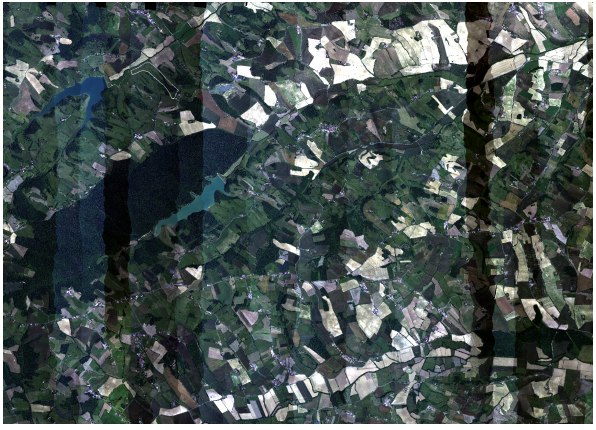
>>> Classification of Hyperspectral Images
>>> GRSS Summer School

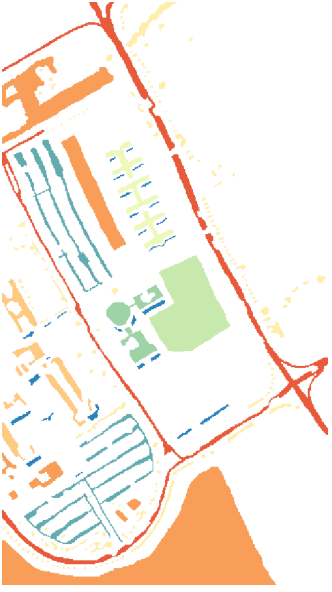
Name: Mathieu Fauvel (UMR Dynafor)
Date: *[2017-04-26 Wed]–[2017-04-27 Thu]*

>>> Classification of hyperspectral image is a challenging problem

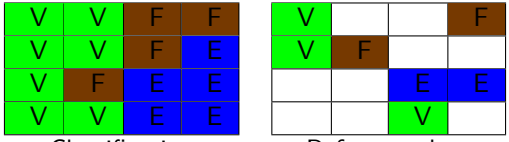
- Remember *Introduction*: High dimensional data
  - High number of features
  - Large volume of pixels
  - Low number of labelled pixels
- Practical issues:
  - Intra-class spectral variability
  - Non-linear pre-processing (atmospheric/geometric corrections)
  - Noise in the data



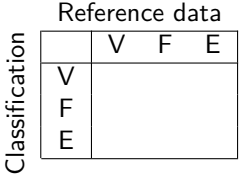
- ⋆ Reference data ?
  - ⋆ Supervised
  - ⋆ Unsupervised
  - ⋆ Semi-supervised
- ⋆ $p(\mathbf{x}, y) \sim ?$:
  - ⋆ Parametric models
  - ⋆ Non parameterics models
- ⋆ Number of classes ?
  - ⋆ Multiclass
  - ⋆ One-class / detection
  - ⋆ Unknown



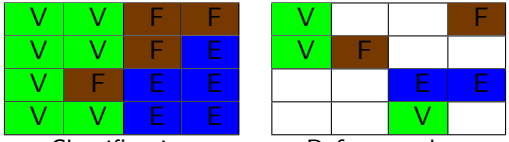
Confusion Matrix: Compute the agreement between the classifier and the reference data.



Construction :



Confusion Matrix: Compute the agreement between the classifier and the reference data.

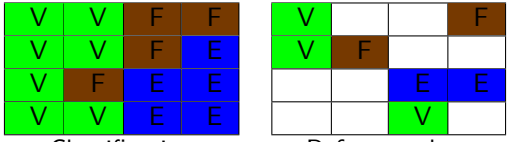


Construction :

Reference data

Classification	V	F	E
V	1		
F			
E			

Confusion Matrix: Compute the agreement between the classifier and the reference data.

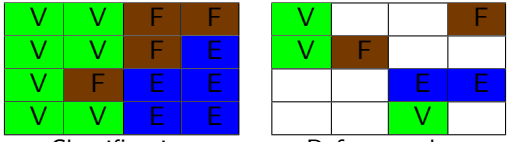


Construction :

Reference data

Classification	V	F	E
V	1		
F		1	
E			

Confusion Matrix: Compute the agreement between the classifier and the reference data.

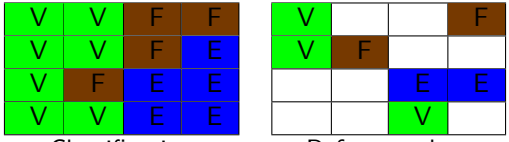


Construction :

Reference data

Classification	V	F	E
V	2		
F		1	
E			

Confusion Matrix: Compute the agreement between the classifier and the reference data.

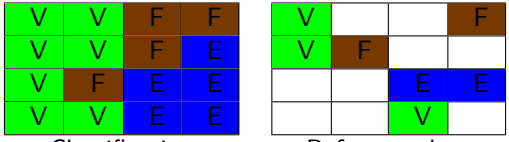


Construction :

Reference data

Classification	V	F	E
V	2	1	
F		1	
E			

Confusion Matrix: Compute the agreement between the classifier and the reference data.

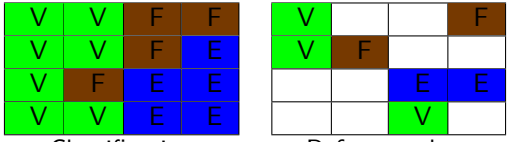


Construction :

Reference data

Classification	V	F	E
V	2	1	
F		1	
E			1

Confusion Matrix: Compute the agreement between the classifier and the reference data.

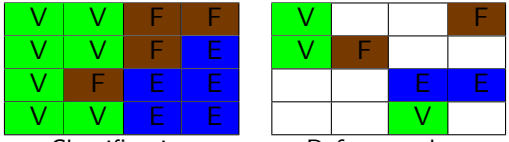


Construction :

Reference data

Classification	V	F	E
V	2	1	
F		1	
E			2

Confusion Matrix: Compute the agreement between the classifier and the reference data.

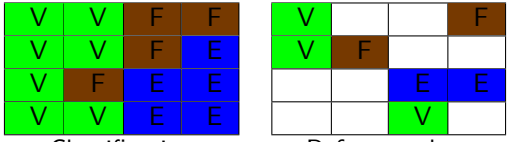


Construction :

Reference data

Classification	V	F	E
V	2	1	
F		1	
E	1		2

Confusion Matrix: Compute the agreement between the classifier and the reference data.



Construction :

Reference data

Classification	V	F	E
V	2	1	0
F	0	1	0
E	1	0	2

Classification \ Reference data	1	...	C	
1	n_{11}	...	n_{1C}	p_1^{ppv}
⋮	⋮	⋱		⋮
C	n_{C1}		n_{CC}	p_C^{ppv}
	p_1^{tpr}	...	p_C^{tpr}	

- ⋆ Overall accuracy: $\frac{\sum_{i=1}^{C} n_{ii}}{n}$
- ⋆ True positive rate (tpr): Percentage of pixels of the reference data belonging to class i that are correctly classified to class i.

$$p_i^{tpr} = \frac{n_{ii}}{\sum_{j=1}^{C} n_{ji}}$$

- ⋆ Positive predicted value (ppv): Percentage of pixels classified to class i that belong actually to class i in the reference data.

