

Multivariate Statistical Methods for Big Data Analysis and Process Improvement

Instructor: Dr. Brandon Corbett

Lecture 10 for ChE 765 | Sep 767, McMaster
University

Image analysis aided by multivariate methods

We will look at

1. Multivariate image analysis (MIA)
2. Also show how classical image analysis tools are combined with multivariate tools, especially PLS

Credits: MACC group: particularly

- ▶ Honglu Yu
- ▶ J. Jay Liu

We will study some applications from their PhD theses today.

References

- ▶ General image analysis book – Gonzalez and Woods: “Digital Image Processing”
- ▶ Good background: [Multivariate image analysis: A review with applications](#)
- ▶ General MIA technique: [Geladi and Grahn](#), 1997 book
- ▶ Paper: [Esbensen and Geladi](#) - Strategy of MIA

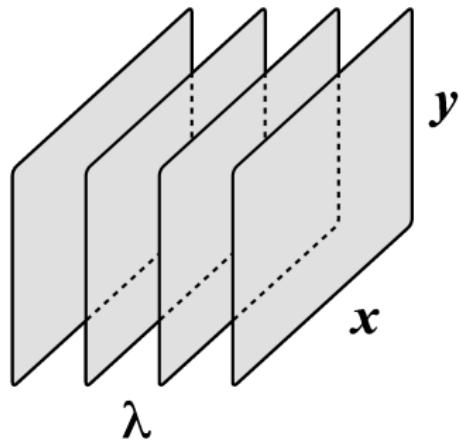
McMaster student applications:

- ▶ Lumber, pulp, steel: [Manish Bharati's PhD thesis](#), 2002
- ▶ Snackfoods and flames: [Honglu Yu's PhD thesis](#), 2003
- ▶ Texture, wavelets, appearance monitoring and optimization: [June Liu's PhD thesis](#), 2004
- ▶ Medical images: [Mark-John Bruwer's PhD thesis](#), 2006
- ▶ Wheat grading with NIR: [Zheng Liu's Masters thesis](#), 2006
- ▶ Oats/groats detection: [Emily Nichols' Masters thesis](#), 2011

Reasons for using image data

- ▶ Conventional sensors are expensive ($\sim \$ 5000$ or more), but digital cameras are relatively cheap
- ▶ Image sensors work well for solids-processing systems, which generally have fewer sensors available
- ▶ Cameras take non-destructive samples
- ▶ Can be implemented on-line
- ▶ Often picks up information not available from other sensors

Conceptual storage of image data



The wavelength dimension (spectral dimension):

- ▶ 1 channel: grayscale
- ▶ 3 channels: colour image, RGB image
- ▶ multiple channels: hyperspectral image (e.g. NIR camera)

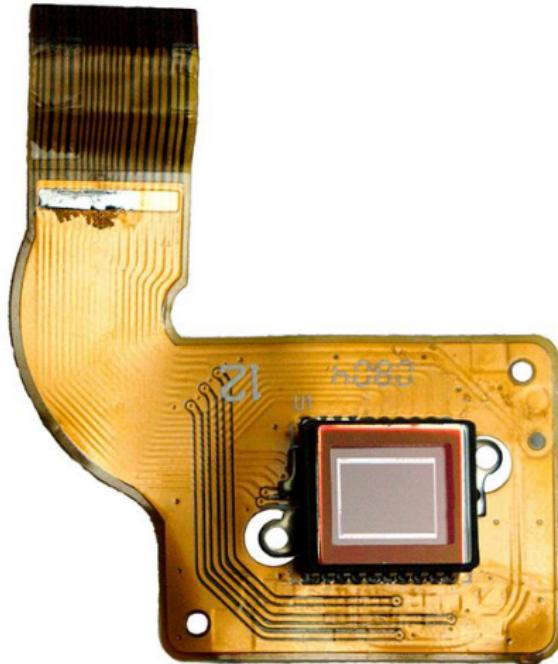
Tradeoff: *usually* we see lower spatial resolution as we acquire more channels, mainly due to cost of acquiring the additional wavelengths

