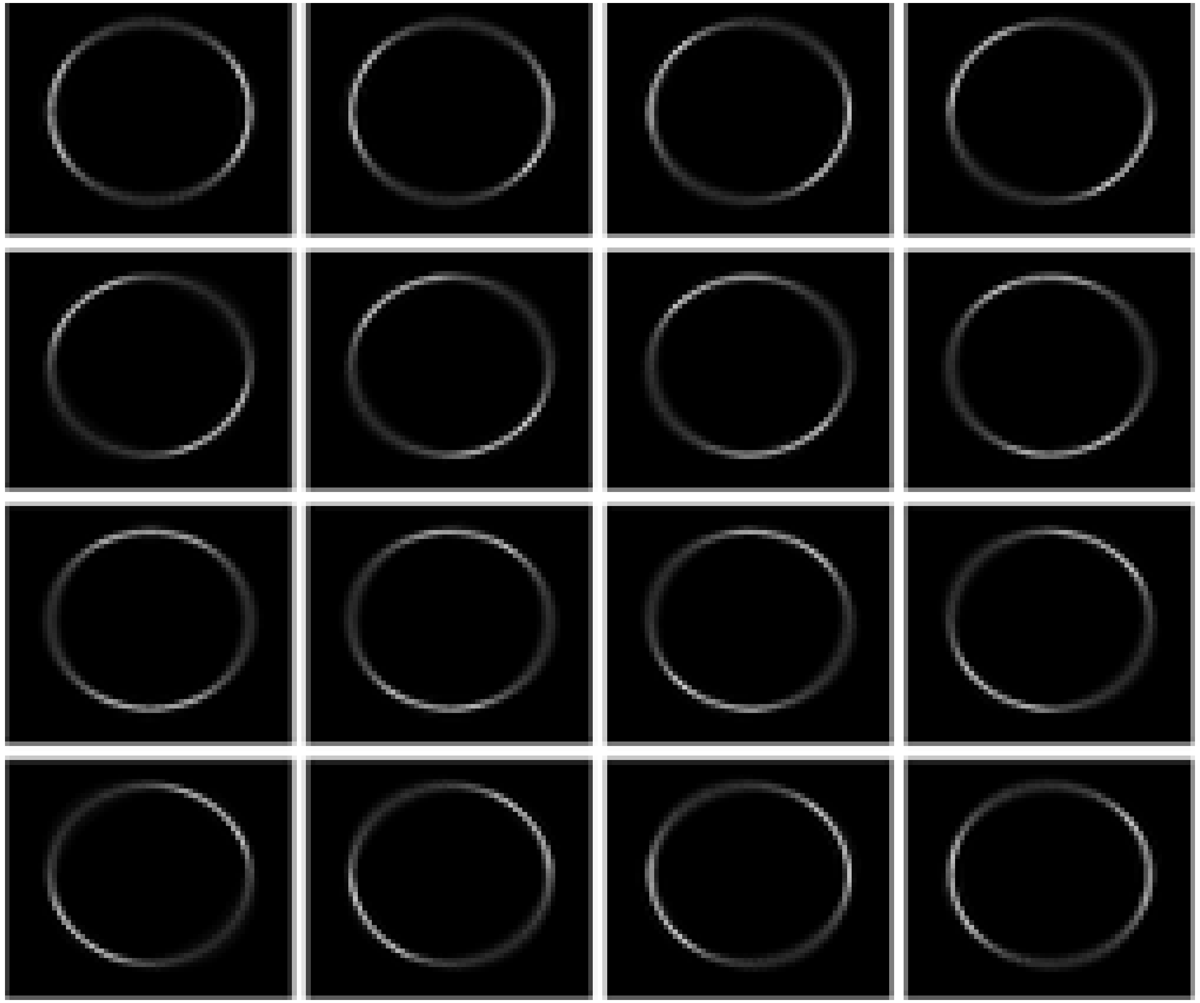


Bank of 16 Gabor filters
white: maximum, Black: minimum

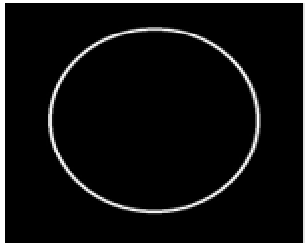


Output of 16 Gabor filters

2D Gabor Filter



Output of 16 Gabor filters

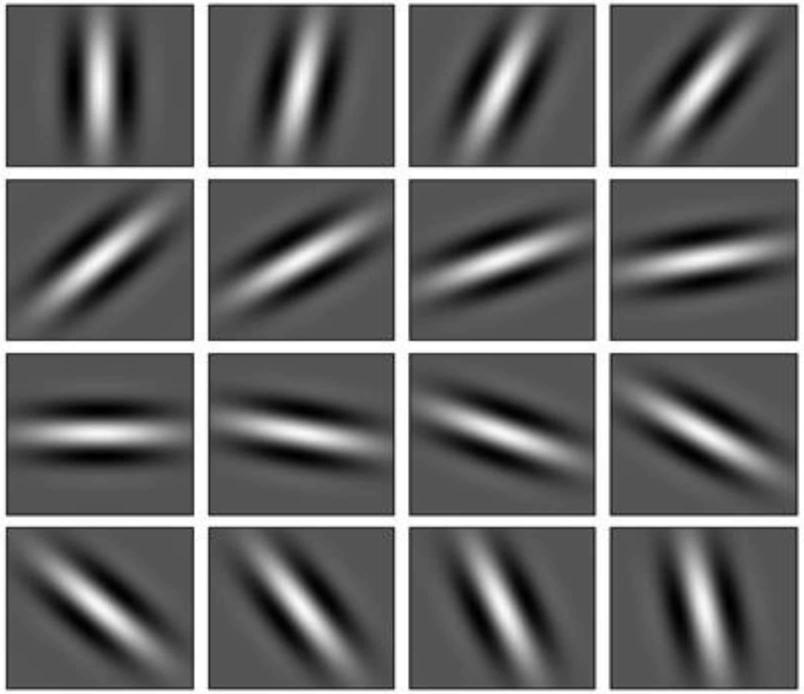


Sum of 16 outputs

2D Gabor Filter: application

- Highlight or extract stripes of an image
- Use a bank of 16 Gabor filters at an orientation every 11.25 (for total of 180^0)
- Filter image with Gabor filters
- Each filter highlights stripe of a specific orientation

Stripes on skin are at different orientations

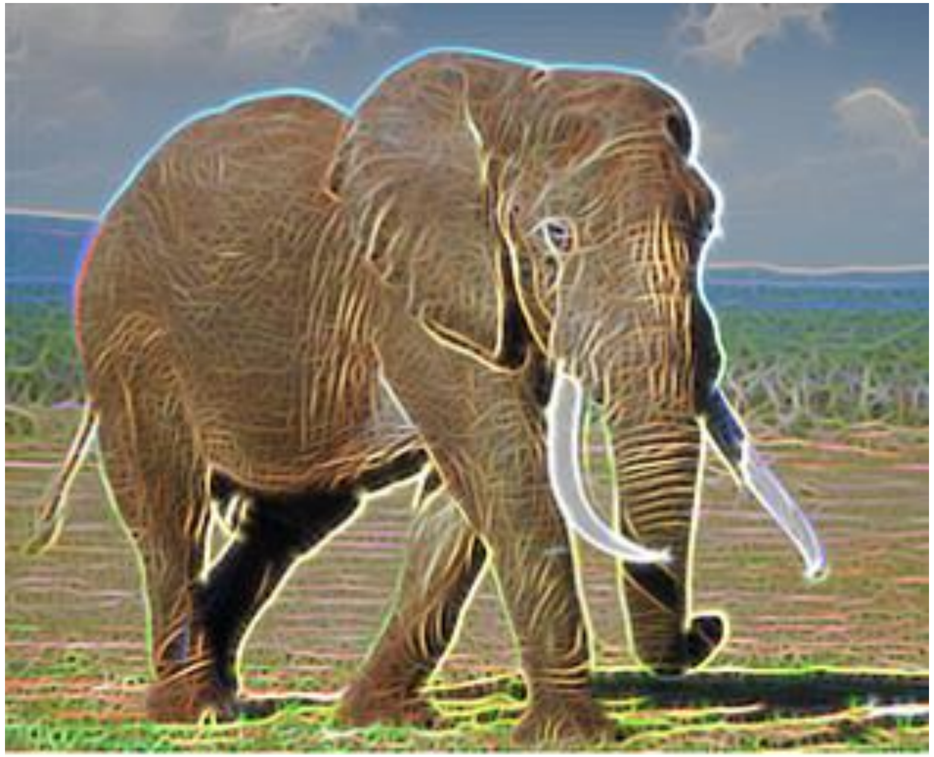


16 Gabor filters

2D Gabor Filter

- Response of image to each Gabor filter shows maximum at edges and at points where texture changes

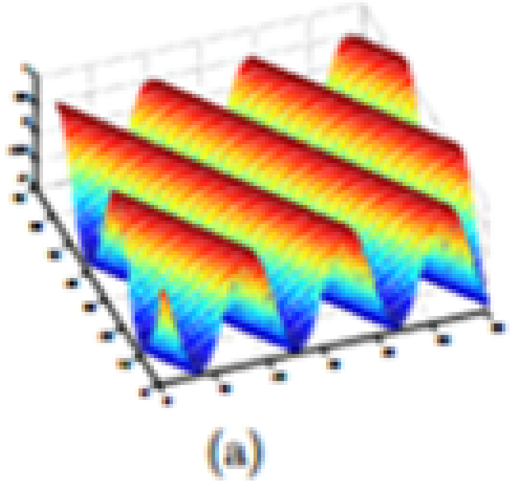
Image



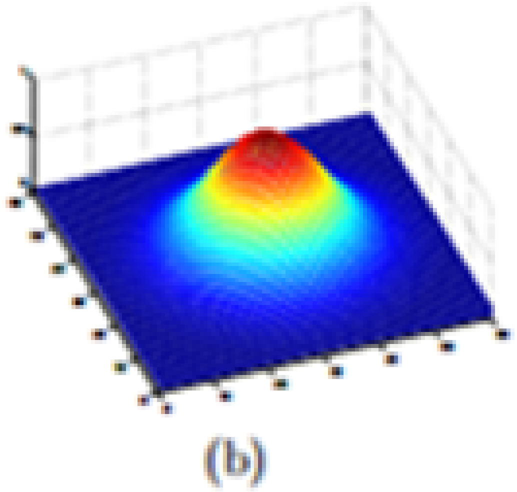
Filtered Image

2D Gabor Filter

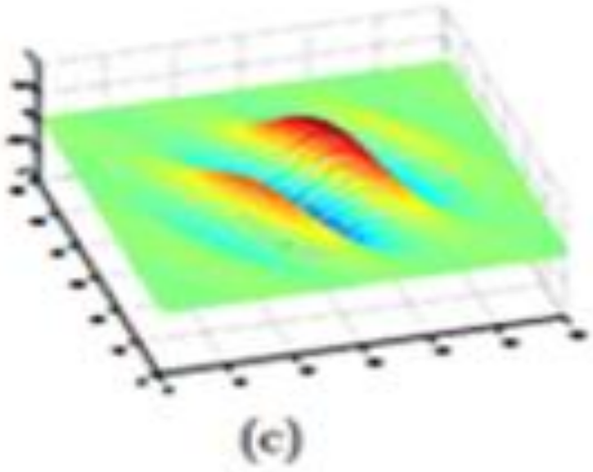
Visual color representation of Gabor filter (red is maximum and blue is minimum)



Sinusoid oriented 30^0 with X-axis



2D Gaussian



Gabor filter

Parameters of 2D Gabor filter

- A 2D Gabor filter can be viewed as a sinusoidal signal of particular frequency and orientation, modulated by a Gaussian wave
- Depends on

λ — Wavelength of the sinusoidal component

ϴ — Orientation of the normal to the parallel stripes of the Gabor function

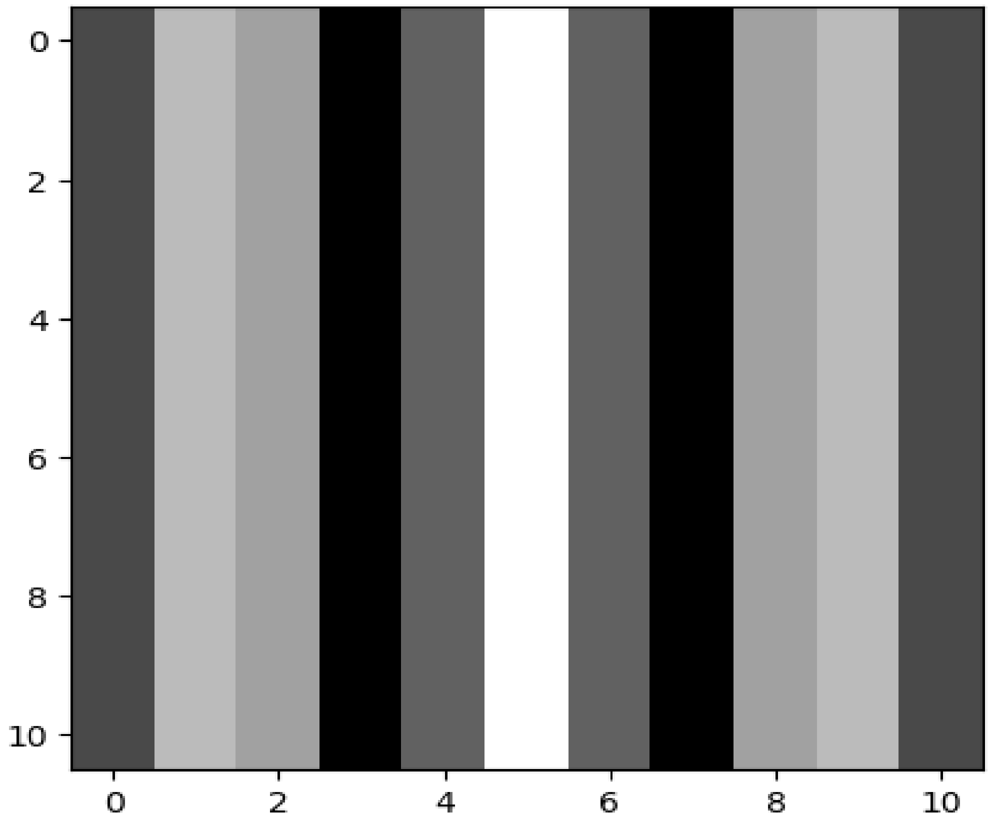
Ψ — Phase offset of the sinusoidal function