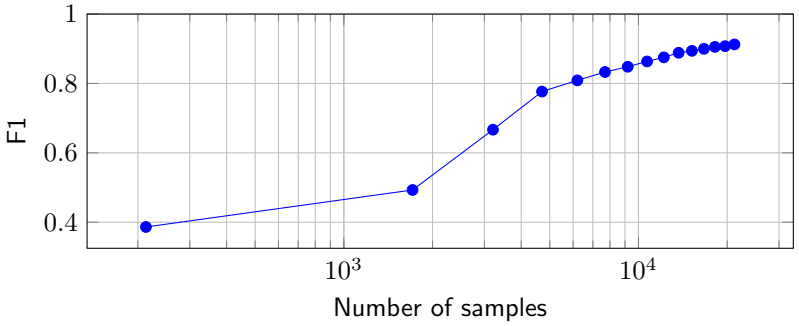
$$p_i^{ppv} = \frac{n_{ii}}{\sum_{j=1}^{C} n_{ij}}$$

- ⋆ F1 score of class i:

$$F1_i = 2\frac{1}{1/p_i^{tpr} + 1/p_i^{ppv}}$$

>>> Experimental set-up

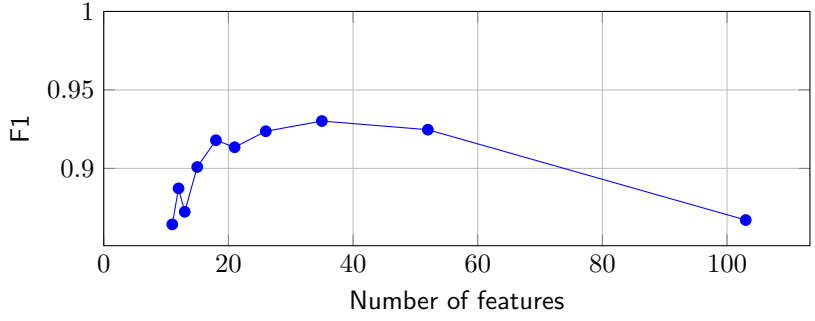
```
import scipy as sp
import rasterTools as rt
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.metrics import f1_score

# Load data set
X,y=rt.get_samples_from_roi('../Data/university.tif','../Data/university_gt.tif')

# Split the data
X_, X_test, y_, y_test = train_test_split(X, y, train_size=0.50,random_state=0,stratify=y)

# Try differents size of the training set
SPLIT = sp.linspace(0.01,0.99,15)
F1,NS = sp.zeros_like(SPLIT),sp.zeros_like(SPLIT)
for i,split in enumerate(SPLIT):
    # Split the data
    X_train, _, y_train, _ = train_test_split(X_, y_, train_size=split,random_state=0,stratify=y_)
    # Learn the classifier
    clf = QuadraticDiscriminantAnalysis()
    clf.fit(X_train,y_train)
    # Predict the classes
    yp = clf.predict(X_test)
    #Compute the F1
    F1[i],NS[i] = f1_score(y_test,yp,average='weighted'),y_train.size
```



- ⋆ Dimension of the data: $d = 103$
- ⋆ Number of parameters to estimate: 49139

>>> Experimental set-up

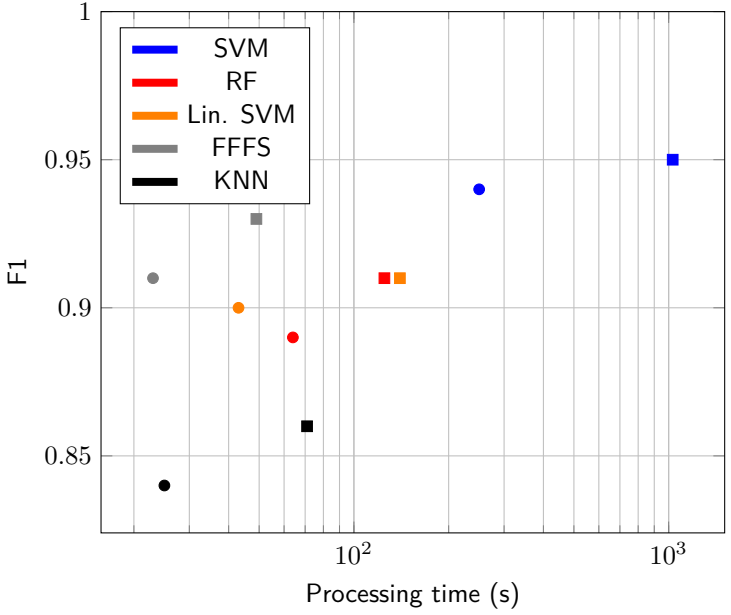
```
import scipy as sp
import rasterTools as rt
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.metrics import f1_score

# Load data set
X,y=rt.get_samples_from_roi('../Data/university.tif','../Data/university_gt.tif')

# Split the data
X_, X_test, y_train, y_test = train_test_split(X, y, train_size=0.25,random_state=0,stratify=y)

# Try differents size of the training set
SKIP = sorted(range(1,11),reverse=True)
FS,NF = sp.zeros_like(SKIP,dtype='float'),sp.zeros_like(SKIP)
for i,skip in enumerate(SKIP):
    # Skip some variables
    X_train =X_[:,::skip]
    # Learn the classifier
    clf = QuadraticDiscriminantAnalysis()
    clf.fit(X_train,y_train)
    # Predict the classes
    yp = clf.predict(X_test[:,::skip])
    #Compute the FS
    FS[i], NF[i] = f1_score(y_test,yp,average='weighted'), X_train.shape[1]
```

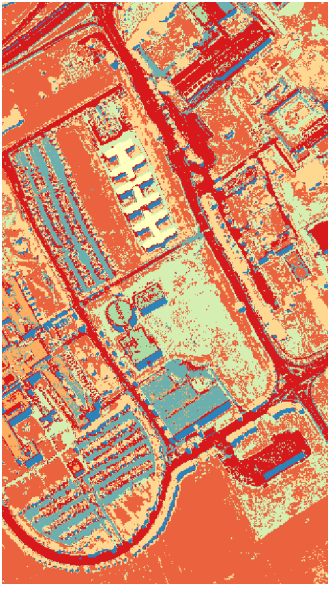

1
0.95
0.9
F1
0
20
40
60
80
100
Number of features

>>> Experimental set-up

```
if __name__ == '__main__':
    # Load data set
    X,y=rt.get_samples_from_roi('../Data/university.tif','../Data/university_gt.tif')
    sc = StandardScaler()
    X = sc.fit_transform(X)

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.1,random_state=0,stratify=y)

    # Parameters
    param_grid_svm = dict(gamma=2.0**sp.arange(-4,4), C=10.0**sp.arange(0,3)) # SVM
    param_grid_linear_svm = dict(C=10.0**sp.arange(-2,3)) # LinearSVM
    param_grid_rf = dict(n_estimators=sp.arange(10,150,10)) # RF
    param_grid_fffs = dict(maxvar=20,threshold=0.001) # FFFS
    param_grid_knn = dict(n_neighbors = sp.arange(1,50,5))
    F1,CT=[],[]

    # Start the classification: SVM
    ts=time.time()
    F1.append(compute_SVM(X_train,y_train,X_test,y_test,param_grid_svm))
    CT.append(time.time()-ts)

    # Start the classification: RF
    ts=time.time()
    F1.append(compute_RF(X_train,y_train,X_test,y_test,param_grid_rf))
    CT.append(time.time()-ts)
```