What do we want from our images

We will see examples where we:

- ▶ Monitor for process problems
- ▶ Make predictions from the image data
- ▶ Optimize and improve our processes

How image data is acquired



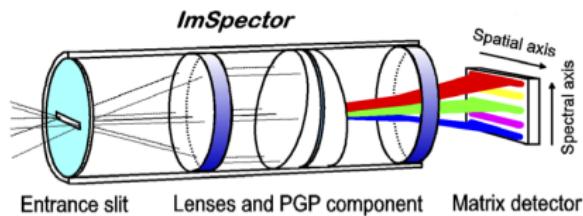
Charge-coupled device (CCD)
senses the light.

Photons received → voltage →
digitized and stored.

Image source: [Wikipedia](#) (CC-BY-SA)

How image data is acquired

NIR cameras (multiple channels)¹

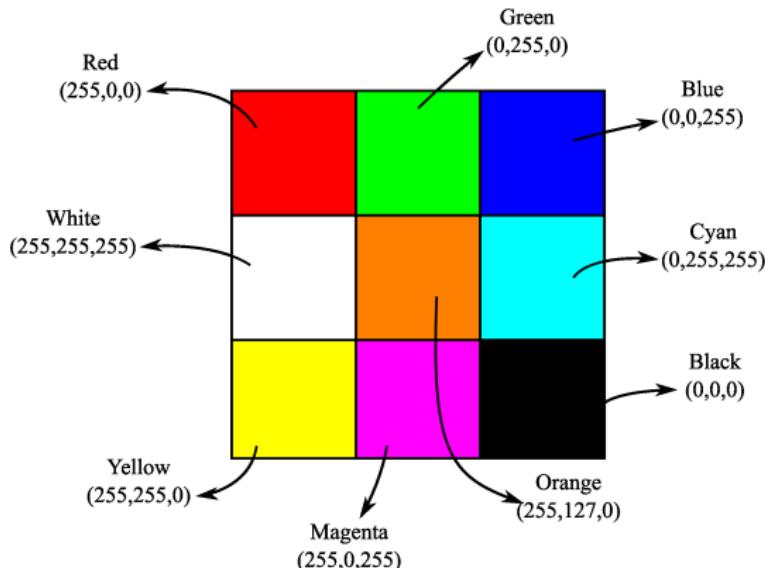


¹From Emily Nichols' thesis (used with permission)

Actual electronic storage of image data

Pixels in the image are stored as a sequence of (usually) integer values

- ▶ RGB pixel with (0, 0, 0) is a black pixel
- ▶ RGB pixel with (0, 0, 255) is a blue pixel
- ▶ RGB pixel with (255, 255, 255) is a white pixel

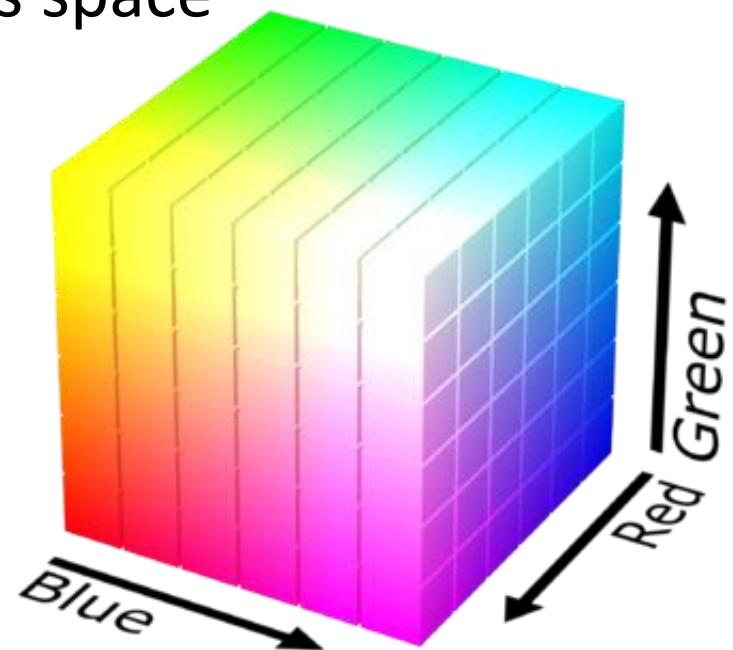


Color Spaces: RGB

- 99.9% of cameras will provide data in this space
- Most image file types use this space

Disadvantages:

- Hard to interpret values
- No value of “Brightness”
- Colors “change” at different rates



Color Spaces: LAB

Variables: Lightness, **a** and **b**

- Any similar change of **a** and **b** has similar visual impact.