σ — Standard deviation of the Gaussian envelope

ϒ — The spatial aspect ratio and specifies the ellipticity of the support of the Gabor function

Ex: 2D Gabor filter

- Size 11x11
- Sigma=4, theta=0, lambda=pi/4, gamma=0



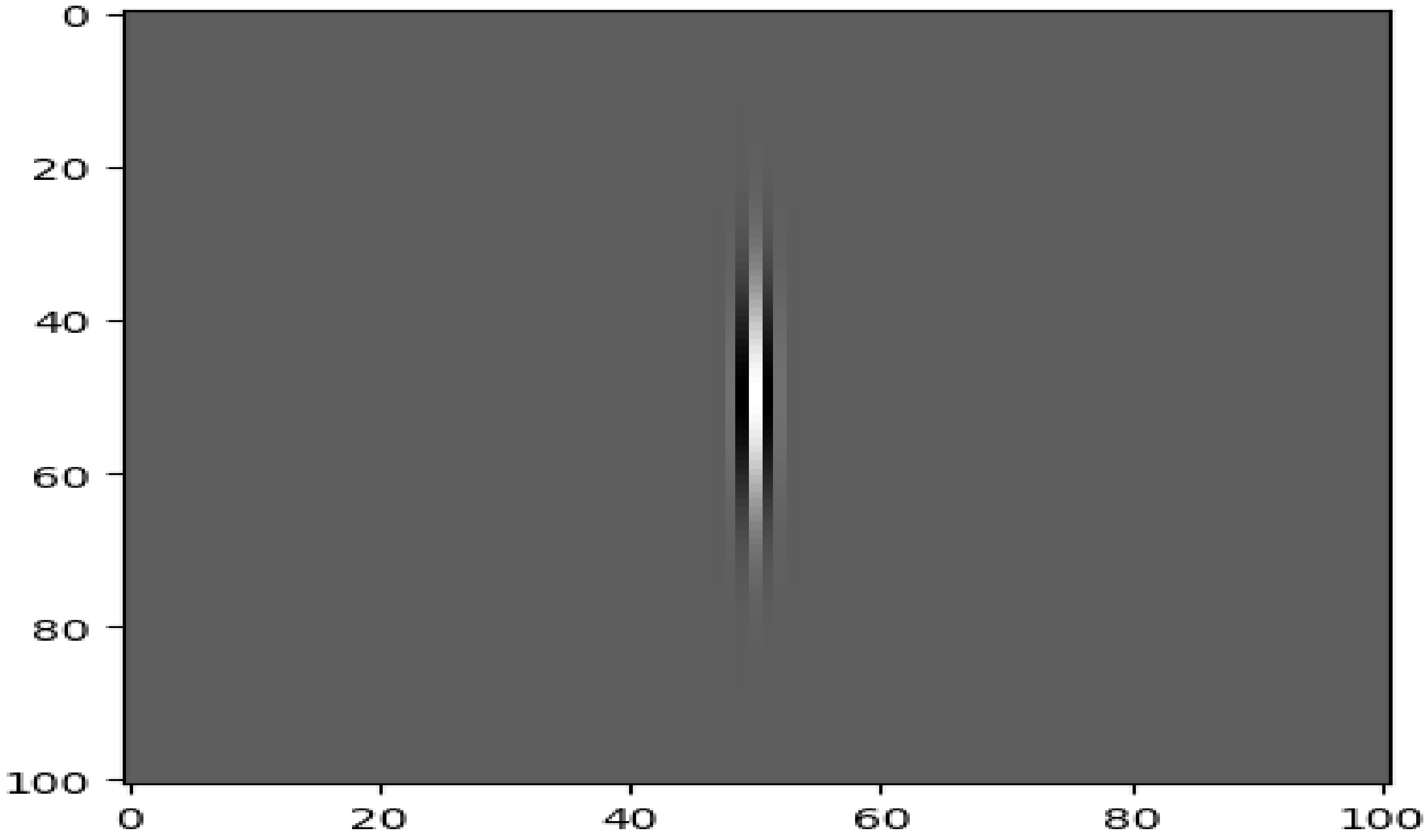
Representation of Filter

- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3
- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3
- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3
- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3
- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3
- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3
- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3
- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3
- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3
- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3
- 0.3	0.5	0.3	- 0.8	- 0.14	1.0	- 0.14	- 0.8	0.3	0.5	- 0.3

Filter with element values

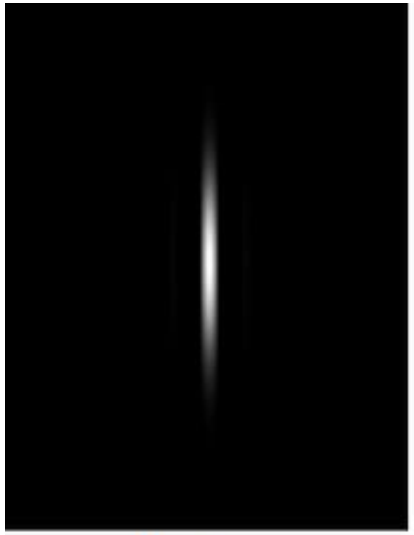
Ex: 2D Gabor filter

- Size 101x101
- Sigma=1, theta=0, lambda=pi/8, gamma=0.1



Generation of 2D Gabor filters

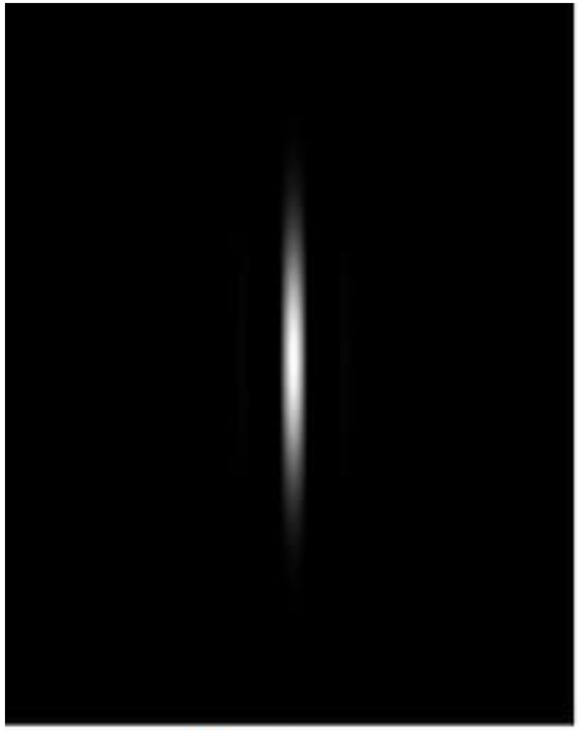
- Parameters control the shape and size of the Gabor function
- Example: $\Theta = 0$, $\sigma = 10$
- Lambda (λ) controls width of the strips of the Gabor function
  - Increasing the wavelength produces thicker stripes



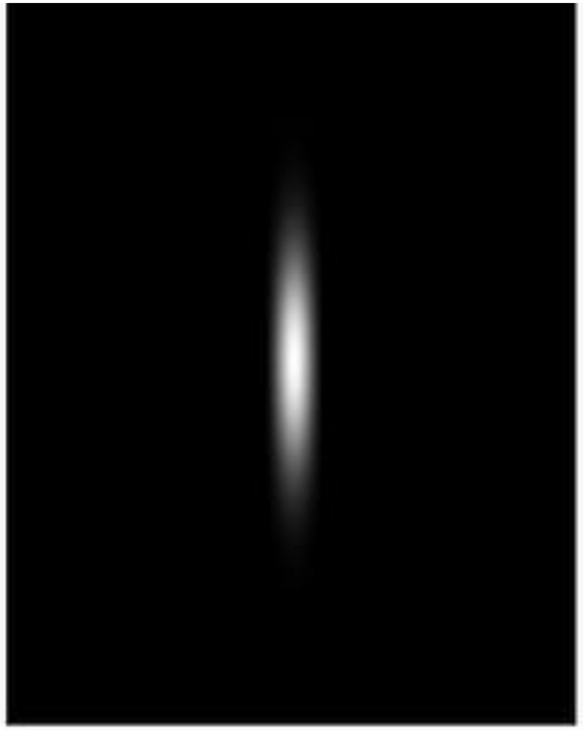
Lambda (λ) = 30

λ — Wavelength of the sinusoidal component
Θ — The orientation of the normal to the parallel stripes of the Gabor function
Ψ — The phase offset of the sinusoidal function
σ — The sigma/standard deviation of the Gaussian envelope