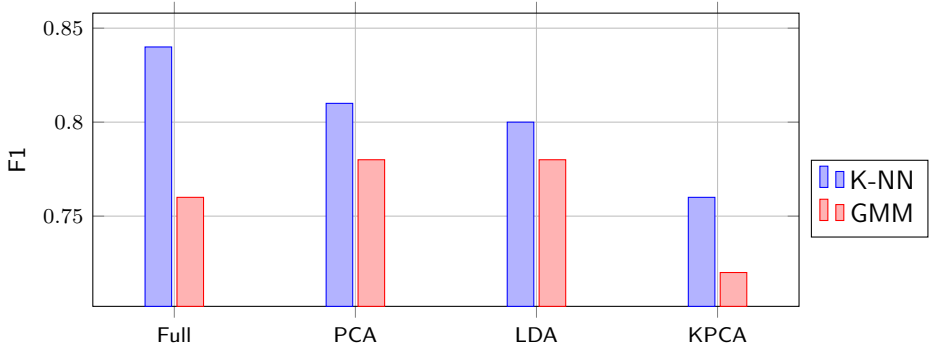
Class	•	■
1	663	1326
2	1865	3730
3	210	420
4	306	613
5	134	269
6	503	1006
7	133	266
8	368	736
9	95	189





>>> PCA, LDA and KPCA

```
DATA = ['../Data/university.tif','../Data/pca_university.tif','../Data/lda_university.tif',
        '../Data/kpca_university.tif']
GT = '../Data/university_gt.tif'

F1_knn,F1_gmm = [],[]
for data in DATA:
    print data
    # Load data set
    X,y=rt.get_samples_from_roi(data,GT)
    sc = StandardScaler()
    X = sc.fit_transform(X)
```



Spatial inter-pixel dependency

- Spatial feature extraction
  - Texture
  - Mathematical morphology
  - Convolution
- Image Segmentation
  - kmeans
  - MeanShift
- Markov Random Field

Spatial inter-pixel dependency

- Spatial feature extraction
  - Texture
  - Mathematical morphology
  - Convolution
- Image Segmentation
  - kmeans
  - MeanShift
- Markov Random Field

Joint use of spectral and spatial information

- Data fusion
  - Input: Feature stacking,
  - Output: fusion of classifier outputs.
- Post classification regularization
  - Majority vote,
  - Region growing from markers.
- Spatial-spectral classifiers:
  - Composite kernel
  - MRF

1. What kind of information is needed ?
2. How to extract it from the data ?
3. How to combine it with the spectral information ?

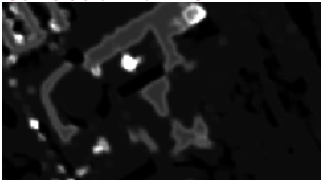
- Template filters
  - Mean, variance, median, entropy, range, ...
- Gabor features
- Wavelet features [ML02]
- Co-occurrence [HSD73]

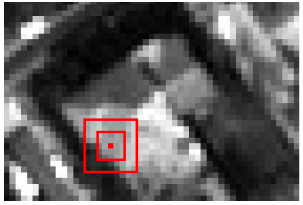
```
otbcli_HaralickTextureExtraction -in ../Data/pca_university.tif -channel 1 \
 -out ../Data/university_haralick.tif -parameters.min 789 parameters.max 64897
```

Color Image

Energy for PC1

Correlation for PC1

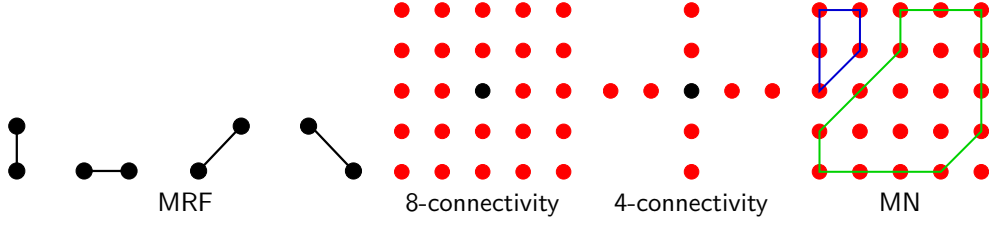




Definition (Morphological neighborhood)

The Morphological Neighborhood of a pixel $\mathbf{x}$ is the set of pixels that belongs to the same spatial structure than $\mathbf{x}$.

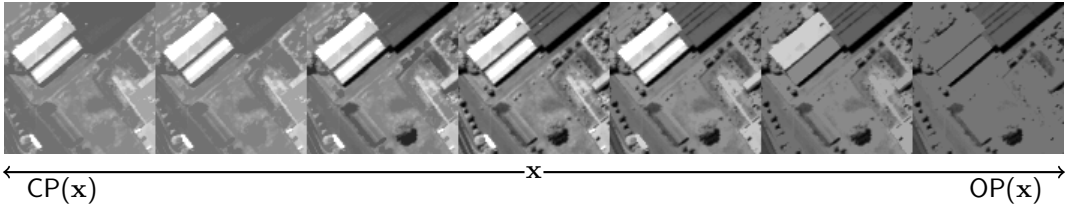
Example (Comparison with some neighborhood systems)



Definition (Morphological profile)

The Morphological Profile of size n is a $(2n+1)$-dimensional vector such as:

$$\mathrm{MP}(\mathbf{x}) = \Big[\mathrm{CP}_n(\mathbf{x}), f(\mathbf{x}), \mathrm{OP}_n(\mathbf{x})\Big].$$



For a given pixel $\mathbf{x}$, information include in the $\mathrm{MP}(\mathbf{x})$ are:

Definition (Morphological profile)

The Morphological Profile of size n is a $(2n+1)$-dimensional vector such as:

$$\mathrm{MP}(\mathbf{x}) = \Big[\mathrm{CP}_n(\mathbf{x}), f(\mathbf{x}), \mathrm{OP}_n(\mathbf{x})\Big].$$

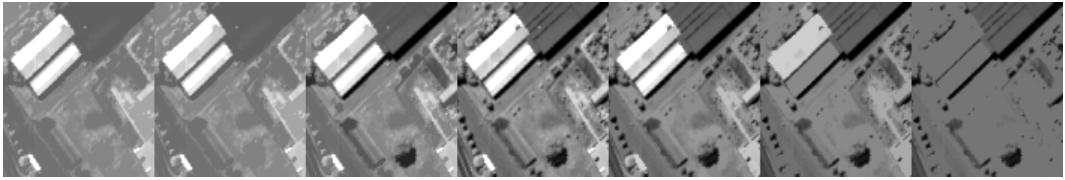
For a given pixel $\mathbf{x}$, information include in the $\mathrm{MP}(\mathbf{x})$ are:

- ⋆ Contrast: Is the structure to which the pixel belongs to darker or lighter than his surrounding neighbors?
- ⋆ Size: Is the structure to which the pixel belongs to small or big compared to G?