Key Advantage:

- Lightness independent
- Large number of represented colors

Key Disadvantage:

- Difficult to calculate

$$L^* = 116 f\left(\frac{Y}{Y_n}\right) - 16$$

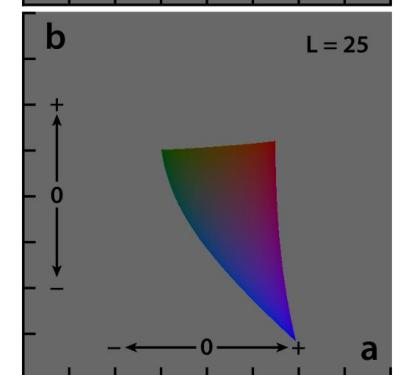
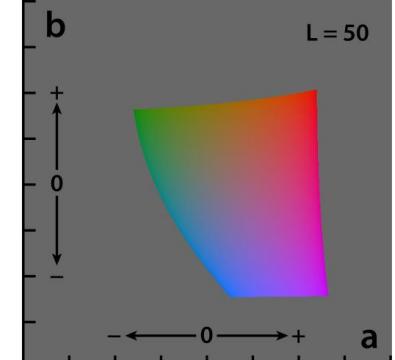
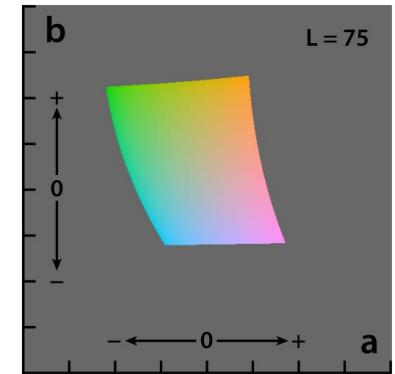
$$a^* = 500 \left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right)$$

$$b^* = 200 \left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right)$$

where

$$f(t) = \begin{cases} \sqrt[3]{t} & \text{if } t > \delta^3 \\ \frac{t}{3\delta^2} + \frac{4}{29} & \text{otherwise} \end{cases}$$

$$\delta = \frac{6}{29}$$



Color Spaces: HSV

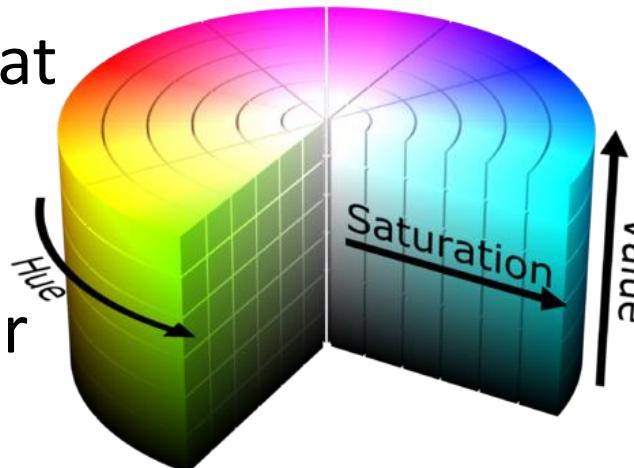
- A simple transformation of RGB space

Key advantages:

- Fast to calculate
- Meaningful colors (for masks)
- Lightness somewhat independent

Key disadvantages:

- Slightly worse color quantification



$$R' = R/255$$

$$G' = G/255$$

$$B' = B/255$$

$$C_{max} = \max(R', G', B')$$

$$C_{min} = \min(R', G', B')$$

$$\Delta = C_{max} - C_{min}$$

Hue calculation:

$$H = \begin{cases} 0^\circ & , C_{max} = R' \\ 60^\circ \times \left(\frac{G' - B'}{\Delta} \text{mod} 6 \right) & , C_{max} = G' \\ 60^\circ \times \left(\frac{B' - R'}{\Delta} + 2 \right) & , C_{max} = B' \\ 60^\circ \times \left(\frac{R' - G'}{\Delta} + 4 \right) & \end{cases}$$

Saturation calculation:

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

Value calculation:

$$V = C_{max}$$

Actual electronic storage of image data

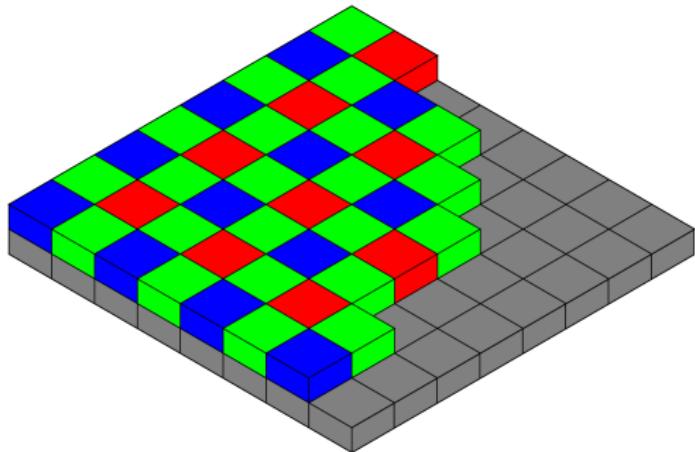
- ▶ Integers from 0 to 255 are stored in 8 bits (one byte) per pixel
- ▶ Medical images require greater spectral resolution: often use 2 bytes: 0 to 65535 (16 bits, but use only 10 or 12 bits)

Tip

Never store your image data as JPEG ← uses lossy compression;
rather store as PNG (lossless compression)

Not all colour images are true colour

True colour images split the light, then use 3 CCD arrays for the R, G, and B channels.



Everyday colour images (cellphones, digital cameras) are in fact from a single CCD with a special array of colour filters: Bayer filter

The R, G, B layers we see in the digital file are algorithmically created from the single CCD matrix.

Image source: [Wikipedia](#) (CC-BY-SA)

Viewing image data: grayscale

Computer monitors display RGB colour images.

Grayscale images displayed with equal weight in the 3 channels

Convert RGB to grayscale:

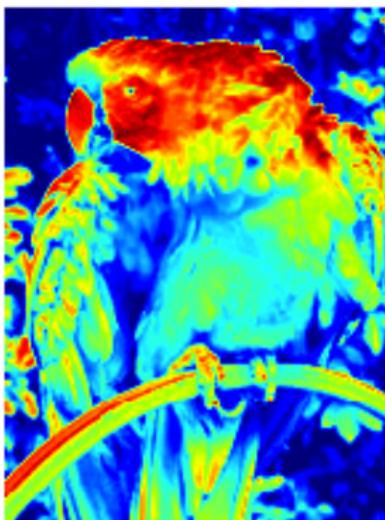
$$\text{Gray} = 0.3R + 0.59G + 0.11B$$

which approximates the
human eye's weighting of
our light-sensitive cones.



Viewing image data: grayscale

A grayscale image has values between 0 and 255 – map each integer to a different colour: `colormap()`

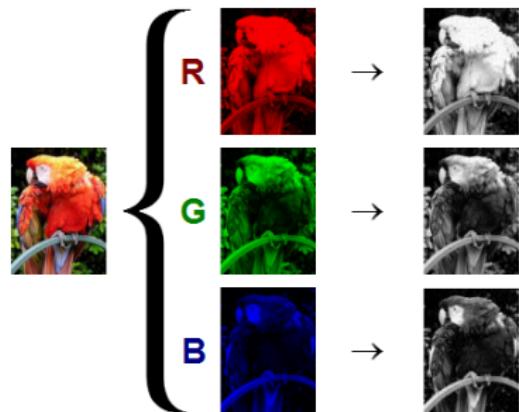


Creates a false-colour image from the grayscale image to help visualization. Many standard colour mappings are available.