Generation of 2D Gabor filters

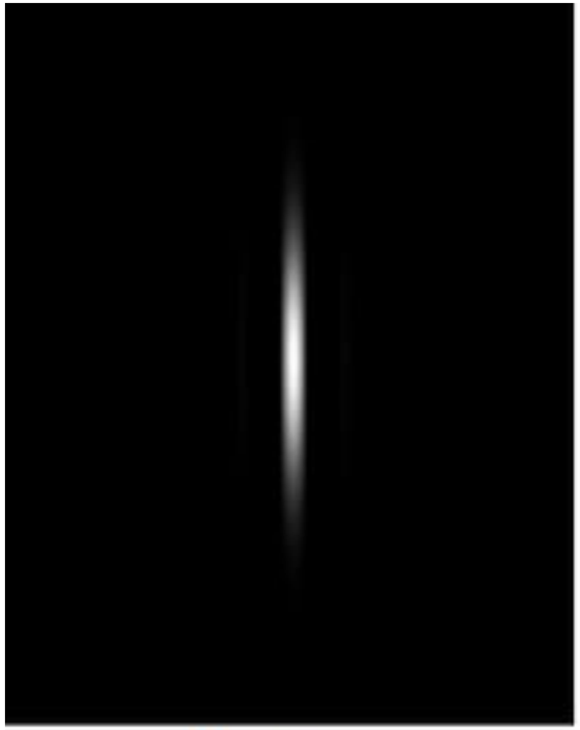


Lambda (λ) = 30

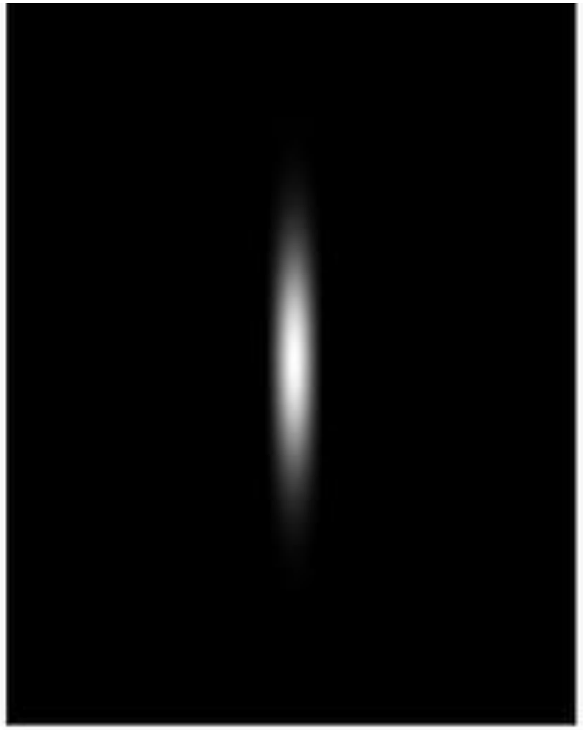


Lambda (λ) = 60

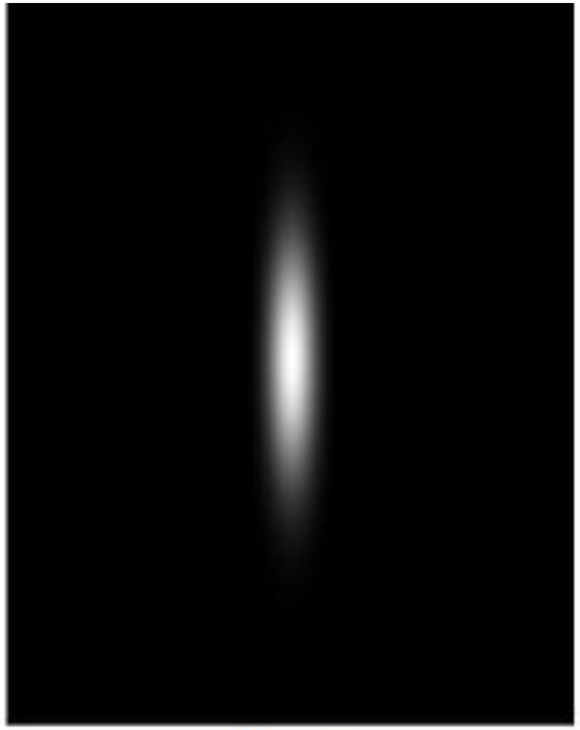
Generation of 2D Gabor filters



Lambda (λ) = 30

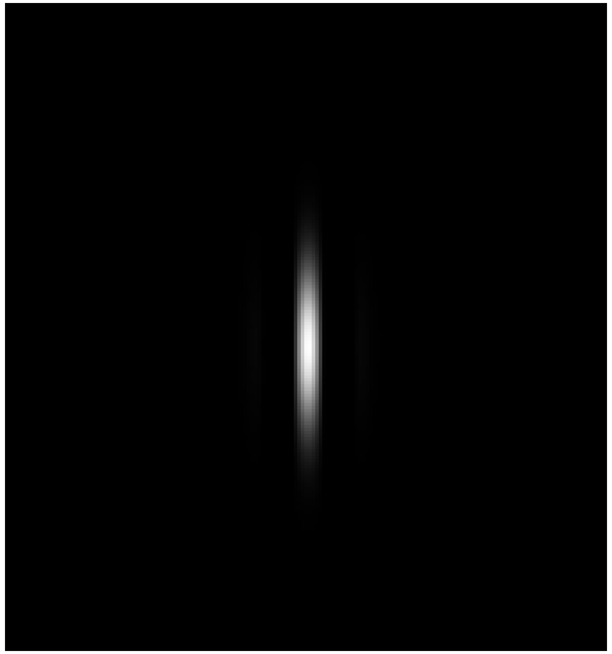


Lambda (λ) = 60



Lambda (λ) = 100

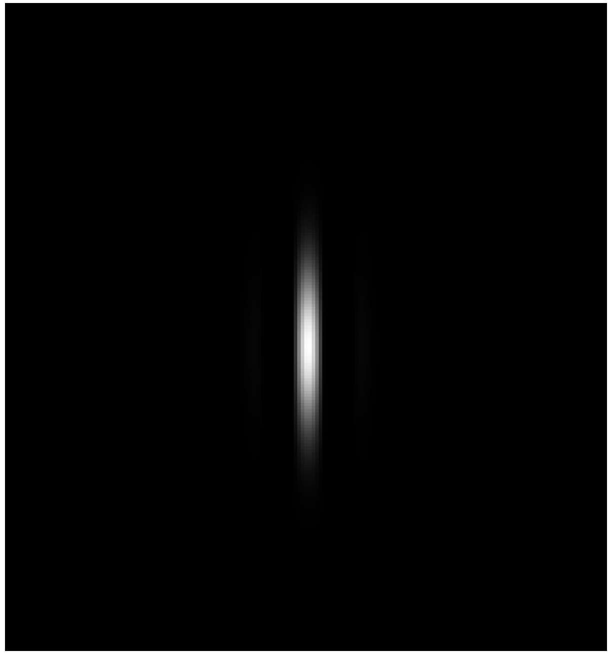
Generation of 2D Gabor filters



Theta (Θ) = 0

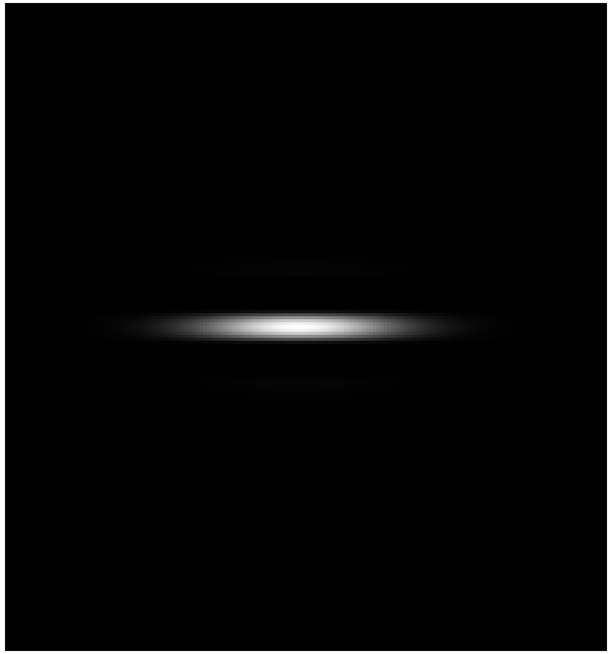
Theta (Θ), Orientation with respect to the normal

Generation of 2D Gabor filters



Theta (Θ) = 0

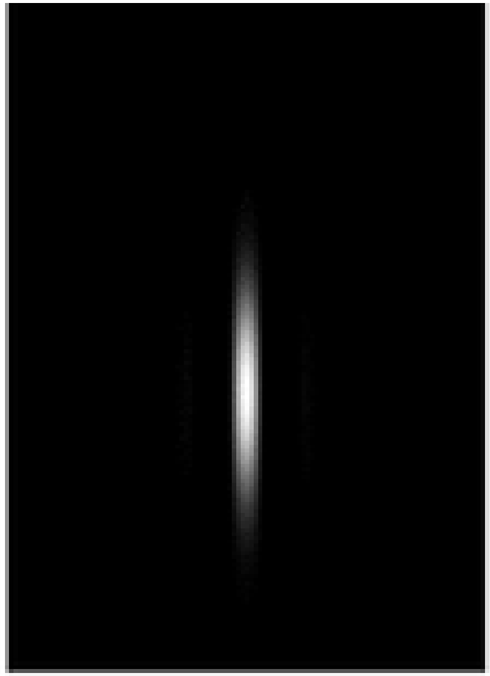
Theta (Θ) = 45



Theta (Θ) = 90

Theta (Θ), Orientation with respect to the normal

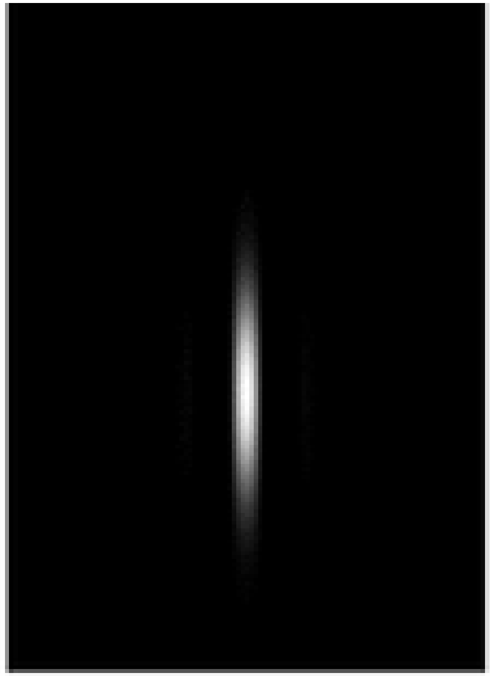
Generation of 2D Gabor filters



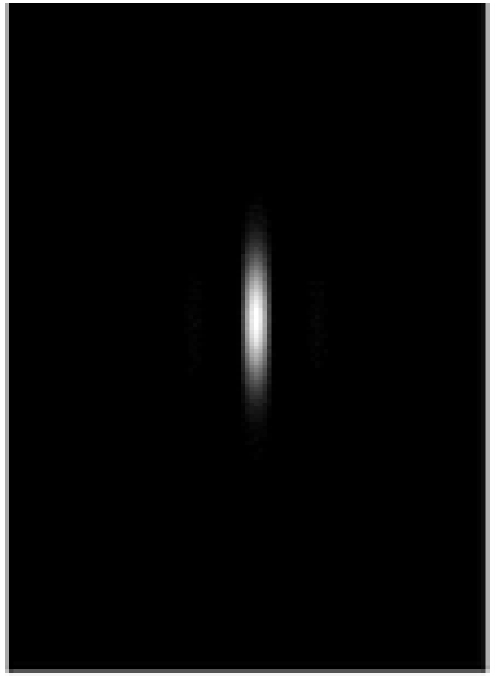
Gamma (ɣ) = 0.25

Gamma (ɣ)
- Specifies aspect ratio of Gabor function
- Shows elipticity of curve

Generation of 2D Gabor filters



Gamma (ɣ) = 0.25

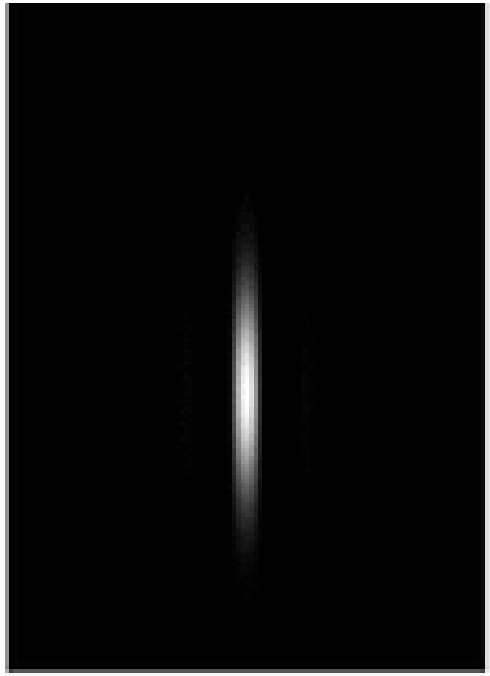


Gamma (ɣ) = 0.5

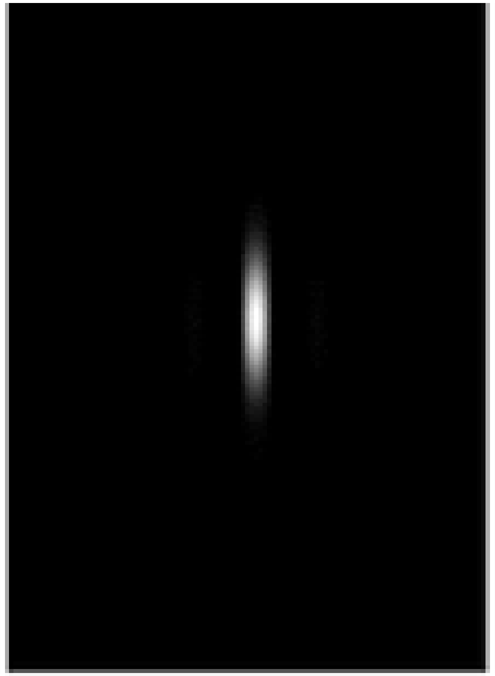
Gamma (ɣ)

- Specifies aspect ratio of Gabor function
- Shows elipticity of curve

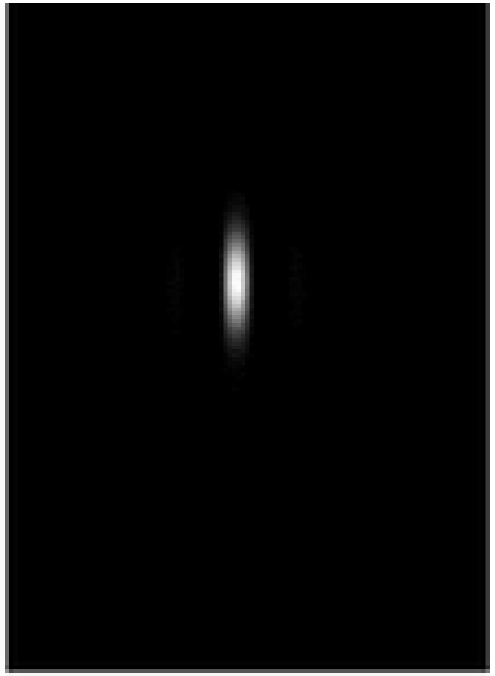
Generation of 2D Gabor filters



Gamma (γ) = 0.25



Gamma (γ) = 0.5



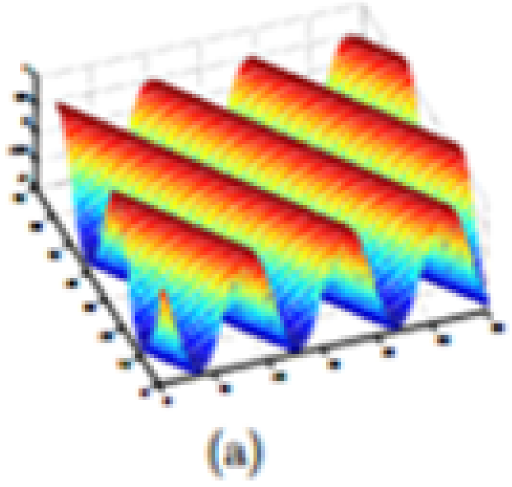
Gamma (γ) = 0.75

Gamma (ɣ)

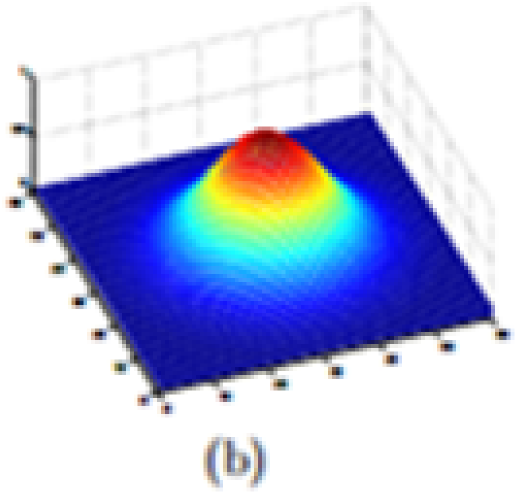
- Specifies aspect ratio of Gabor function
- Shows elipticity of curve

2D Gabor Filter

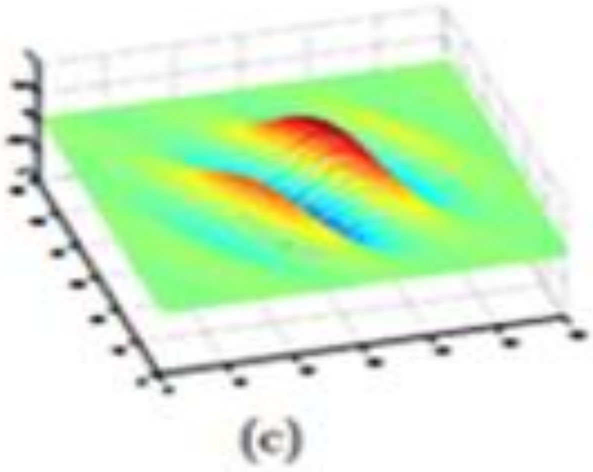
Visual color representation of filter (red is maximum and blue is minimum)