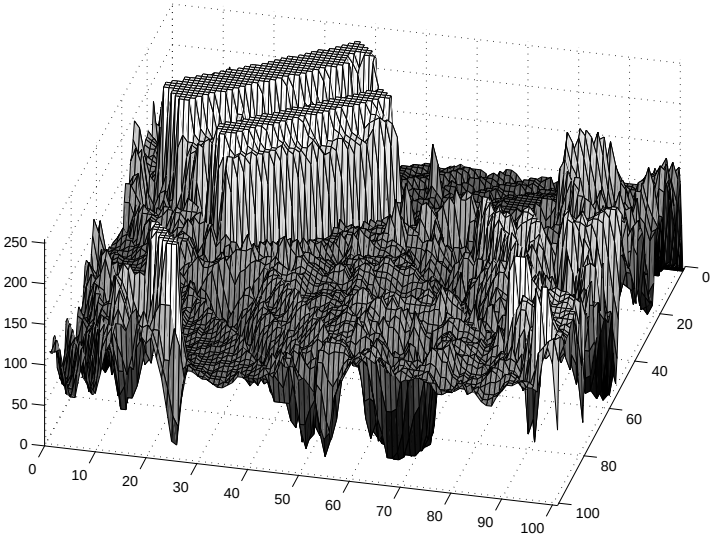
Definition (Derivative of the morphological profile)

The Derivative of the Morphological Profile of size n is a $(2n)$-dimensional vector such as:

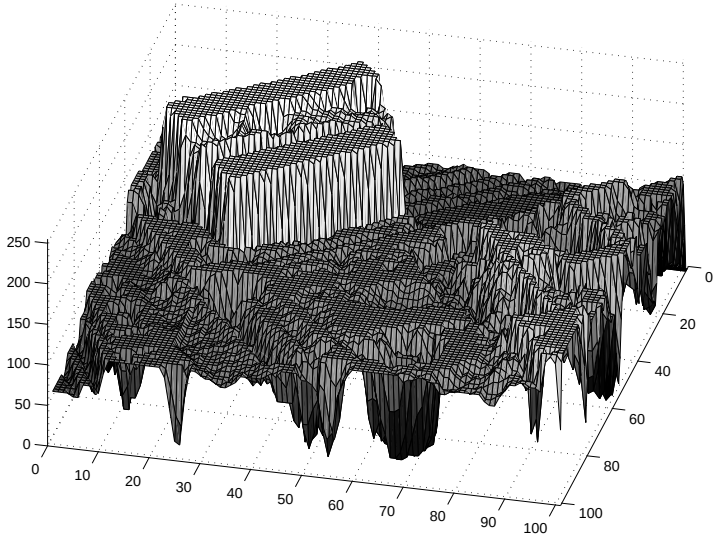
$$\mathrm{DMP}(\mathbf{x}) = \Big[|\phi_n(\mathbf{x}) - \phi_{n-1}(\mathbf{x})|, \ldots, |\gamma_{n-1}(\mathbf{x}) - \gamma_n(\mathbf{x})|\Big].$$



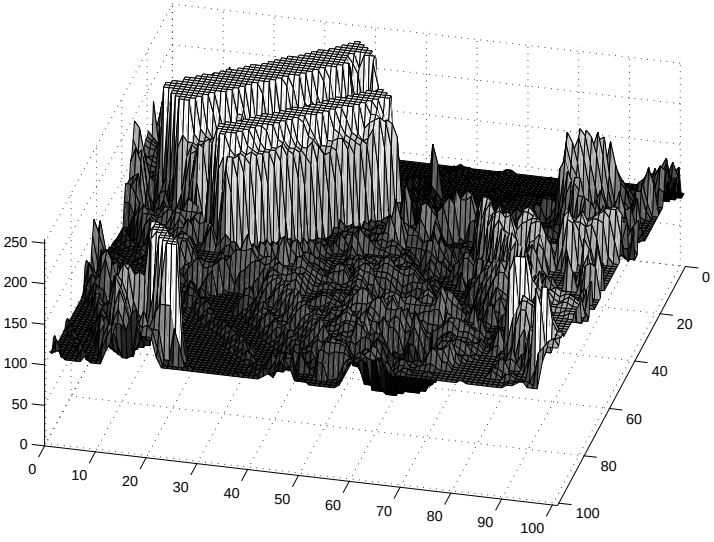
- Geodesics filters only act on extrema structures



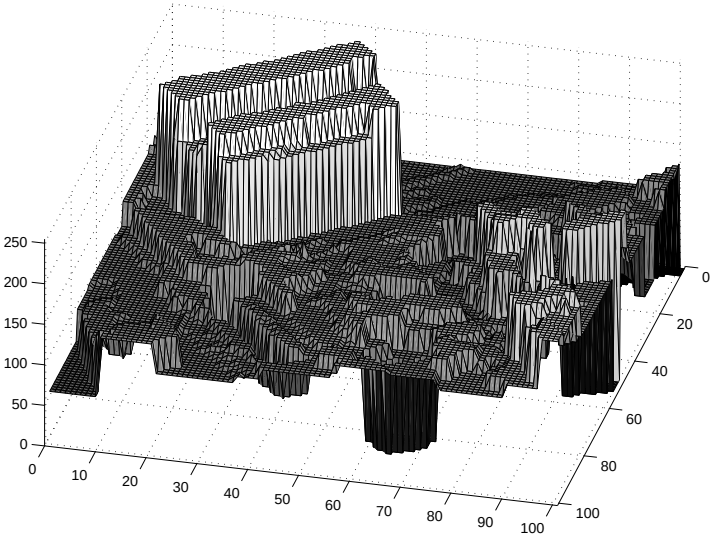
- ⋆ Geodesics filters only act on extrema structures



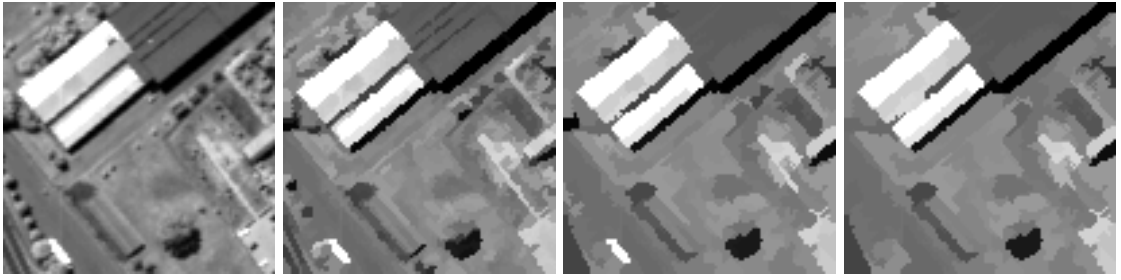
- ⋆ Geodesics filters only act on extrema structures



- ⋆ Geodesics filters only act on extrema structures
- ⋆ Self-complementary area filter

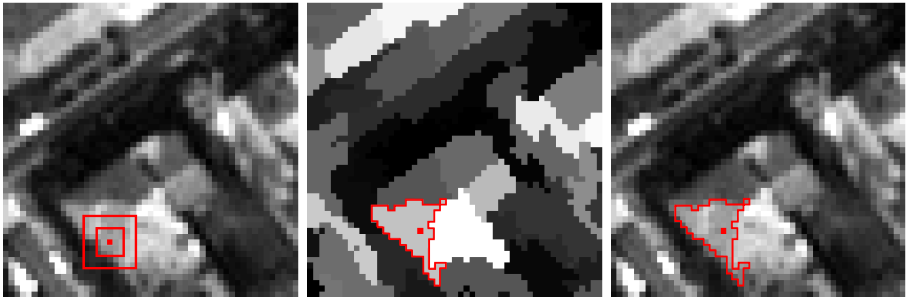


- ⋆ Self-complementarity: $\Psi = \mathbf{C}\Psi \Rightarrow$ each structure is processed equally.
- ⋆ Area filter: Removes small structures (area = number of pixels).
- ⋆ Algorithm:
  1. Label all the flat zones that satisfy the area criterion λ,
  2. Grow the labelled flat zones until a partition of the image is reached.