Viewing image data: colour

Show each RGB channel individually, in only that channel (middle), or as grayscale (right)

- ▶ Low integer values will be dark
- ▶ High integer values will be light

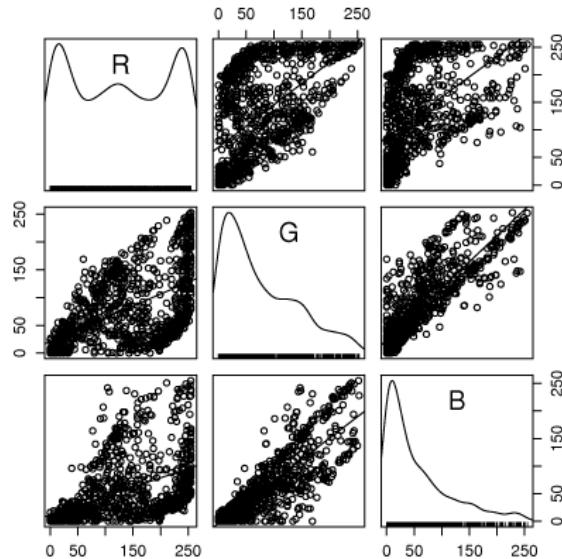


Viewing image data

Images with 4 or more wavelengths: create a *false-colour* composite image by picking any 3 channels.

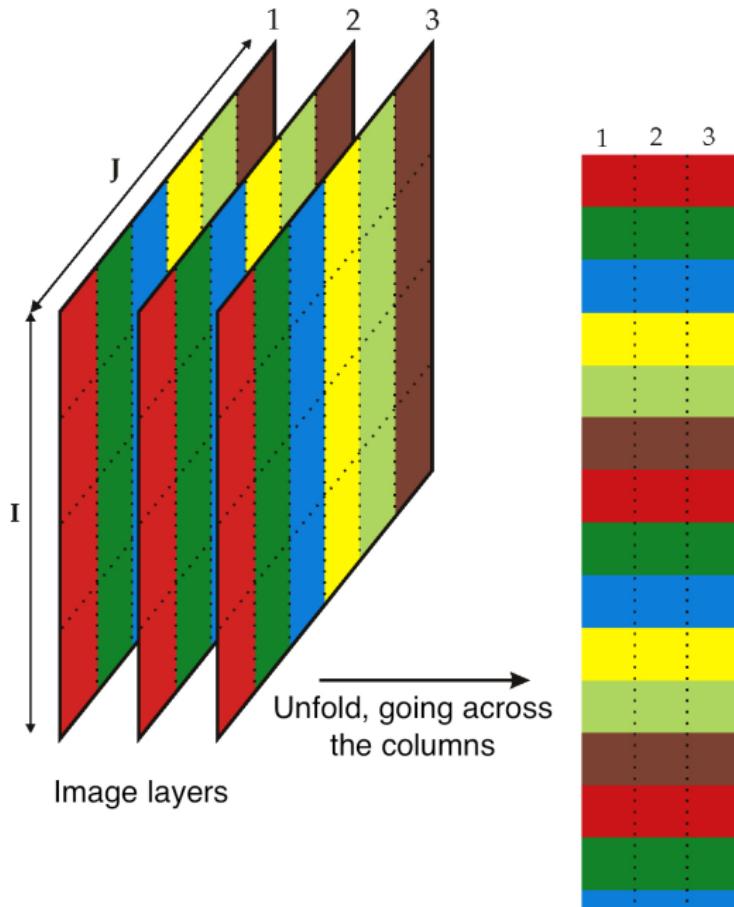
Using PCA for multivariate image analysis (MIA)

Multiple channels in the images are often strongly correlated, especially for NIR images