Sinusoid



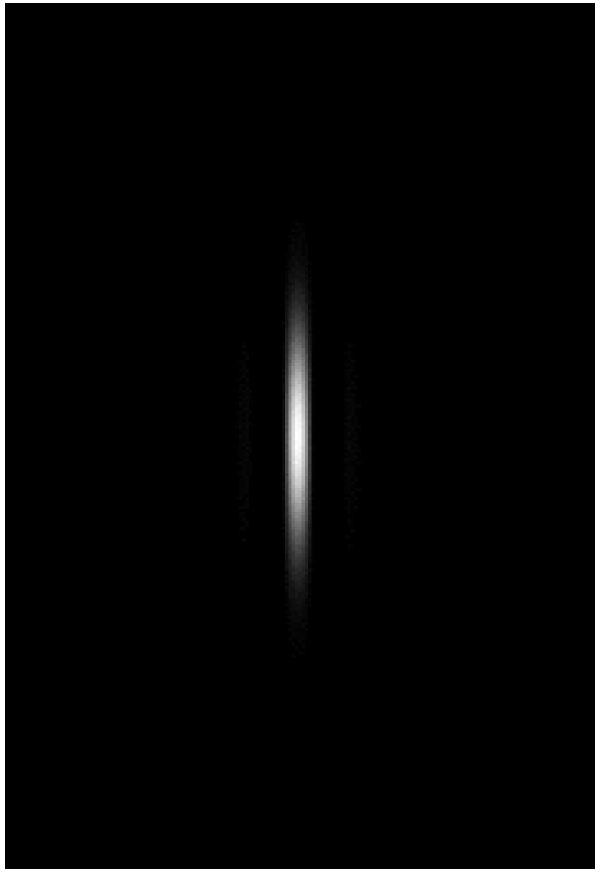
2D Gaussian with std, σ



Gabor filter

- Bandwidth or sigma controls the overall size of the Gabor envelope
- Large bandwidth of the envelope allows more stripes

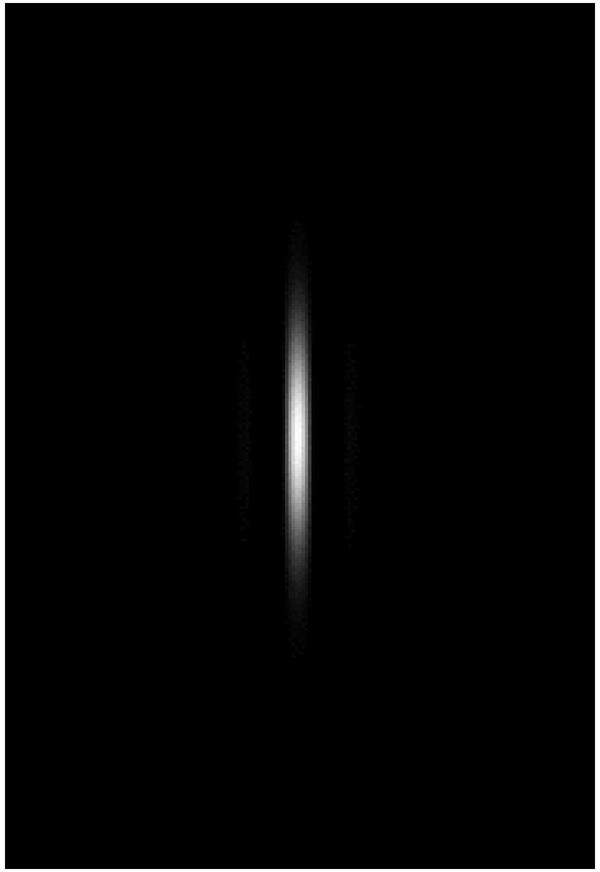
Generation of 2D Gabor filters



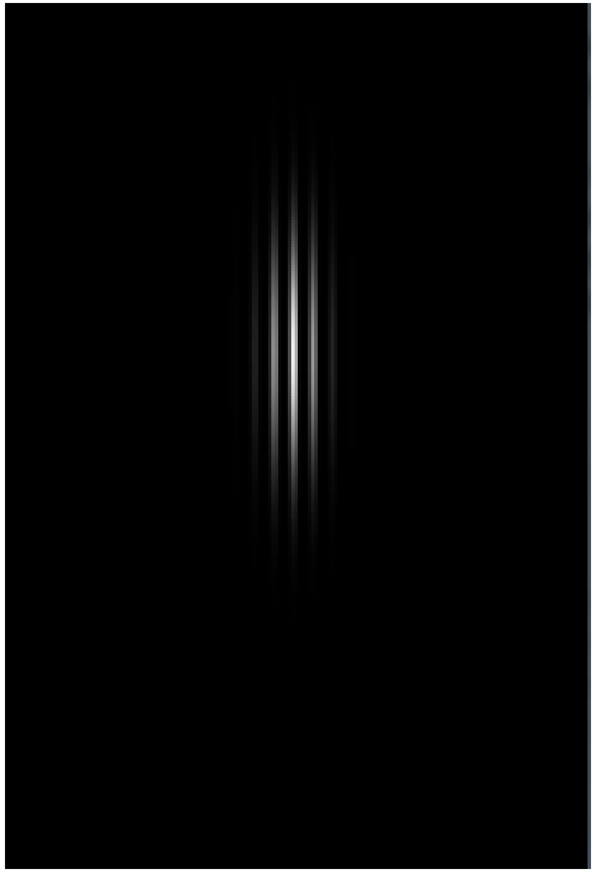
Sigma (σ) = 10

Large bandwidth of the envelope allows more stripes

Generation of 2D Gabor filters



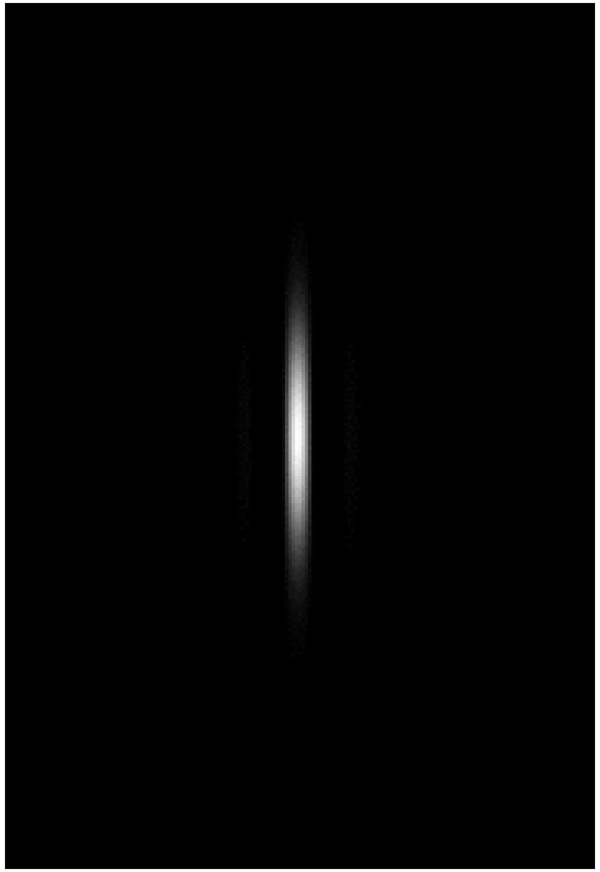
Sigma (σ) = 10



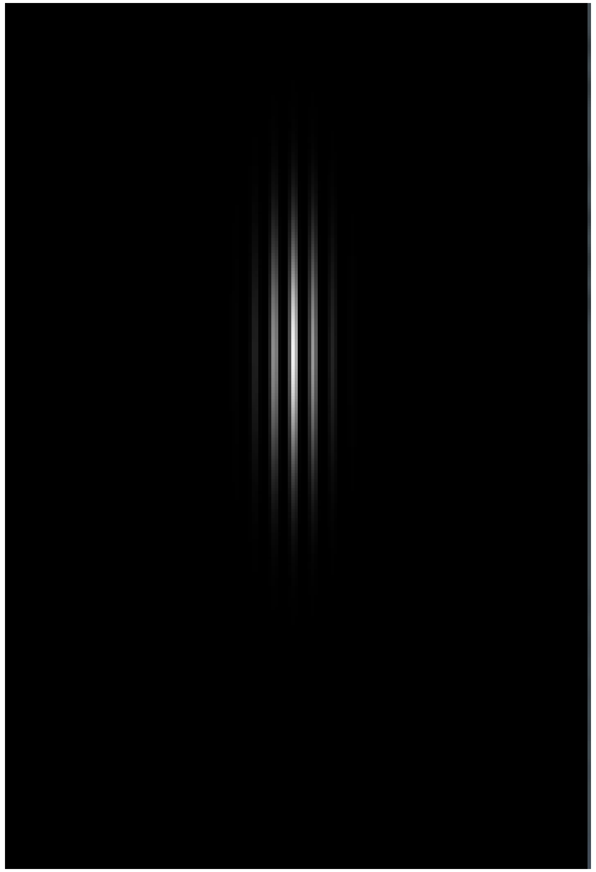
Sigma (σ) = 30

Large bandwidth of the envelope allows more stripes

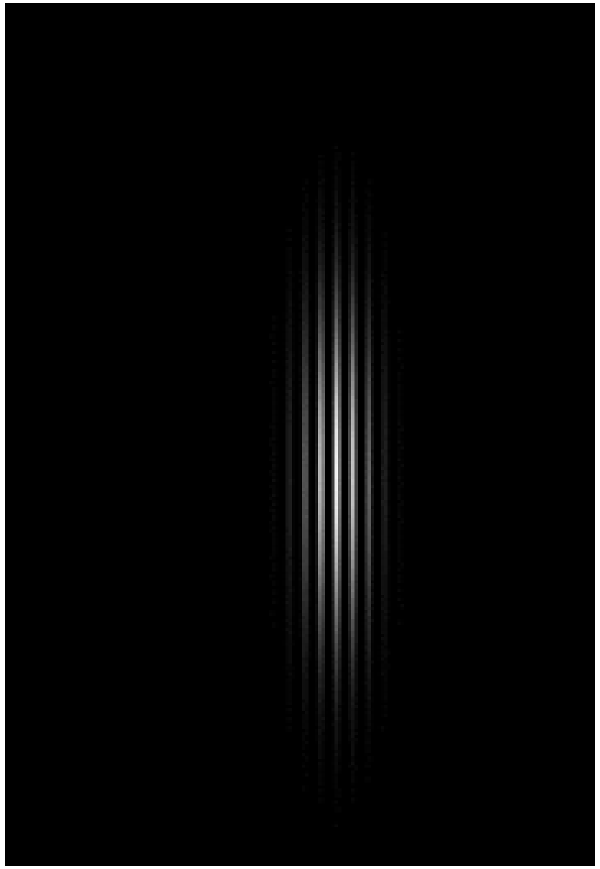
Generation of 2D Gabor filters



Sigma (σ) = 10



Sigma (σ) = 30



Sigma (σ) = 45

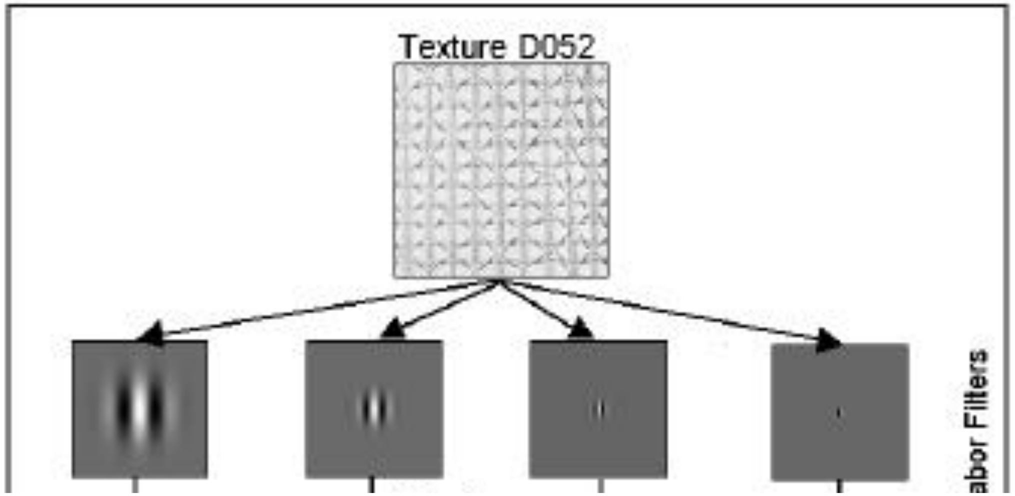
Large bandwidth of the envelope allows more stripes

2D Gabor Filter

- Image is passed through each filter of the filter bank
- Filtered outputs are added to generate final output
- Edge which gets detected is the angle at which the Gabor filter is oriented