Original $\lambda = 10$ $\lambda = 30$ $\lambda = 40$

- Extract the inter-pixel dependency Υ:



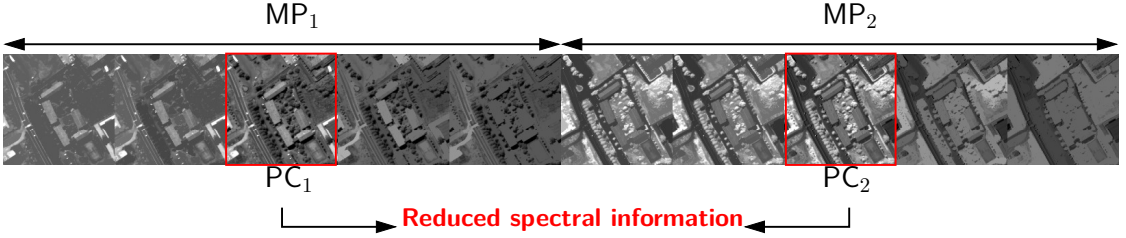
$\mathbf{x}$ $\lambda = 30$ $\Omega_{\mathbf{x}}$

- $\Upsilon_{\mathbf{x}} = \text{median}(\Omega_{\mathbf{x}})$:
  - Structure: What are the pixels related to $\mathbf{x}$?
  - Contrast: Local gray-level distribution

Definition (Extended Morphological Profile)

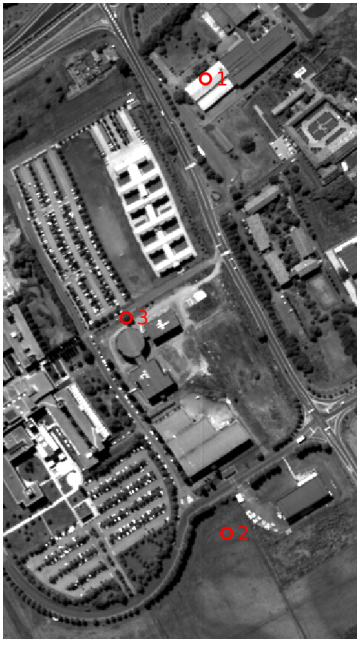
The EMP of size $n \times p$ is a $(2n+1)p$-dimensional vector made of the MP build with the p first principal components:

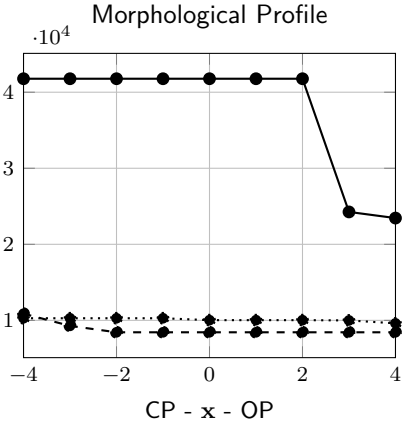
$$\mathrm{EMP}(\mathbf{x}) = \Big[\mathrm{MP}_1(\mathbf{x}), \ldots, \mathrm{MP}_p(\mathbf{x})\Big].$$



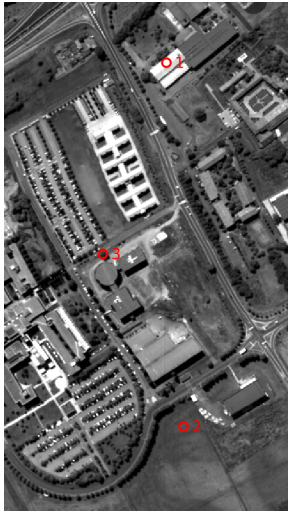
- ⋆ Fusion of morphological and spectral features
- ⋆ PCA, FDA, Kernel-PCA ...
- ⋆ *Same methods for self-complementary area filter*

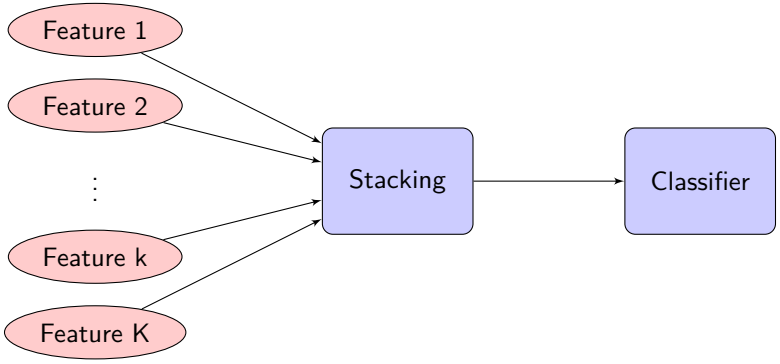
```
    # Start the computation
    for i in xrange(no):
        # Structuring elements
        se = disk(radius+i*2)

        # Compute opening per reconstruction
        temp = erosion(im,se)
        out[:,:,no+1+i] = reconstruction(temp,im,method='dilation')

        # Compute closing per reconstruction
        temp = dilation(im,se)
        out[:,:,no-1-i] = reconstruction(temp,im,method='erosion')

    return out
if __name__ == '__main__':
    # Load image
    im,GeoT,Proj = rt.open_data('../Data/pca_university.tif')

    # Apply the Morphological profile on each PC
    EMP = []
    for i in xrange(3):
        EMP.append(morphological_profile(im[:,:,i]))
    EMP = sp.concatenate(EMP,axis=2)
    rt.write_data("../Data/emp_pca_university.tif",EMP,GeoT,Proj)
```


1
3
2


Morphological Profile
$\cdot 10^4$
4
3
2
1
−4
−2
0
2
4
CP - x - OP

Where is the *closing* and the *opening* ?