Strong green and blue interaction

Unfolding the image data

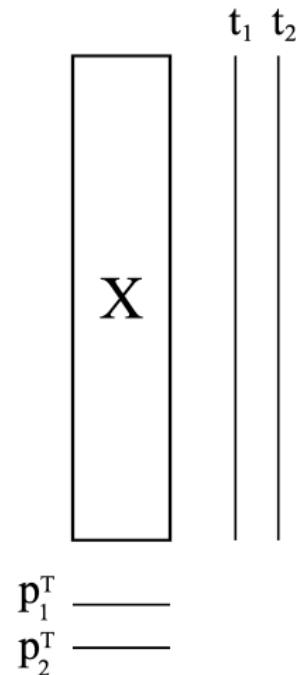


Other unfolding types possible, but this way makes sense to understand the correlation between the channels.

Sometimes called “*matricizing*” the image.

Calculating the PCA model

Once \mathbf{X} is unfolded, we have a long, thin matrix.



Use the kernel PCA method on unfolded image data \mathbf{X} :

$$\begin{aligned}\mathbf{P} &= \text{eigenvectors of } \mathbf{X}^T \mathbf{X} \\ \mathbf{T} &= \mathbf{XP}\end{aligned}$$

We may choose to use all $A = K$ eigenvectors in \mathbf{P} , or when K is large – e.g. NIR images – we only retain the first few ($A < K$):

$$\begin{aligned}\mathbf{X} &= \mathbf{TP}^T + \mathbf{E} \\ \mathbf{E}_a &= \mathbf{X}_{a-1} - \mathbf{t}_a \mathbf{p}_a^T \\ \mathbf{X}_{a-1} &= \text{original image data}\end{aligned}$$

We don't usually preprocess the image (there isn't much difference if you do, but it is expensive).

Using the PCA model

We can investigate the model in the usual way:

- ▶ loadings
- ▶ binary score plots, e.g t_1 vs t_2 : not practical
- ▶ SPE and Hotelling's T^2 (not practical)

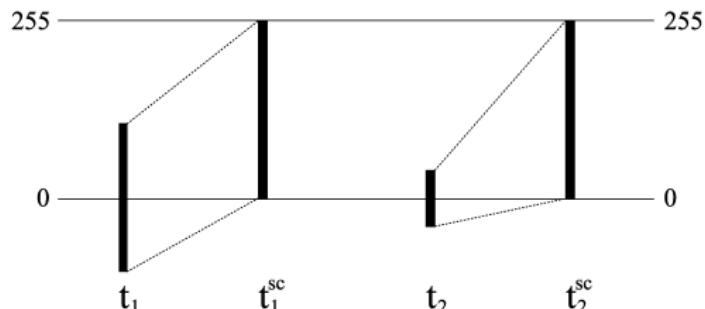
What do we want from the scores and SPE anyway?

Refolding the scores and residuals

Notice that each score t_a and each residuals in SPE have the same shape as the original image. It is natural to display these as grayscale images.