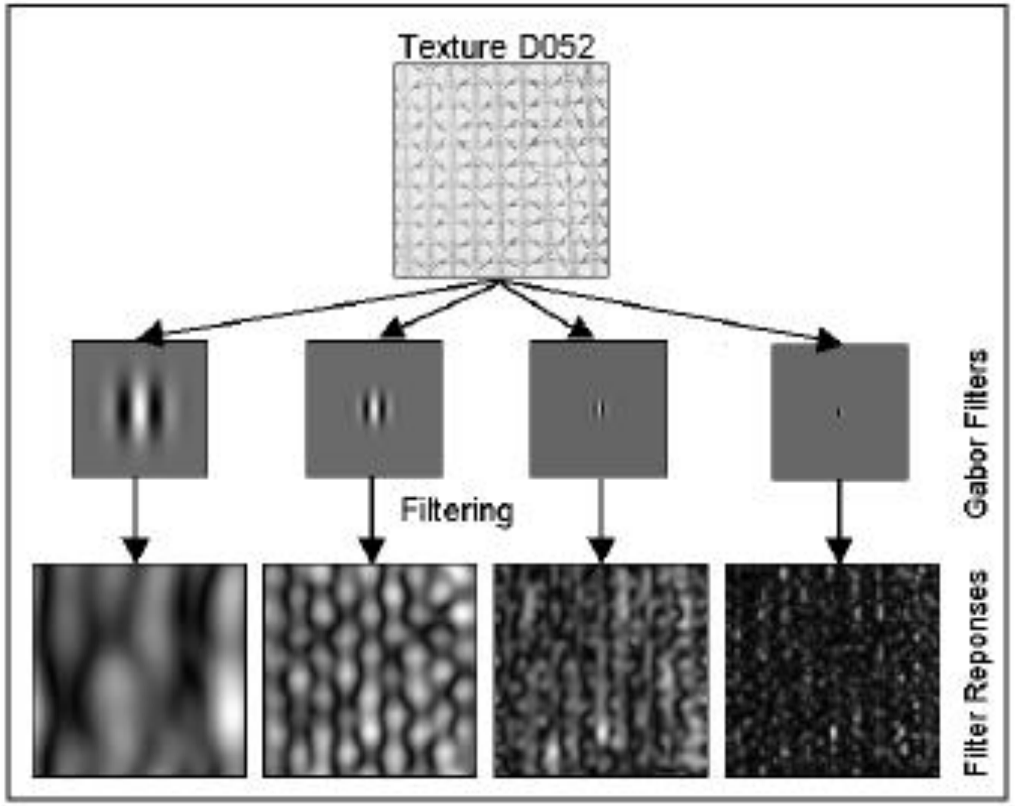
2D Gabor Filter

- Image is passed through each filter of the filter bank
- Filtered outputs are added to generate final output
- Edge which gets detected is the angle at which the Gabor filter is oriented



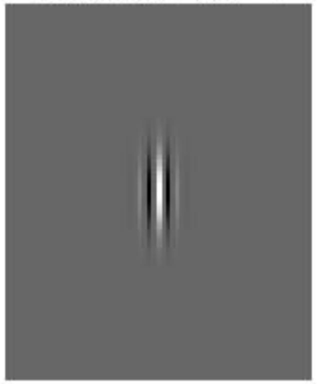
2D Gabor Filter

- Image is passed through each filter of the filter bank
- Filtered outputs are added to generate final output
- Edge which gets detected is the angle at which the Gabor filter is oriented

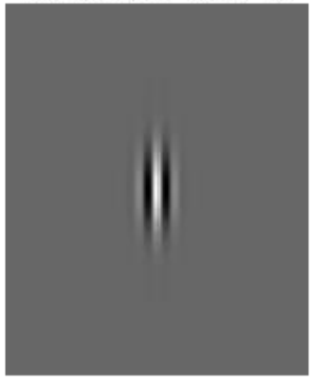


Response of Gabor filter

Re[g(x,y)], $\theta = 0$, $\lambda = 4$



Re[g(x,y)], $\theta = 0$, $\lambda = 8$



=

$\theta = 0$, $\lambda = 4$

$\theta = 0$, $\lambda = 8$

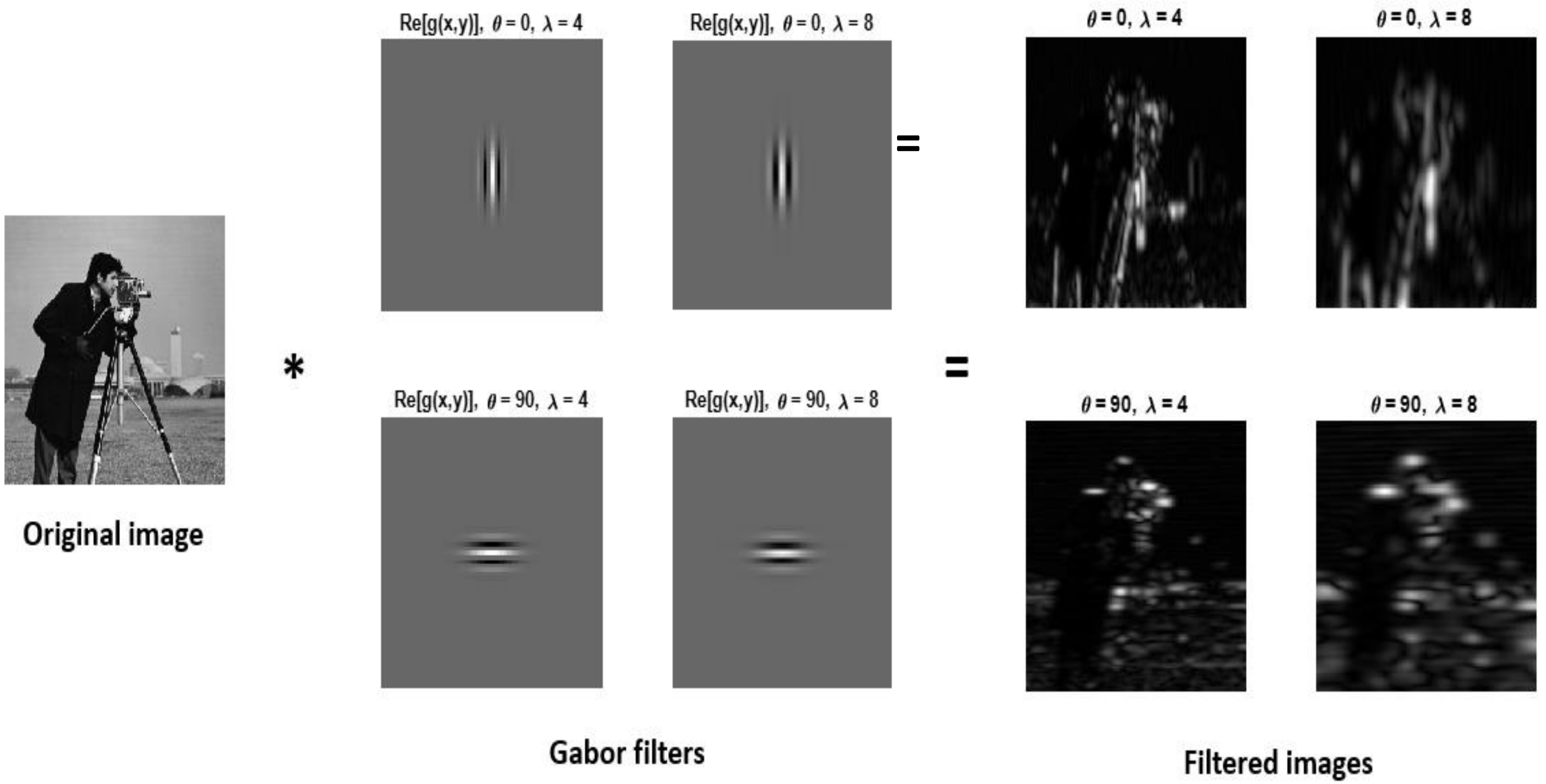
*

Original image

Gabor filters

Filtered images

Response of Gabor filter



Parameters of 2D Gabor filter

- A 2D Gabor filter can be represented by real and an imaginary components

Complex

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

λ — Wavelength of the sinusoidal component
Θ — Orientation of the normal to the parallel stripes of the Gabor function
Ψ — The phase offset of the sinusoidal function
σ — The sigma/standard deviation of the Gaussian envelope
ϒ — The spatial aspect ratio and specifies the ellipticity
X and **y** are pixel locations

Parameters of 2D Gabor filter

- A 2D Gabor filter can be represented by real and an imaginary components

Complex

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

Real

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

Imaginary

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

λ — Wavelength of the sinusoidal component
Θ — Orientation of the normal to the parallel stripes of the Gabor function
Ψ — The phase offset of the sinusoidal function
σ — The sigma/standard deviation of the Gaussian envelope
ϒ — The spatial aspect ratio and specifies the ellipticity
X and **y** are pixel locations

Parameters of 2D Gabor filter

- A 2D Gabor filter can be represented by real and an imaginary components

Complex

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

Real

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

Imaginary

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

where

$$x'=x\cos\theta+y\sin\theta$$

and

$$y'=-x\sin\theta+y\cos\theta$$

λ — Wavelength of the sinusoidal component
ϴ — The orientation of the normal to the parallel stripes of the Gabor function
Ψ — The phase offset of the sinusoidal function
σ — The sigma/standard deviation of the Gaussian envelope
ϒ — The spatial aspect ratio and specifies the ellipticity
X and **y** are pixel values

Generate features using Gabor filters

- Determine real or the imaginary part of the filtered image
  - Phase of the response represents orientation of edges
  - Amplitude of the response represents strength of edge
  - Magnitudes of responses with the same orientation can be taken as a feature vector
- Apply PCA on bank of filters to reduce number of filter elements (dimensions) and train model using filtered image

Ex: Extract features with Gabor filters

170	245	0	74	149
234	42	64	138	160
32	53	128	202	224
96	117	192	213	21
106	181	255	10	85

Image

-1	0	1
2	1	2
1	-2	0

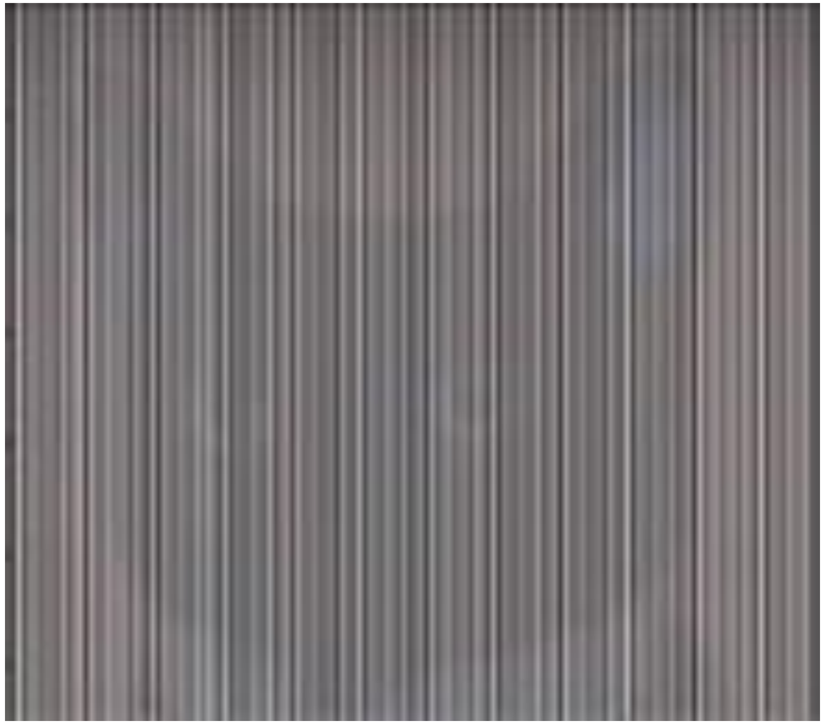
Kernel of Gabor filter

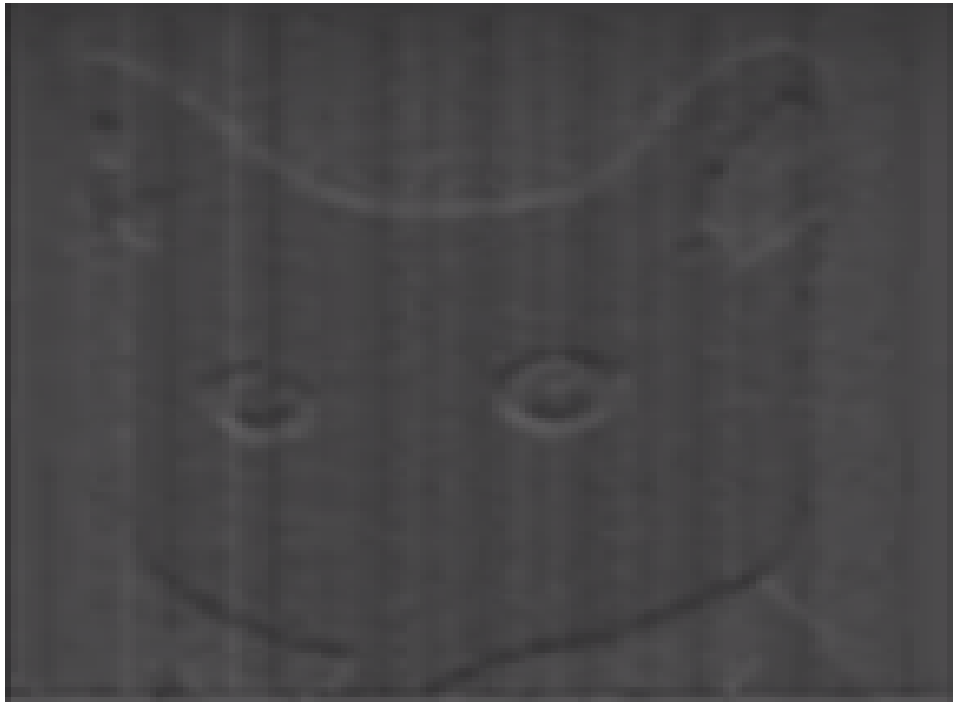
	?	?	?	
	?	?	?	
	?	?	?	