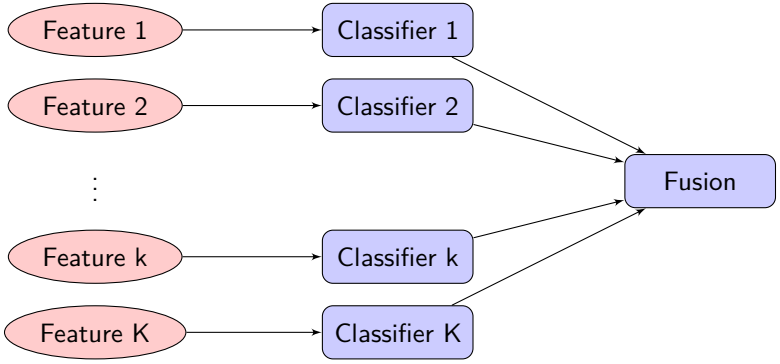



Feature 1
Feature 2
Feature k
Feature K
Stacking
Classifier

- Extract several spatial descriptors
  - EMP,
  - Texture,
  - Histogram of oriented gradients (HOG),
  - ...
- *Optional*: apply feature extraction
  - Spectral features,
  - Spatial features,
  - Both
- Stack all the features into a "big vector"

- In [Fau+08]:
  - Extrat EMP
  - Apply PCA/LDA to the spectral and spatial features
  - Stack the first PCs of spectral/spatial feautres
  - Classification with SVM

Method	# Features	OA
Spectral	103	79.5
EMP	27	79.1
S+EMP	130	83.5
S-DBFE+EMP-DBFE	27+10	88.0


Feature 1
Feature 2
Feature k
Feature K
Classifier 1
Classifier 2
Classifier k
Classifier K
Fusion

- Fusion of classifier outputs:
  - At the decision level

$$C_1 : \{y_1\};$$
$$C_2 : \{y_2\};$$
$$\vdots$$
$$C_K : \{y_K\}$$

  - At the membership level

$$C_1 : \{m_{11}, \ldots, m_{1C}\};$$
$$C_2 : \{m_{21}, \ldots, m_{2C}\};$$
$$\vdots$$
$$C_K : \{m_{K1}, \ldots, m_{KC}\}$$

- Decision level: Majority vote
- Membership level: Probabilistics methods, fuzzy logic, Dempster-Shafer ...

- In [FCB06]: Fusion of SVM
- Use the distance to the hyperplane
- *Absolute maximum* fusion rule
- Two classifiers with different intputs: Spectral and EMP

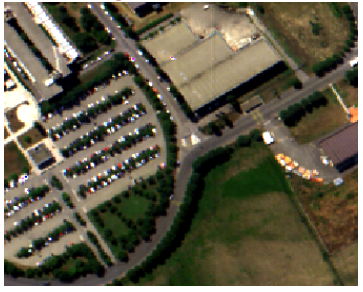
Feature	OA
Spectral	81.0
EMP	85.2
Output fusion	89.6

- ⋆ Simple to implement:

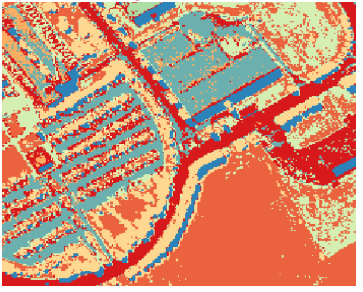
```
# Concatenate the spectral and spatial features and do scaling
IM_EMP = sp.concatenate((im[:,::2],EMP.astype(im.dtype)),axis=1)
```

- ⋆ Good classification accuracy: $F1 = 0.99$
- ⋆ sptial

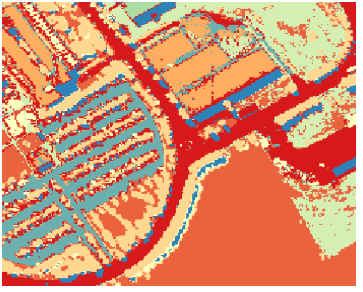
Color Image



Spectral only



Spectral + EMP

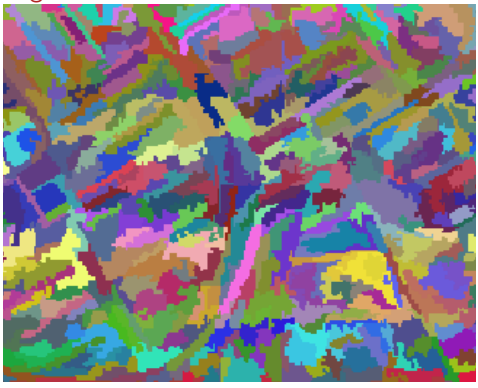


- Main ideas
  - Segmentation of the image: partition the image into non-overlapping homogeneous zones
  - Spatial regularization of the thematic map
- Issues:
  - Segmentation of hyperspectral images is tricky !
  - Spatial regularization
- From [Fau+13]:
  - Segmentation:
    - Image processing: Watershed, region growing, mean-shift, ...
    - Statistical: GMM, K-means ...
  - Regularization:
    - Majority voting,
    - Region growing

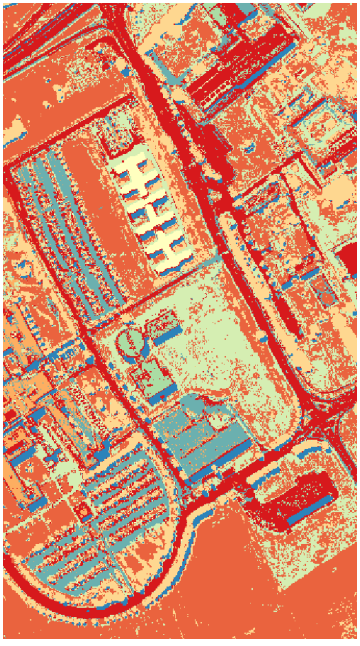
```
# Using Mean Shift
otbcli_Segmentation -in ../Data/pca_university.tif -mode raster -mode.raster.out ../Data/mean_shift_unive
    -filter.meanshift.minsize 50
```

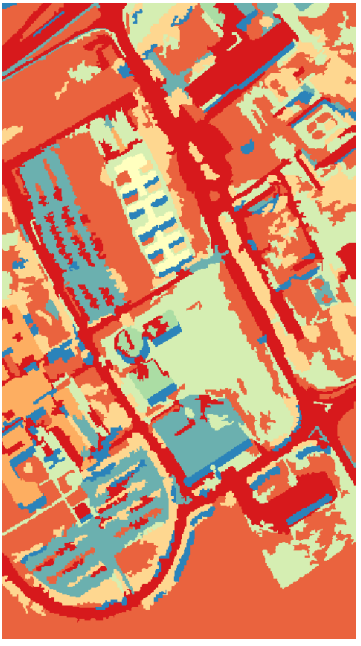
Color Image

Segmented

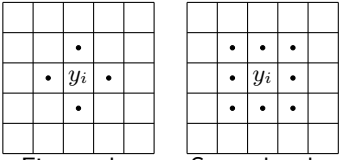


```
import rasterTools as rt
import scipy as sp
from scipy.stats import mode

# Load Thematic Map
im,GeoT,Proj = rt.open_data('../Data/tm_university_svm.tif')
out = sp.empty_like(im)

# Load segmented image
segmented,GeoT,Proj = rt.open_data('../Data/mean_shift_university.tif')

# Do the majority vote
for l in sp.unique(segmented):
    t = sp.where(segmented==l)
    y = im[t]
    out[t] = mode(y, axis=None)[0][0]

# Write the new image
rt.write_data("../Data/tm_university_fusion_mv.tif",out,GeoT,Proj)
```





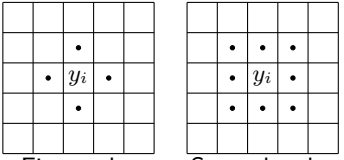
- ⋆ Markovian hypothesis/condition [MSB13]: $p(y_i = c|\mathbf{x}_i, \mathcal{N}_i)$
- ⋆ $\mathcal{N}_i$: neighborhood of pixel i