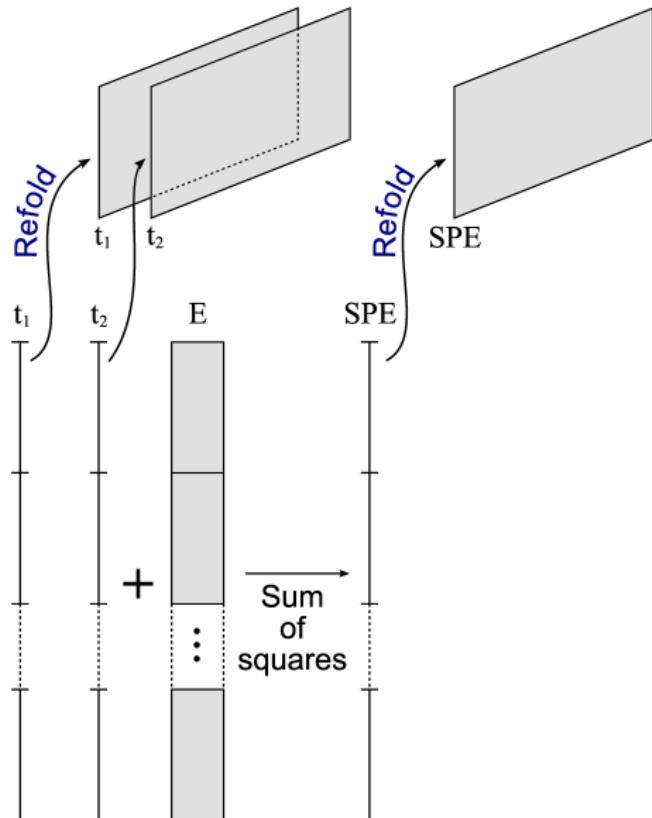
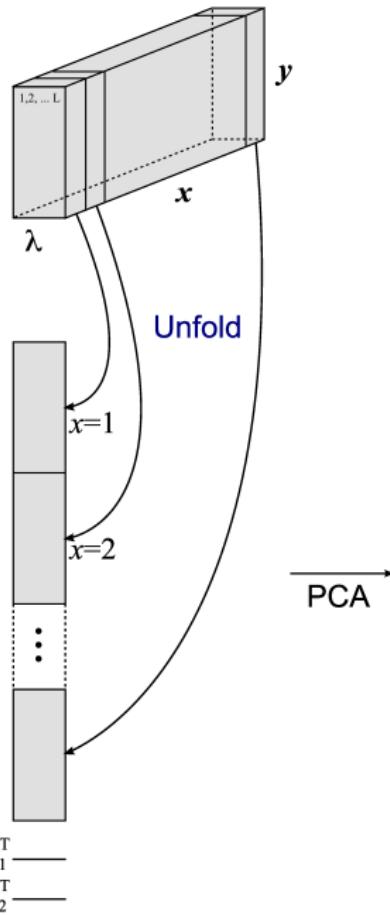
- ▶ grayscale displays require values between 0 and 255

The calculated t_a values have mean of zero. To display t_a^{sc} in the software:



$$t_a^{sc} = \left\lfloor 255 \cdot \frac{t_a - t_a^{\min}}{t_a^{\max} - t_a^{\min}} \right\rfloor$$

Rather display t_a^{sc} with colour coding.



Binary score plots

After scores are scaled as integers between 0 and 255:

- ▶ fewer unique combinations of t_h vs t_v
- ▶ show it as a 2D histogram instead
 - ▶ h = horizontal axis
 - ▶ v = vertical axis
 - ▶ height: use a colour proportional to number of pixels at (t_h, t_v) combination

Let's take a look at an example ...

An example

Original RGB image



Image is included with
MACCMIA

You can follow along in class

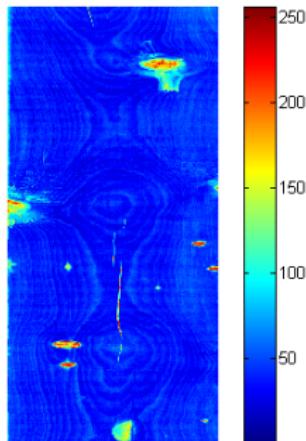
More details in **Bharati,
MacGregor and Tropper**

The first component

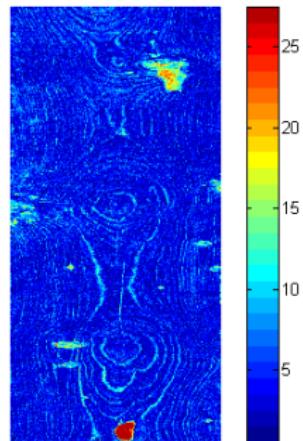
The RGB image from the data file



Intensity image of principal component 1



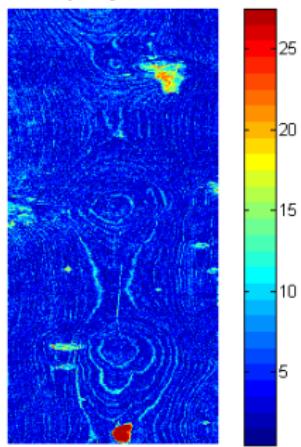
Residual intensity image after 1 PC extracted