Filtered image

Extract features with Gabor filters

- Selection of correct combination of filters is important
- A particular combination suits the requirement
- To remove vertical stripes, apply horizontal Gabor filter





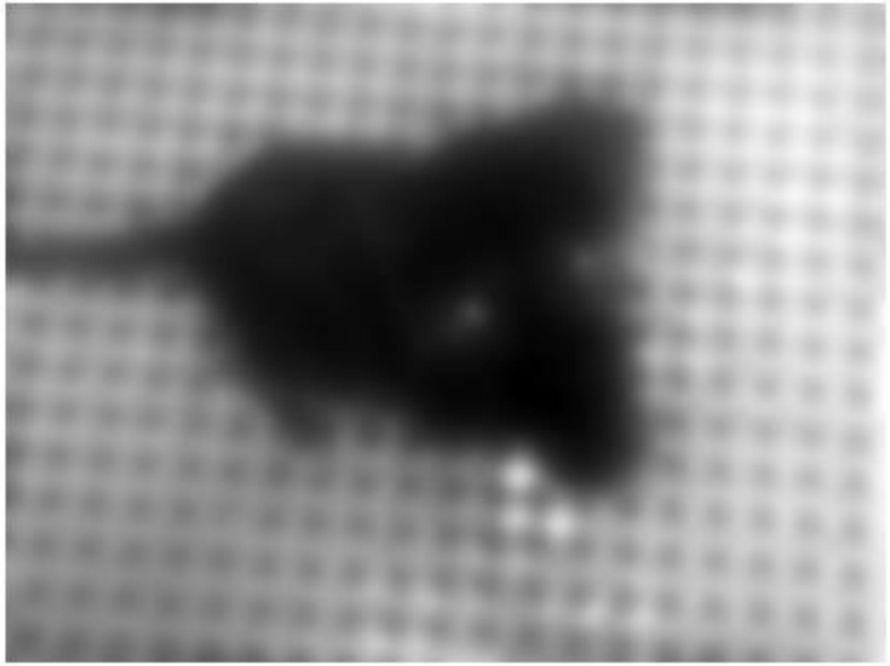
Texture Segmentation using Gabor filters

- Texture of regular and periodic pattern of the floor is different from smooth texture of the dog's fur
- Segment dog from the bathroom floor

Texture Segmentation using Gabor filters

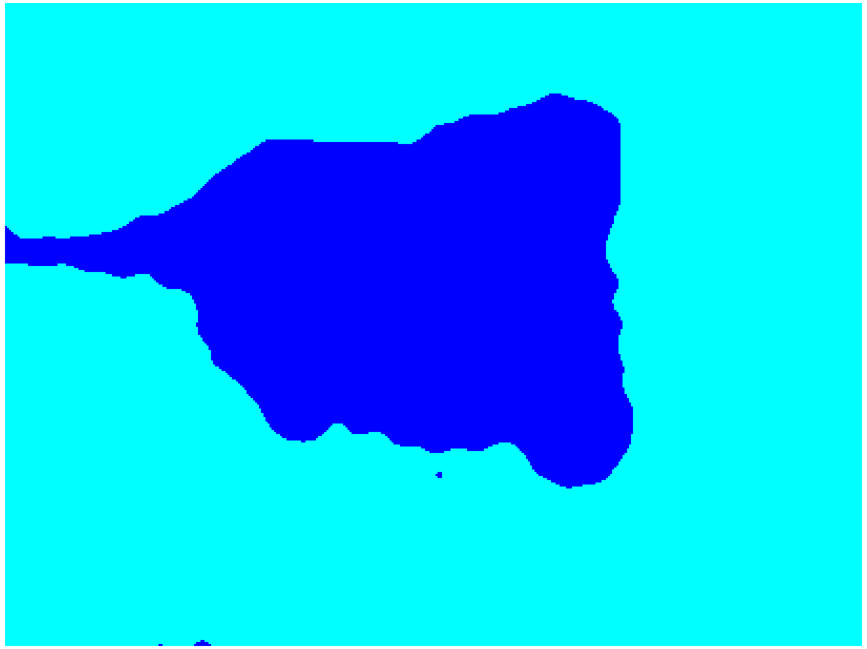
- Design an array of Gabor Filters different wavelengths and orientations
- Use orientations (ϴ) between 0 and 150^0 degrees in steps of 30 degrees
- Use wavelength in 2×sqrt(2), 4×sqrt(2), ….

Texture Segmentation using Gabor filters



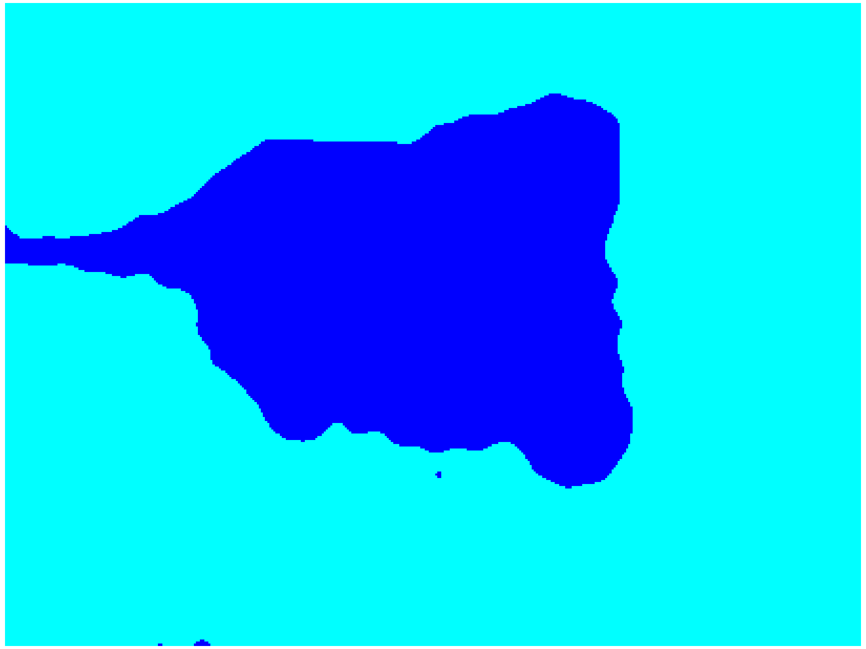
Output of Gabor filter

- Apply K-means to segment image



- Segmented region of Gabor filter outpout

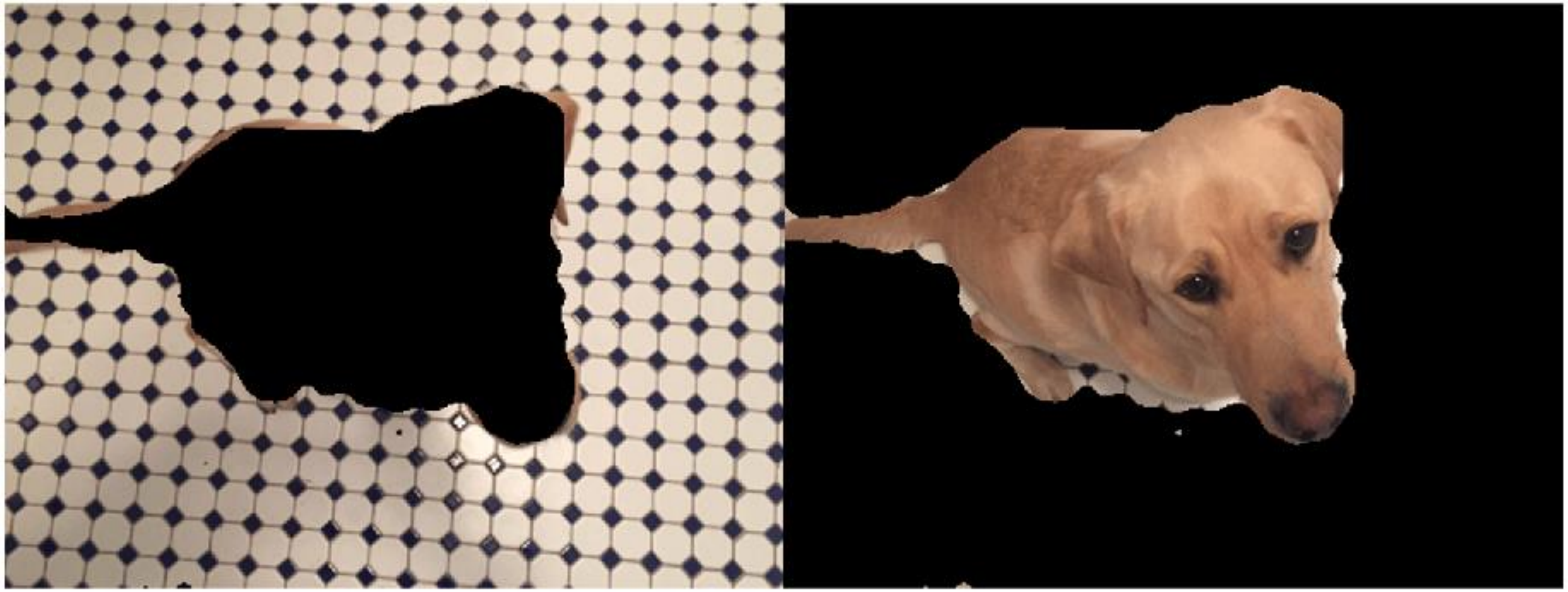
Texture Segmentation using Gabor filters



Segmented region of Gabor filter output

Corresponding segmented region of image

Texture Segmentation using Gabor filters



- Segmented region

Gray Level Co-occurrence Matrix (GLCM)

- Is a statistical method used for image processing and computer vision
- Useful for texture analysis and feature extraction
- Reveals properties about the spatial distribution of the gray levels in the texture image

Create GLCM

- Calculate how often a pixel with the intensity (gray-level) value *i* occurs with a specific spatial relation to a pixel with the value *j*
- Element of the matrix is the number of times pixels with value, i occurs with a specified relationship to a pixel with value, j in the image
- Number of gray levels in the image determines the size of GLCM

GLCM example

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

2-bit Image matrix

GLCM example

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

2-bit Image matrix

Pixel values, j →

Pixel values, i ↓	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

Initialize GLCM

- Consider (0,1) spatial relationship
- Offset is 0 rows and 1 column

GLCM example

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

2-bit Image matrix

Pixel values, j →

Pixel values, i ↓

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

Initialize GLCM

	0	1	2	3
0	2	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

Partial GLCM for (0,1) spatial relationship

- Consider (0,1) spatial relationship
- Offset is 0 rows and 1 column

GLCM example

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

2-bit Image matrix

Pixel values, j →

Pixel values, i ↓	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

Initialize GLCM

	0	1	2	3
0	2	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

Partial GLCM for (0,1) spatial relationship

- Consider (0,1) spatial relationship
- Offset is 0 rows and 1 column

	0	1	2	3
0	2	2	1	0
1	0	2	0	0
2	0	0	3	1
3	0	0	0	1

GLCM for (0,1) spatial relationship

Other relationships like (1,0), (0,-2) etc are possible

Features of GLCM

- Elements along the diagonal show number of pairs with the same intensity
- Elements which are one cell away from the diagonal represent number of pixel pairs with a difference of one grey level (0-1, 1-2, 2-3 ..)
- Elements which are two cells away from the diagonal number of pixels pairs with intensity difference of 2
- The farther away from the diagonal, the greater the difference between pixel grey levels

10	8	5			
8	10	8	5		
5	8	10	8	5	
	5	8	10	8	5
		5	8	10	8
			5	8	10

GLCM

Ex: Statistics using GLCM

j→

i

2	2	1	0
0	2	0	0
0	0	3	1
0	0	0	1

GLCM

Statistical parameters are

- Contrast
- Dissimilarity
- Energy
- Entropy
- Homogeneity
- Correlation

Normalize GLCM

i↓ \ j→	0	1	2	3
0	2	2	1	0
1	0	2	0	0
2	0	0	3	1
3	0	0	0	1

GLCM

Normalize GLCM

GLCM ($j\rightarrow$ columns, $i\downarrow$ rows)

i \ j	0	1	2	3
0	2	2	1	0
1	0	2	0	0
2	0	0	3	1
3	0	0	0	1

Normalized GLCM ($j\rightarrow$ columns, $i\downarrow$ rows)

i \ j	0	1	2	3
0	2/12	2/12	1/12	0
1	0	2/12	0	0
2	0	0	3/12	1/12
3	0	0	0	1/12

- Normalization makes GLCM scale invariant
- Elements of normalized GLCM = Probability, p(i,j)

Contrast of image derived from GLCM

$$\text{Contrast} = \sum_{i,j} |i-j|^2 p(i,j)$$

- Uses weights related to the distance from the GLCM diagonal
- Returns a intensity contrast between a pixel and its neighbor
- High value indicate large differences between neighboring pixel intensities
- Diagonal elements show 0 contrast
- Contrast is more for pixels which are away from the diagonal

Contrast of image derived from GLCM

$$\text{Contrast} = \sum_{i,j} |i - j|^2 p(i,j)$$

- If contrast = 0, image has pixels with constant intensity
- Increases exponentially as intensity difference between reference and target pixel increases
- Contrast is defined for a specific relationship used for GLCM
- For (0,1) relationship, it shows contrast in horizontal direction and with next pixel

Contrast derived from Example GLCM

j→

i↓	0	1	2	3
0	2/12	2/12	1/12	0
1	0	2/12	0	0
2	0	0	3/12	1/12
3	0	0	0	1/12

Normalized GLCM

$$\text{Contrast} = \sum_{i,j} |i-j|^2 p(i,j)$$

$= (0-1)^2\,(2/12) + (0-2)^2\,(1/12) + (2-3)^2\,(2/12)$

$= 0.5833$

Ex: GLCM feature, contrast of image

Image

GLCM (0,1 relation)

a)

Ex: GLCM feature, contrast of image

Image

GLCM (0,1 relation)

a)

Ex: GLCM feature, contrast of image

Image | GLCM (0,1 relation)

a)

b)

Ex: GLCM feature, contrast of image

Image GLCM (0,1 relation)

a)

b)

Ex: GLCM feature, contrast of image

- Relationship is (1,0)

$$\begin{bmatrix} 10 & 10 & 20 & 20 \\ 10 & 20 & 20 & 30 \\ 30 & 20 & 10 & 10 \\ 30 & 30 & 20 & 20 \end{bmatrix}$$

Image

Ex: GLCM feature, contrast of image

- Relationship is (1,0)

$$\begin{bmatrix} 10 & 10 & 20 & 20 \\ 10 & 20 & 20 & 30 \\ 30 & 20 & 10 & 10 \\ 30 & 30 & 20 & 20 \end{bmatrix}$$

