Project 3.
Texture Classification

S M Al Mahi

Oklahoma State University

March 16, 2018

Project Objective

Objectives

1. Classification of 59 given Texture Brodatz datset
2. Implementation of Laplacian Pyramid for Texture Classification
3. Implementation of Gabor Filter Bank for Texture Classification
4. PCC or Percentage of Correctly classified texture

Laplacian Pyramid Texture Classification

Choose the number of scales in the Laplacian pyramid and the smoothing filter used prior to each down-sampling

↓

For each of the given texture images, we develop a M-scale Laplacian pyramid and construct a feature vector by computing certain statistics (e.g., variance) for each scale in the Laplacian pyramid

↓

For all texture images, we normalize their feature vectors by scaling each coefficient by the maximum and minimum values in that channel across all textures. This will construct a texture library with all normalized feature vectors.

↓

For an unknown texture image (or a small patch), we compute and normalize the corresponding feature vector and find the best match in the library.

Gabor Filter Bank Texture Classification

Construct a *M*-scale and *N*-orientation Gabor filter bank (totally *M*x*N* channels)

↓

For each of the given texture images, we convolve *M*x*N* filters with the image and construct a feature vector by computing certain statistics, (e.g., variance) of the filtering output for each channel.

↓

For all texture images, we normalize their feature vectors by scaling each coefficient by the maximum and minimum values in that channel across all textures. This will construct a texture library with all normalized feature vectors.

↓

For an unknown texture image (or a small patch), we compute and normalize the corresponding feature vector and find the best match in the library.

Issues With Texture Classification

- **Scales:** Mean, variance, skewness, Kurtiosis
- **Size and Orientation:** Kernel window size, orientation and standard deviation

Laplacian Basic Implementation 67.53%

Texture Num	PCC
D01	78
D02	86
D03	47
D04	33
D05	81
D06	75
D07	76
D08	66
D09	86
D10	66
D11	43
D12	79
D13	75
D14	83
D15	78
D16	83
D17	92
D18	28
D19	71
D20	84
D21	100
D22	63
D23	64
D24	79
D25	41
D26	79
D27	67
D28	38
D29	51
D30	74
D31	53
D32	100
D33	93
D34	63
D35	73
D36	73
D37	56
D38	89
D39	49
D40	51
D41	45
D42	52
D43	75
D44	99
D45	69
D46	65
D47	89
D48	85
D49	97
D50	48
D51	32
D52	59
D53	92
D54	43
D55	65
D56	39
D57	50
D58	47
D59	67
AVG PCC	67.53

Basic Laplacian Pyramid Building Pitfalls

- Kernels can be smoothed in all layers using same parameters
- Distance metric for all the layers are same
- All the pixels are convoluted similar way if same variance and window size is used

Practical Observation in Laplacian Pyramid

- Laplacian Pyramid is Built by Smoothing and Down Sampling
- It does not make sense to use same smoothing parameter for a image which half in size than previous layer
- Specifically, smallest layer convoluted with large kernel with small σ can make it look like uniform distribution
- Pixels nearby boundary are affected by boundary in convolution
- Distance Metric in all layer are not the same

Laplacian Pyramid Best AVG PCC 76.98%

Texture Num	PCC
D01	77
D02	88
D03	64
D04	42
D05	98
D06	86
D07	80
D08	72
D09	87
D10	73
D11	76
D12	83
D13	65
D14	75
D15	82
D16	86
D17	95
D18	36
D19	91
D20	95
D21	100
D22	62
D23	77
D24	59
D25	72
D26	95
D27	84
D28	89
D29	51
D30	89
D31	82
D32	43
D33	83
D34	98
D35	80
D36	83
D37	81
D38	93
D39	55
D40	60
D41	45
D42	71
D43	93
D44	100
D45	100
D46	94
D47	88
D48	88
D49	97
D50	80
D51	35
D52	67
D53	91
D54	52
D55	94
D56	48
D57	43
D58	82
D59	87
AVG PCC	76.98