First-order Second order

- ⋆ When Y is a Markov Random Field:
  - ⋆ $P(Y|\mathbf{X}) \propto \exp(-U(Y|\mathbf{X}))$
  - ⋆ $U(Y|\mathbf{X}) = \sum_{i=1}^{n} U(y_i|\mathbf{x}_i, \mathcal{N}_i)$
  - ⋆ $U(y_i|\mathbf{x}_i, \mathcal{N}_i) = \Omega(\mathbf{x}_i, y_i) + \beta\mathcal{E}(y_i, \mathcal{N}_i)$
- ⋆ Spectral term: $\Omega(\mathbf{x}_i, y_i) = -\log[p(\mathbf{x}_i|y_i)]$
- ⋆ Spatial term (*Potts model*): $\mathcal{E}(y_i, \mathcal{N}_i) = \sum_{j \in \mathcal{N}_i}[1 - \delta(y_i, y_j)]$

- ⋆ Markovian hypothesis/condition [MSB13]: $p(y_i = c|\mathbf{x}_i, \mathcal{N}_i)$
- ⋆ $\mathcal{N}_i$: neighborhood of pixel i



First-order Second order

- ⋆ When Y is a Markov Random Field:
  - ⋆ $P(Y|\mathbf{X}) \propto \exp(-U(Y|\mathbf{X}))$
  - ⋆ $U(Y|\mathbf{X}) = \sum_{i=1}^{n} U(y_i|\mathbf{x}_i, \mathcal{N}_i)$
  - ⋆ $U(y_i|\mathbf{x}_i, \mathcal{N}_i) = \Omega(\mathbf{x}_i, y_i) + \beta\mathcal{E}(y_i, \mathcal{N}_i)$
- ⋆ Spectral term: $\Omega(\mathbf{x}_i, y_i) = -\log[p(\mathbf{x}_i|y_i)]$
- ⋆ Spatial term (*Potts model*): $\mathcal{E}(y_i, \mathcal{N}_i) = \sum_{j \in \mathcal{N}_i}[1 - \delta(y_i, y_j)]$
- ⋆ Function to be optimized

$$U(Y|\mathbf{X}) = \sum_{i=1}^{n} \Big\{ -\log[p(\mathbf{x}_i|y_i)] + \beta \sum_{j \in \mathcal{N}_i} [1 - \delta(y_i, y_j)] \Big\}$$

- Global optimization is not tractable [Li09]: iteration of local optimization on

$$-\log[p(\mathbf{x}_i|y_i)] + \beta \sum_{j \in \mathcal{N}_i} [1 - \delta(y_i, y_j)]$$

- Iterated conditional mode:
  - Scan all the pixels: change the label to maximize the local energy
  - Stop when convergences is reached
- Advanced algorithms
  - Simulated annealing [Tar+10]
  - Graph-cut [MSB13]

- Global optimization is not tractable [Li09]: iteration of local optimization on

$$-\log[p(\mathbf{x}_i|y_i)] + \beta \sum_{j \in \mathcal{N}_i} [1 - \delta(y_i, y_j)]$$

- Iterated conditional mode:
  - Scan all the pixels: change the label to maximize the local energy
  - Stop when convergences is reached
- Advanced algorithms
  - Simulated annealing [Tar+10]
  - Graph-cut [MSB13]
- For hyperspectral images: needs classification algorithms robust to the dimensionality!

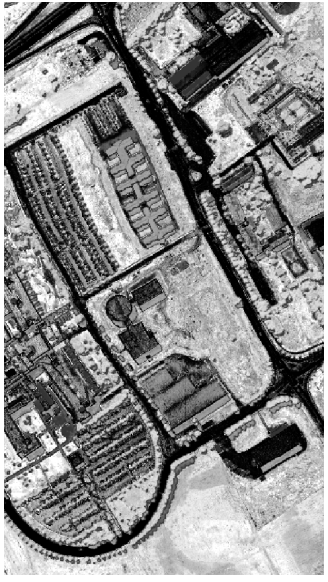
- ICM main loop:

```
# Iterate until convergence
while (diff[-1] > th) and (niter < 100):
    old_labels= labels.copy() # Make a copy of the old labels
    for i in xrange(1,nl-1): # Scan each line
        for j in xrange(1,nc-1): # Scan each column
            energy = []
            labels_ = old_labels[i-1:i+2,j-1:j+2].copy()
            for c in xrange(C): # Compute the energy for the different classes
                labels_[1,1] = c+1
                energy.append(compute_energy(proba[i,j,c],labels_,beta))
            arg = sp.argmin(energy) # Get the maximum energy term for the local configuration
            labels[i,j] = arg + 1
    diff.append(1 - sp.sum(old_labels == labels ).astype(float)/nc/nl) # Compute the changes
    niter += 1
# Clean data
del old_labels
return diff
```

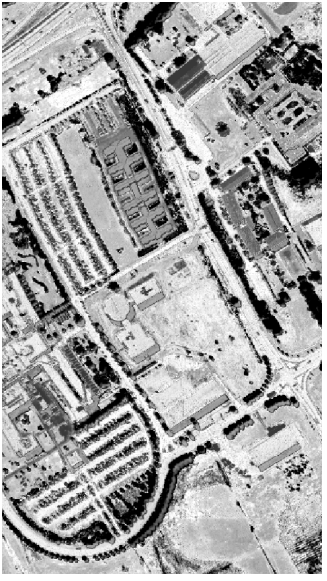
- Ask SVM for probability outputs

```
clf.probability= True
clf.fit(X_train,y_train)

# Predict the whole image and the probability map
labels = clf.predict(im).reshape(h,w)
proba = -clf.predict_log_proba(im).reshape(h,w,y.max())
```