Image

j →

i ↓

$$\begin{array}{cc} & \begin{matrix} 10 & 20 & 30 \end{matrix} \\ \begin{matrix} 10 \\ 20 \\ 30 \end{matrix} & \begin{bmatrix} 1 & 3 & 1 \\ 1 & 2 & 2 \\ 1 & 0 & 1 \end{bmatrix} \end{array}$$

GLCM

Ex: GLCM feature, contrast of image

- Relationship is (1,0)

$$\begin{bmatrix} 10 & 10 & 20 & 20 \\ 10 & 20 & 20 & 30 \\ 30 & 20 & 10 & 10 \\ 30 & 30 & 20 & 20 \end{bmatrix}$$

Image

j →

$$\begin{array}{c c} & \begin{array}{ccc} 10 & 20 & 30 \end{array} \\ \begin{array}{c} 10 \\ 20 \\ 30 \end{array} & \begin{bmatrix} 1 & 3 & 1 \\ 1 & 2 & 2 \\ 1 & 0 & 1 \end{bmatrix} \end{array}$$

i ↓

GLCM

$$\begin{array}{c c} & \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \end{array} & \begin{bmatrix} 1/12 & 3/12 & 1/12 \\ 1/12 & 2/12 & 2/12 \\ 1/12 & 0 & 1/12 \end{bmatrix} \end{array}$$

Normalized GLCM

$$\text{Contrast} = \sum_{i,j} |i-j|^2 p(i,j)$$

$= (0-1)^2(3/12) + (0-2)^2(1/12) + (1-0)^2(1/12) + (1-2)^2(2/12) + (2-0)^2(1/12)$

Dissimilarity using GLCM

$$dissimilarity = \sum_{i,j} |i - j| p(i,j)$$

- Measures average difference in intensity between neighboring pixels
- Dissimilarity increases linearly with the difference between pairs of pixels
- High dissimilarity values indicate greater heterogeneity in texture

Dissimilarity derived from Example GLCM

$$dissimilarity = \sum_{i,j} |i-j| p(i,j)$$

j→

i↓	0	1	2	3
0	2/12	2/12	1/12	0
1	0	2/12	0	0
2	0	0	3/12	1/12
3	0	0	0	1/12

Normalized GLCM

Dissimilarity = 0.4166

Energy derived from GLCM

$$Energy = \sum_{i,j} p(i,j)^{\,2}$$

- Is sum of squared elements in the GLCM
- Range is from 0 to 1
- Is 1 for a constant image
- Also known as uniformity or angular second moment

Energy derived from Example GLCM

$$Energy = \sum_{i,j} p(i,j)^{\ 2}$$

j→

i↓	0	1	2	3
0	2/12	2/12	1/12	0
1	0	2/12	0	0
2	0	0	3/12	1/12
3	0	0	0	1/12

Normalized GLCM

- High value indicate more uniform texture

- Energy = 0.4082

Homogeneity derived from GLCM

$$homogeneity = \sum_{i,j} \frac{p(i,j)}{1 + |(i-j)|}$$

- Is inverse of the contrast
- Measures how similar the pixel values are to their neighbors
- Measures the number of elements close to the diagonal
- High values indicate that more elements are concentrated near or along the diagonal
- Therefore high value indicates more uniform texture or less contrast
- Range is from 0 to 1

Homogeneity derived from Example GLCM

$$homogeneity = \sum_{i,j} \frac{p(i,j)}{1 + |(i-j)|}$$

i↓ \ j→	0	1	2
0	0	0.5	0.5
1	0.5	0.5	0
2	0	0	0

Probability, p(i,j)

Homogeneity = [0.5/(1+|0-1|)]+ [0.5/(1+|1-2|)]+ [0.5/(1+|1-0|)]+ [0.5/(1+|0-1|)]

Entropy derived from GLCM

$$entropy = \sum_{i,j} -p(i,j)log_2(p(i,j))$$

- Measures the randomness or complexity of the texture
- Measure of how pairs pixels with specific relationship are equally distributed
- Entropy is large for coarse texture

Entropy derived from GLCM

$$entropy = \sum_{i,j} -p(i,j)log_2(p(i,j))$$

10	10	10	10
10	10	10	10
10	10	10	10
10	10	10	10

Image

Entropy derived from GLCM

$$entropy = \sum_{i,j} -p(i,j)log_2(p(i,j))$$

10	10	10	10
10	10	10	10
10	10	10	10
10	10	10	10

Image

$$\begin{array}{cc} & 10 \\ 10 & [12] \end{array}$$

GLCM for (0,1)

$$[1]$$

Normalized GLCM (i=0 1nd j=0

entropy = p(0,0) $\log_2$p(0,0)
= -1$\log_2$(1)
= 0

GLCM mean, variance

- Is mean of number of times a pixel's occurrence with a specific relationship for current pixel, i or for j

$$\mu_i = \sum_{i,j} -ip(i,j) \qquad \mu_j = \sum_{i,j} -jp(i,j)$$

- Variance is dependent on the mean, and the dispersion around the mean

$$\sigma_i^2 = \sum_{i,j} p(i,j)(i - \mu_i\ )^2 \qquad \sigma_j^2 = \sum_{i,j} p(i,j)(j - \mu_j\ )^2$$

Mean derived from Example GLCM

$$\mu_i = \sum_{i,j} -ip(i,j)$$

$$\mu_j = \sum_{i,j} -jp(i,j)$$

μ_i =1.833

μ_j =1.917

j→

i↓	0	1	2	3
0	1/12	2/12	0	0
1	1/12	3/12	2/12	0
2	0	2/12	2/12	1/12
3	0	0	1/12	1/12

Normalized GLCM

Variance derived from Example GLCM

$$\sigma_i^2 = \sum_{i,j} p(i,j)(i - \mu_i\ )^2$$

$\sigma_i^2 = 1.878$

$$\sigma_j^2 = \sum_{i,j} p(i,j)(j - \mu_j\ )^2$$

$\sigma_j^2 = 1.298$

$$Correlation = \sum_{i,j} (i - u_i)(j - \mu_j)p(i,j)/(\sigma_i \sigma_j)$$

j→

i↓	0	1	2	3
0	1/12	2/12	0	0
1	1/12	3/12	2/12	0
2	0	2/12	2/12	1/12
3	0	0	1/12	1/12

Normalized GLCM

Mean and Variance derived from Example GLCM

μ_i =1.833

μ_j =1.917

$\sigma_i^2 = 1.878$

$\sigma_j^2 = 1.298$

i↓ \ j→	0	1	2	3
0	1/12	2/12	0	0
1	1/12	3/12	2/12	0
2	0	2/12	2/12	1/12
3	0	0	1/12	1/12

Normalized GLCM

$$Correlation = \sum_{i,j}(i - u_i)(j - \mu_j)p(i,j)/(\sigma_i \sigma_j)$$

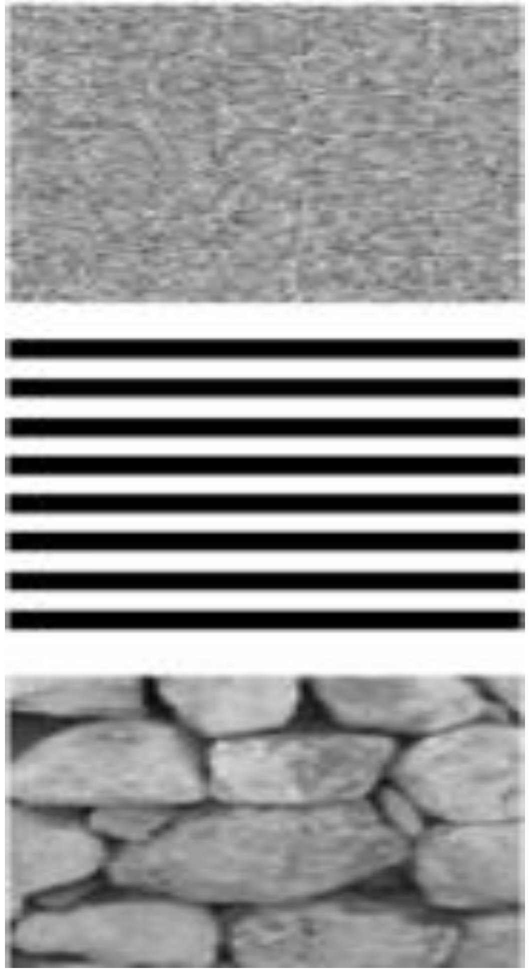
Correlation = 1.089/ 1.256 = 0.868

Correlation from GLCM

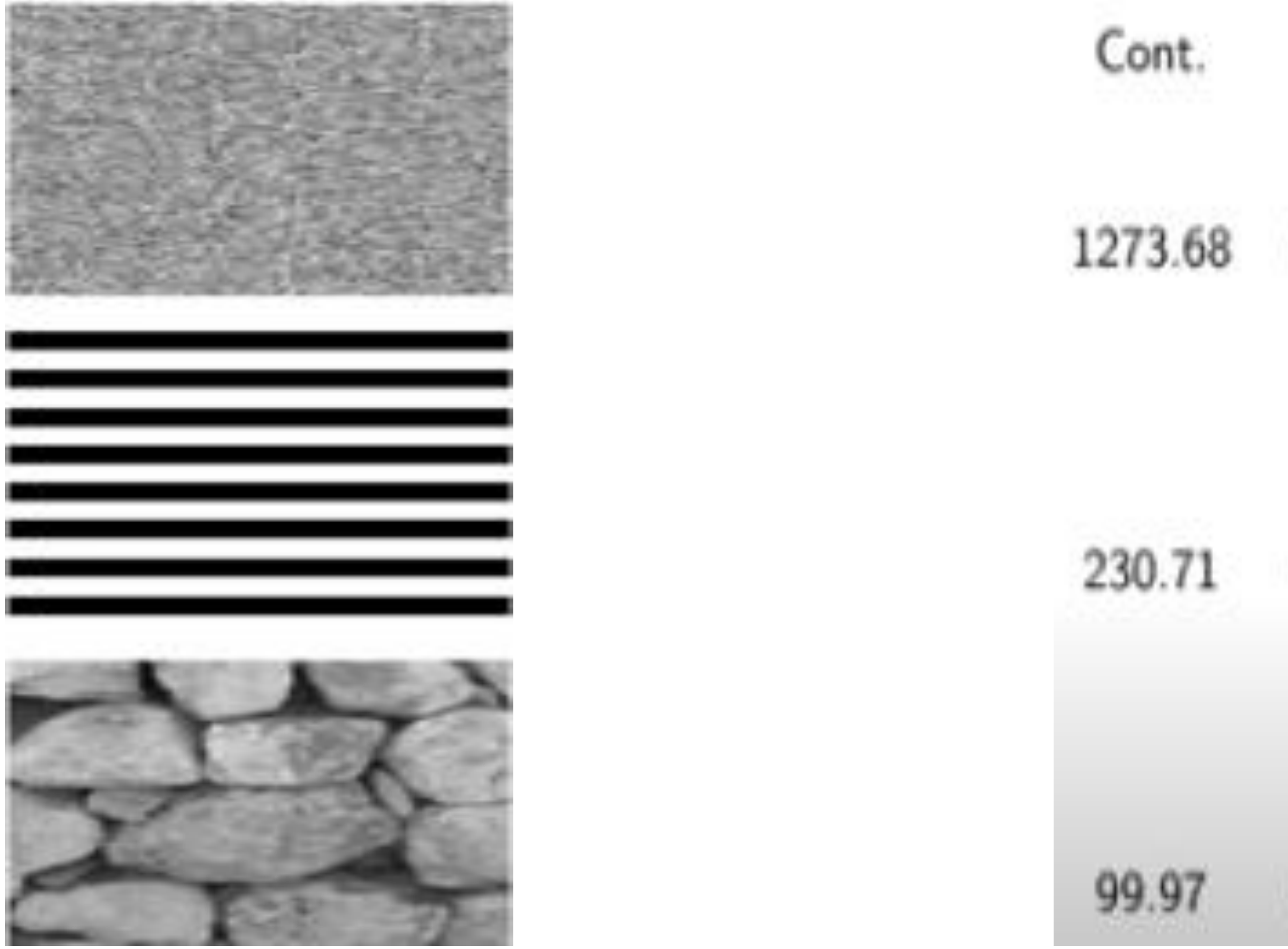
$$Correlation = \sum_{i,j} (i - u_i)(j - \mu_j)p(i,j)/(\sigma_i \sigma_j)$$

- Measure of how correlated a pixel is to its neighbor in the image
- Range is -1 to 1
- 1 or -1 for a perfectly positively or negatively correlated image.
- NaN for constant image