Laplacian Pyramid Insights: Good Case PCC 100%

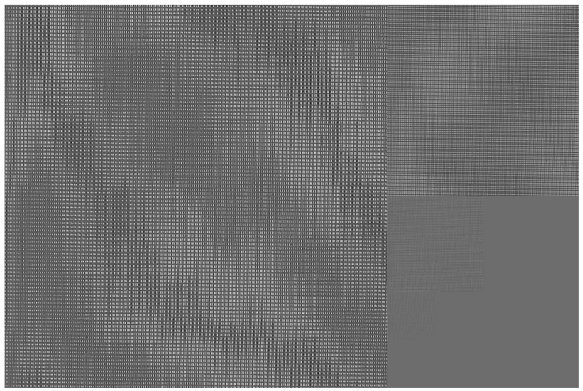

Figure: Laplacian Pyramid Library D21

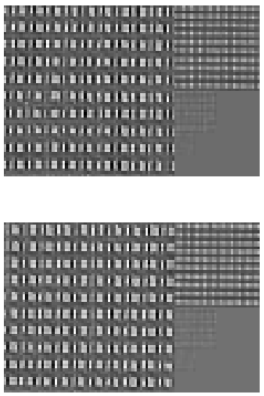

Figure: Laplacian Pyramid Blocks 56 and 100th blocks

Laplacian Pyramid Insights: Bad Case PCC 28%

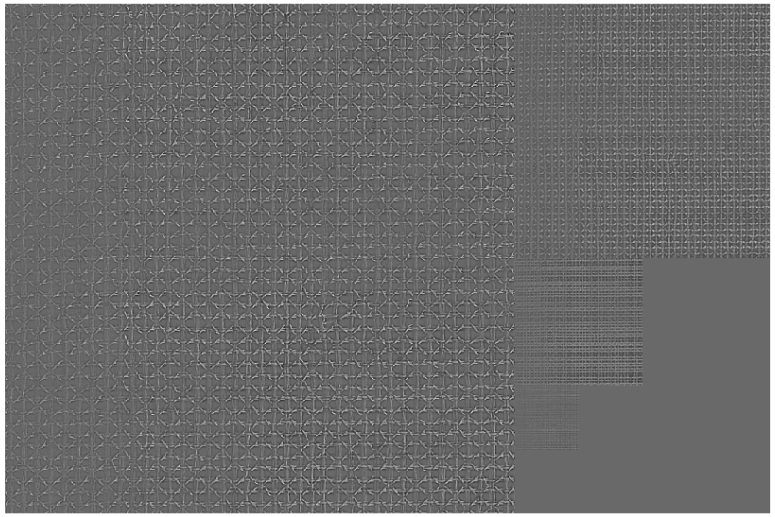

Figure: Laplacian Pyramid Library D51

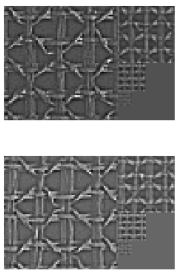

Figure: Laplacian Pyramid Blocks 45 and 100th blocks

Laplacian Pyramid Optimization

Table: Optimum Kernel Variance Parameter

Window Size	Variance
8	.883
16	.865
32	.854
64	.8525
80	.825
160	.848
320	.849
640	.847

Laplacian Pyramid Optimization

- Mahalonobis distance gave best PCC
- Euclidean Distance give 71.3% PCC which was outperformed by Mahalonobis
- Variance and Kurtiosis used combinedly for 640 and 320 sized kernel
- For 160 and smaller kernel only variance was used

Gabor Filter Texture Classification

- 4 Scale and 6 orientation; 24 channels have been used for Gabor Filter Bank
- Varian, Kurtiosis and Skewness combinedly were used to build feature vector
- Cosine Distance Produced best PCC of 87.58%
- Mahalonobis Euclidean Distance Produced PCC 85.12%

Log-Gabor Pyramid AVG PCC 87.58%

TextureNum	PCC
D01	76
D02	96
D03	100
D04	49
D05	99
D06	98
D07	100
D08	64
D09	95
D10	97
D11	94
D12	96
D13	95
D14	98
D15	88
D16	100
D17	100
D18	77
D19	98
D20	100
D21	100
D22	64
D23	94
D24	97
D25	53
D26	99
D27	77
D28	94
D29	83
D30	67
D31	69
D32	97
D33	69
D34	100
D35	90
D36	70
D37	90
D38	100
D39	92
D40	99
D41	92
D42	100
D43	100
D44	100
D45	99
D46	100
D47	91
D48	100
D49	100
D50	89
D51	78
D52	83
D53	93
D54	49
D55	100
D56	90
D57	10
D58	83
D59	86
AVG PCC	87.58

Log Gabor Kernels and Spread

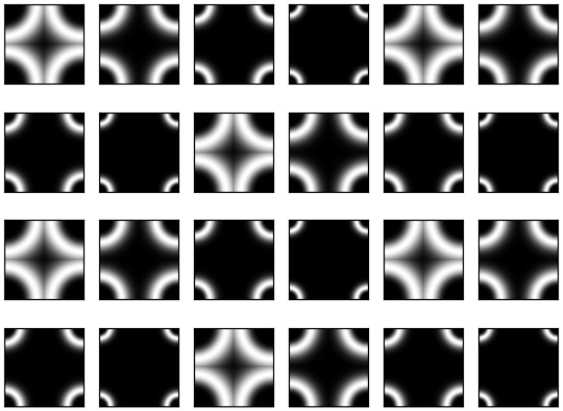

Figure: Log Gabor kernel. Gabor kernel has a different signature which was shown in the class but we are using log gabor here

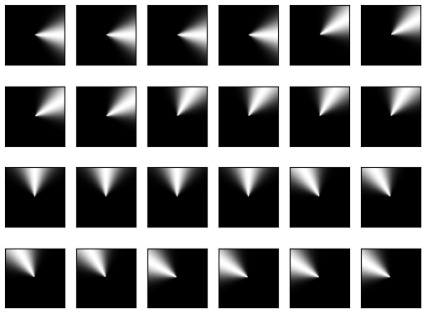

Figure: Spread orientation

Conclusion

- **Thank You**
- **Questions ?**