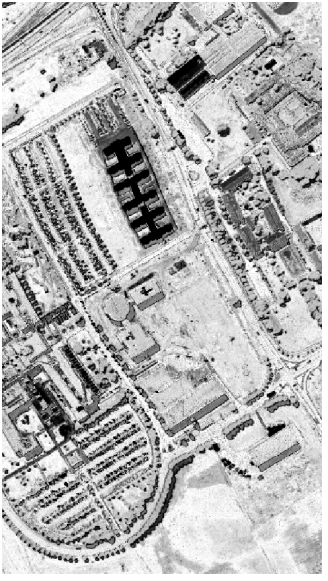
Asphalt



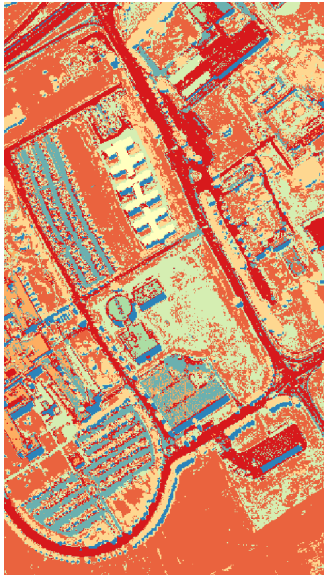
Tree



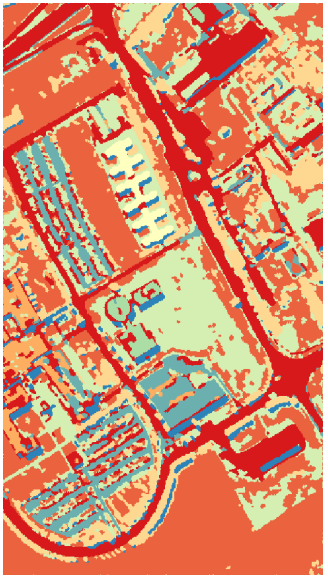
Metal sheet



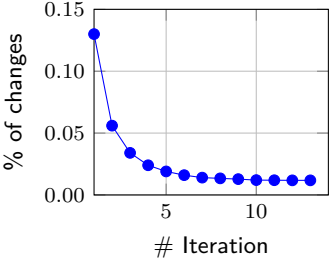
Spectral only



MRF

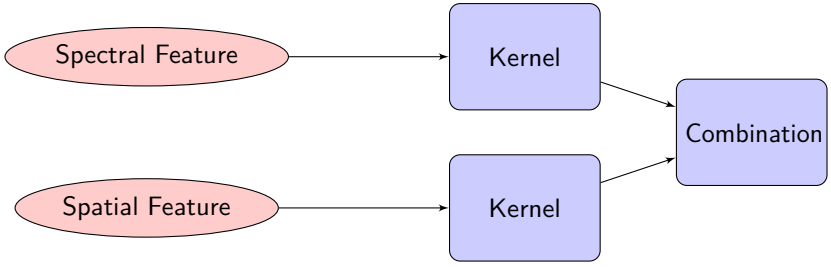


Iteration



- Let k_1 and k_2 be positive semi-definite, and $\lambda_{1,2} > 0$ then:
  1. $\lambda_1 k_1$ is a valid kernel
  2. $\lambda_1 k_1 + \lambda_2 k_2$ is positive semi-definite.
  3. $k_1 k_2$ is positive semi-definite.
  4. $\exp(k_1)$ is positive semi-definite.
  5. $g(\mathbf{x}_i)g(\mathbf{x}_j)$ is positive semi-definite, with $g : \mathbb{R}^d \to \mathbb{R}$.
- A kernel is usually seen as a measure of similarity between two samples. It reflects in some sens, how two samples are similar.

- Let k_1 and k_2 be positive semi-definite, and $\lambda_{1,2} > 0$ then:
  1. $\lambda_1 k_1$ is a valid kernel
  2. $\lambda_1 k_1 + \lambda_2 k_2$ is positive semi-definite.
  3. $k_1 k_2$ is positive semi-definite.
  4. $\exp(k_1)$ is positive semi-definite.
  5. $g(\mathbf{x}_i)g(\mathbf{x}_j)$ is positive semi-definite, with $g : \mathbb{R}^d \to \mathbb{R}$.
- A kernel is usually seen as a measure of similarity between two samples. It reflects in some sens, how two samples are similar.
- In image classification. It is possible to build kernels that includes information from the spatial domain.
  - Local correlation
  - Spatial position
  - Morphological feature,
  - ...

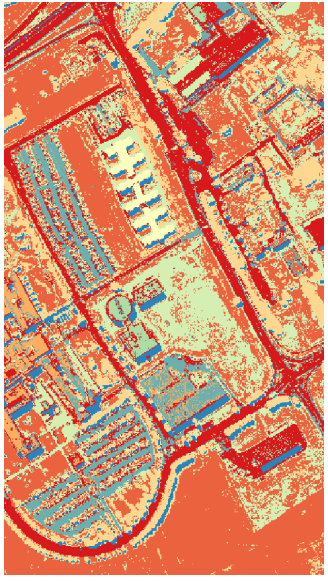


- From [Cam+06]:
  - Feature fusion: $k_{\text{spatial+spectral}}$
  - Direct summation: $k_{\text{spatial}} + k_{\text{spectral}}$
  - Weighted summation: $\mu k_{\text{spatial}} + (1-\mu) k_{\text{spectral}}$, $0 \leq \mu \leq 1$
- Can be extended to more than two kernels: *multiple kernel learning* (tricky)

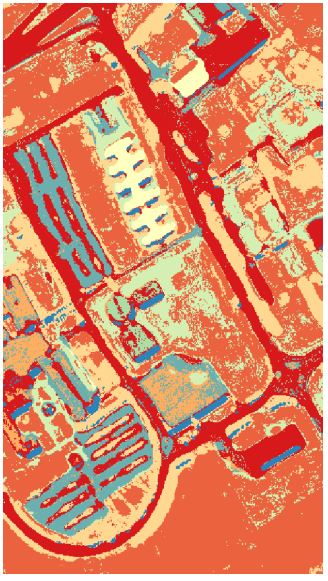
- Combination of
  - Spectral bands
  - Spatial features: local median computed on a moving window
- Weighted summation kernel + SVM

```
# Create a pipeline
pipe = Pipeline([
    ('CK',CompositeKernel()),
    ('SVM',SVC())
])

# Optimize parameters
cv_params = dict([
    ('CK__gamma', 2.0**sp.arange(-3,3)),
    ('CK__mu', sp.linspace(0,1,num=11)),
    ('SVM__kernel', ['precomputed']),
    ])
```

Spectral only



Composite kernel



Parameters

- ⋆ F1 $= 0.89$
- ⋆ $\mu = 0.8$

Camps-Valls, G. et al. "Composite kernels for hyperspectral image classification". In: *IEEE Geoscience and Remote Sensing Letters* 3.1 (Jan. 2006), pp. 93–97. ISSN: 1545-598X. DOI: 10.1109/LGRS.2005.857031.

Fauvel, Mathieu, Jocelyn Chanussot, and Jon Atli Benediktsson. "A combined support vector machines classification based on decision fusion". In: *Proc. IEEE Intl. Geoscience and Remote Sensing Symposium*. 2006, pp. 2494–2497.

Fauvel, Mathieu et al. "Advances in spectral-spatial classification of hyperspectral images". In: *Proceedings of the IEEE* 101.3 (2013), pp. 652–675.

Fauvel, Mathieu et al. "Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles". In: *Geoscience and Remote Sensing, IEEE Transactions on* 46.11 (2008), pp. 3804–3814.

Haralick, R. M., K. Shanmugam, and I. Dinstein. "Textural Features for Image Classification". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-3.6 (Nov. 1973), pp. 610–621. ISSN: 0018-9472. DOI: 10.1109/TSMC.1973.4309314.

Li, Stan Z. *Markov Random Field Modeling in Image Analysis*. 3rd. Springer Publishing Company, Incorporated, 2009. ISBN: 9781848002784.

Mercier, G. and M. Lennon. "On the characterization of hyperspectral texture". In: *IEEE International Geoscience and Remote Sensing Symposium*. Vol. 5. 2002, 2584–2586 vol.5. DOI: 10.1109/IGARSS.2002.1026708.

Moser, G., S. B. Serpico, and J. A. Benediktsson. "Land-Cover Mapping by Markov Modeling of Spatial-Contextual Information in Very-High-Resolution Remote Sensing Images". In: *Proceedings of the IEEE* 101.3 (Mar. 2013), pp. 631–651. ISSN: 0018-9219. DOI: 10.1109/JPROC.2012.2211551.

Tarabalka, Yuliya et al. "SVM-and MRF-based method for accurate classification of hyperspectral images". In: *IEEE Geoscience and Remote Sensing Letters* 7.4 (2010), pp. 736–740.