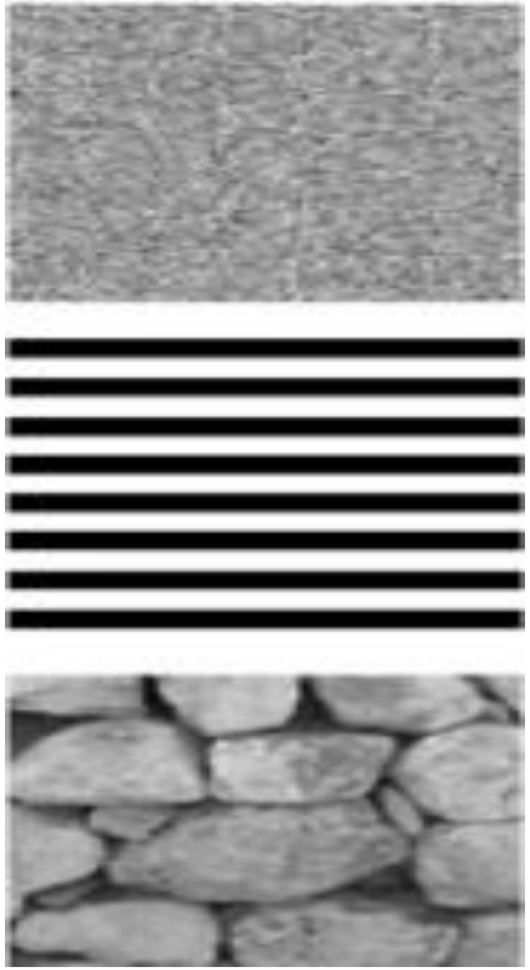
GLCM features



GLCM features

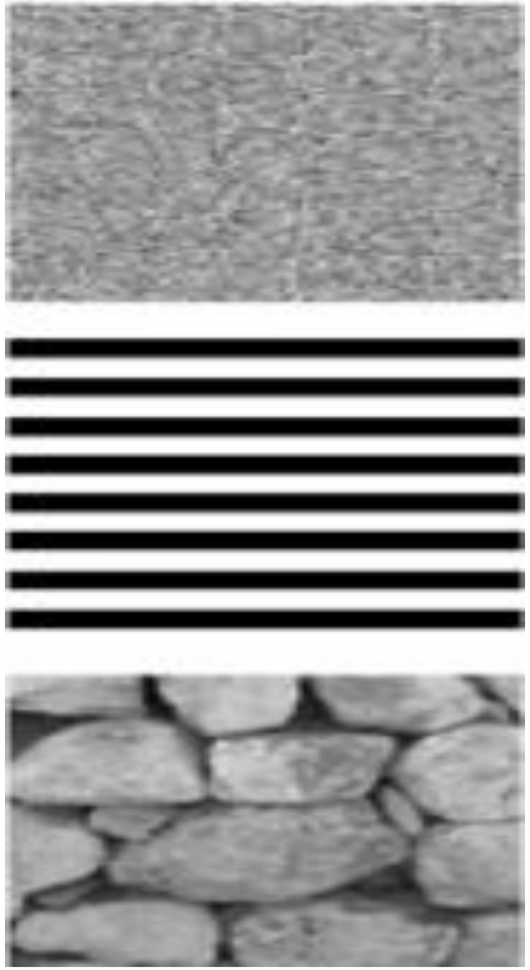


GLCM features



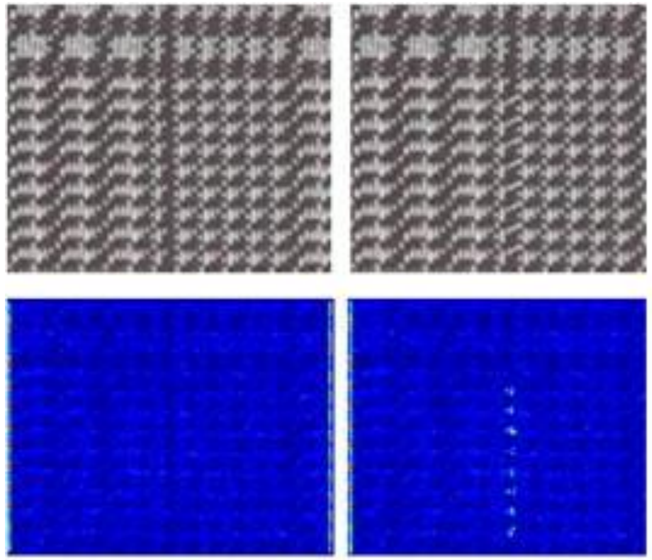
Cont.	Homog.
1273.68	0.328
230.71	0.512
99.97	0.639

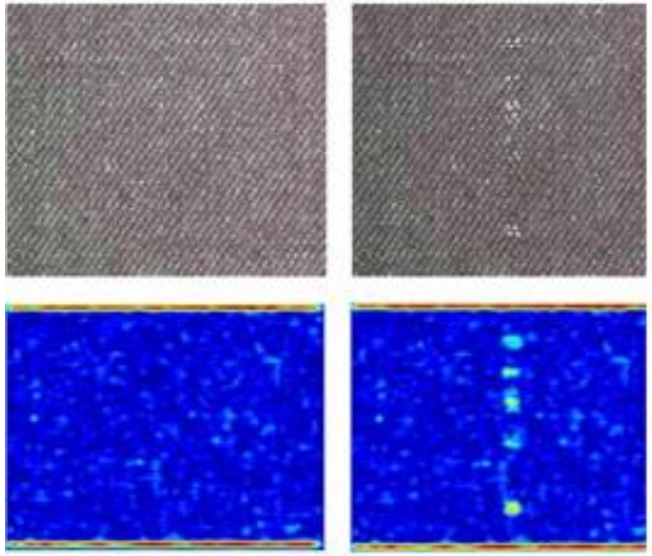
GLCM features



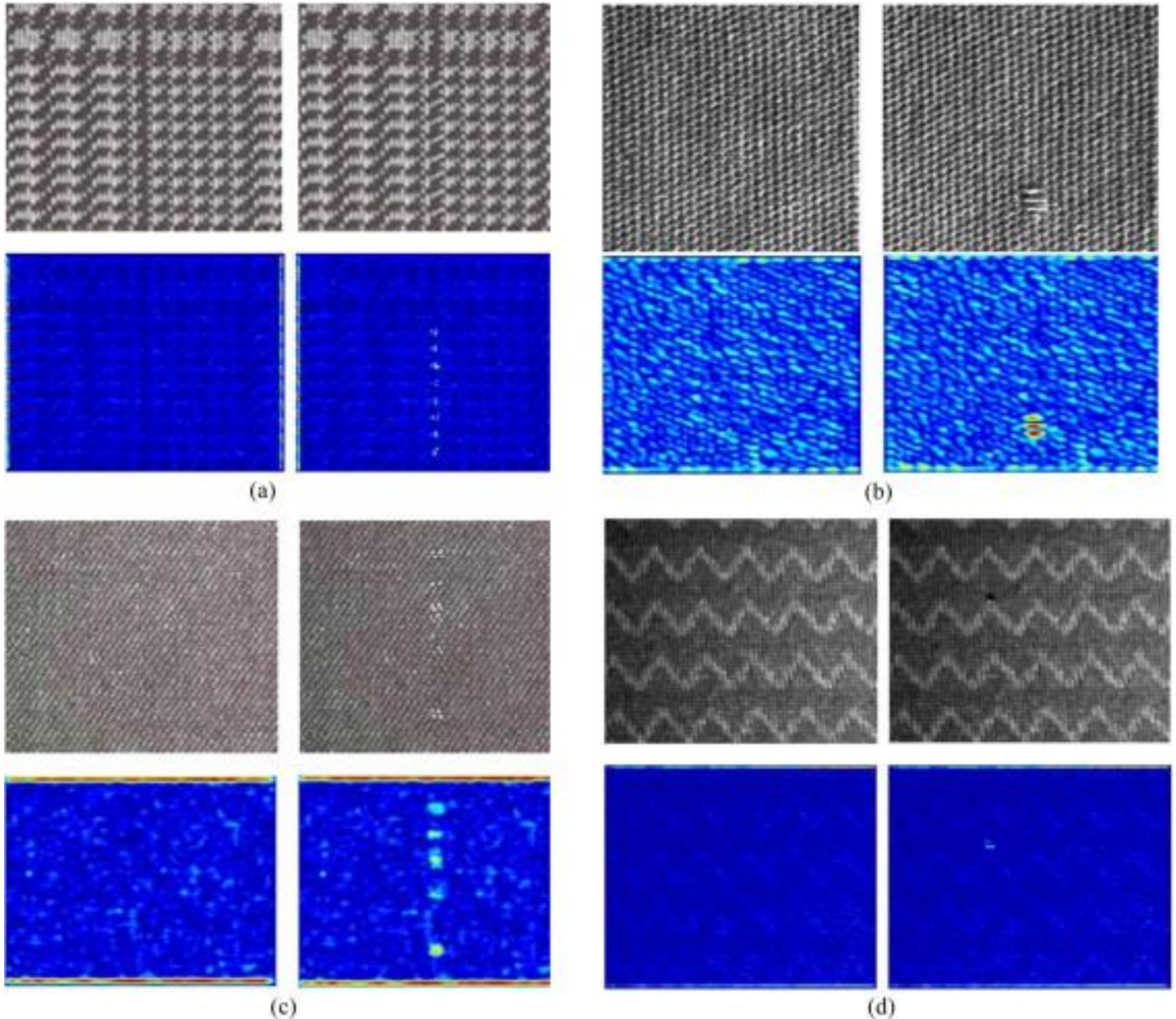
Cont.	Homog.	Entrop.
1273.68	0.328	5.741
230.71	0.512	1.320
99.97	0.639	4.323

GLCM for detection of defects in fabric


(a)


(c)

GLCM for detection of defects in fabric



1. Determine GLCM features of image without defect
2. Train machine learning model with features
3. Apply features of defective or not defective images as test samples
4. Use model to classify defective image

Texture Classification

1. Convert the images to grayscale
2. GLCM is typically computed on grayscale images
3. Normalize the image to ensure consistent gray level distribution
4. Choose the direction (0°, 45°, 90°, 135°) and distance (usually 1 pixel, but others can be used) for computing the GLCM.
5. Ex: relationship, (1,1), (2,2), etc have direction, 45°
6. For each image, compute the GLCM for each direction and distance
7. Extract Features like image uniformity, Contrast, correlation, homogeneity entropy
6. Combine the extracted features from the GLCM into a feature vector for each image
7. Feature vector represents the texture characteristics of the image

Image Classification

5. Divide the dataset into training and testing sets
6. Apply a classification algorithm such as Support Vector Machine (SVM), Random Forest, k-Nearest Neighbors (k-NN), or Neural Networks to classify the images based on their GLCM features
7. Train the classifier using the training set
8. Test the classifier on the testing set and evaluate its performance using metrics such as accuracy, precision, recall, and F1 score

Application of texture image

- **Image Classification:** Generate GLCM for a small area of the image and determine texture features. Use SVM, Random forest etc to classify images
- **Texture Classification:** Differentiate between different types of textures in an image
- **Image Segmentation:** Identify regions of interest based on texture features
- **Pattern Recognition:** Enhance the performance of pattern recognition algorithms by including texture information.

References

- https://www.oreilly.com/library/view/programming-computer-vision/9781449341916/ch02.html
- https://www.baeldung.com/cs/image-processing-feature-descriptors
- https://sbme-tutorials.github.io/2018/cv/notes/9_week9.html
- https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/
- https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f
- https://medium.com/@dnemutlu/hog-feature-descriptor-263313c3b40d
- https://learnopencv.com/histogram-of-oriented-gradients/
- https://debuggercafe.com/image-recognition-using-histogram-of-oriented-gradients-hog-descriptor/
- https://www.google.com/amp/s/iq.opengenus.org/object-detection-with-histogram-of-oriented-gradients-hog/amp/
- https://www.google.com/url?sa=t&source=web&rct=j&url=https://m.youtube.com/watch%3Fv%3D5nZGnYPyKLU&ved=2ahUKEwjn2ISl1sz-AhUERmwGHe-aB3EQo7QBegQIDBAF&usg=AOvVaw3j2q__19MNeHimMyT1lewO

References

- https://nptel.ac.in/courses/108103174
- https://www.google.com/url?sa=t&source=web&rct=j&url=https://m.youtube.com/watch%3Fv%3Dm-_lRRScetk&ved=2ahUKEwjn2ISl1sz-AhUERmwGHe-aB3EQo7QBegQIAxAF&usg=AOvVaw1Hfzvqvgo3grrqA8jyZiY9
- https://sbme-tutorials.github.io/2018/cv/notes/6_week6.html
- https://medium.com/data-breach/introduction-to-harris-corner-detector-32a88850b3f6
- https://www.baeldung.com/cs/harris-corner-detection
- https://www.codingninjas.com/codestudio/library/harris-corner-detection
- https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.cs.umd.edu/class/fall2019/cmsc426-0201/files/12_HarrisCornerDetection.pdf&ved=2ahUKEwj_q4fY5br-AhWu-jgGHeTBCJAQFnoECD8QAQ&usg=AOvVaw0WjY5eRFeu-vCUFu-g6o90
- https://fiveko.com/feature-points-using-harris-corner-detector/

References

- https://www.google.com/url?sa=t&source=web&rct=j&url=https://m.youtube.com/watch%3Fv%3DPtc4dEnPwt8&ved=2ahUKEwjF6lnyw8X-AhWvTWwGHRR9CWcQo7QBegQIAxAF&usg=AOvVaw1NG86Bu1KDU3Dc0H1EuVNs
- https://towardsdatascience.com/sift-scale-invariant-feature-transform-c7233dc60f37
- http://www.scholarpedia.org/article/Scale_Invariant_Feature_Transform
- https://www.google.com/amp/s/www.geeksforgeeks.org/sift-interest-point-detector-using-python-opencv/amp/
- https://www.codingninjas.com/codestudio/library/scale-invariant-feature-transform-sift
- https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/
- https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40
- https://nptel.ac.in/courses/108103174

References

- https://levelup.gitconnected.com/the-integral-image-4df3df5dce35
- https://in.mathworks.com/help/images/integral-image.html
- https://traffic.nayan.co/blog/AI/Integral-Image/
- https://www.researchgate.net/figure/9x9-15x15-box-filter-Filters-Dyy-left-and-Dxy-right-for-two-successive-scale-levels_fig3_330349878
- https://www.google.com/url?sa=t&source=web&rct=j&url=https://m.youtube.com/watch%3Fv%3DaEclsx_aT6U&ved=2ahUKEwjU0dSK7ML-AhXf-zgGHYNiAiMQwqsBegQIUxAE&usg=AOvVaw19Y_Ibe14zebgND0Yoiu6g
- https://medium.com/data-breach/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e
- https://en.m.wikipedia.org/wiki/Speeded_up_robust_features#:~:text=The%20SURF%20algorithm%20is%20based,local%20neighborhood%20description%2C%20and%20matching.