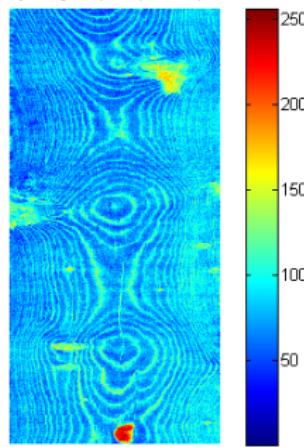
$$p_1 = [-0.72, -0.59, -0.37]$$

The second component

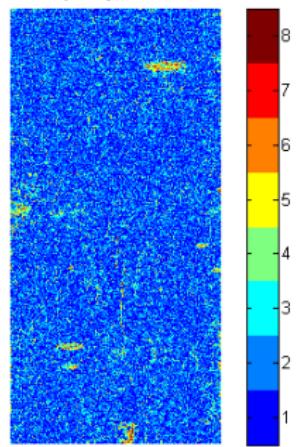
Residual intensity image after 1 PC extracted



Intensity image of principal component 2



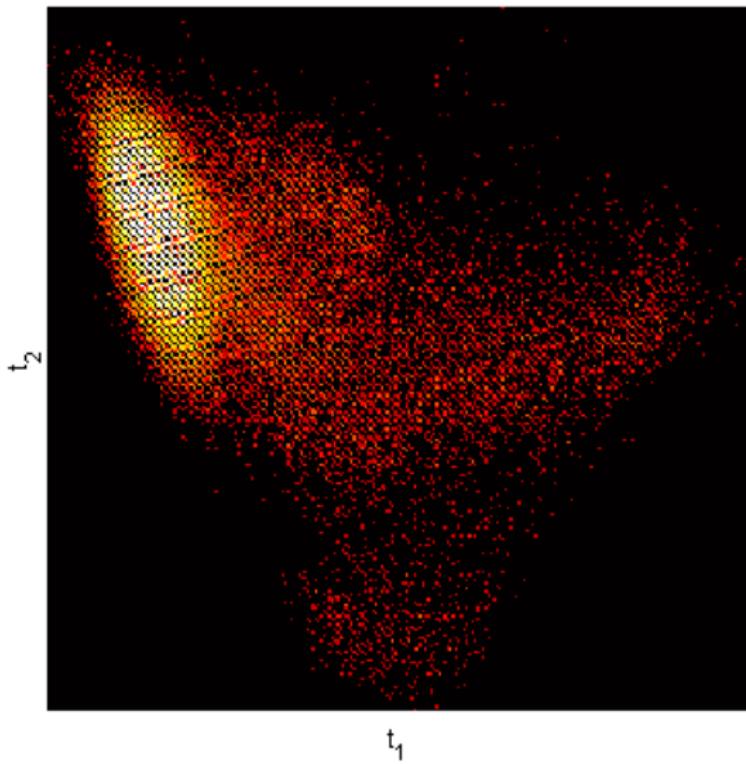
Residual intensity image after 2 PC's extracted



$$p_2 = [0.53, -0.13, -0.84]$$

Score plot

Score space density histogram

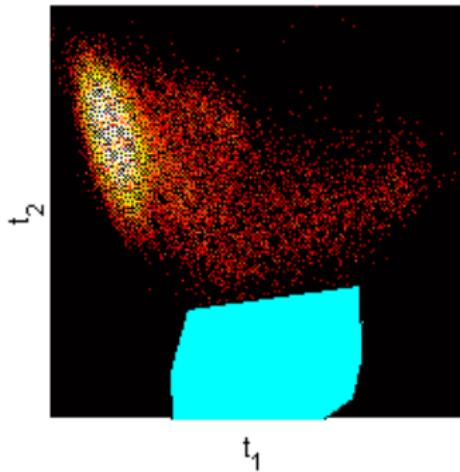


Masking in the score space

Image, with mask mapped back

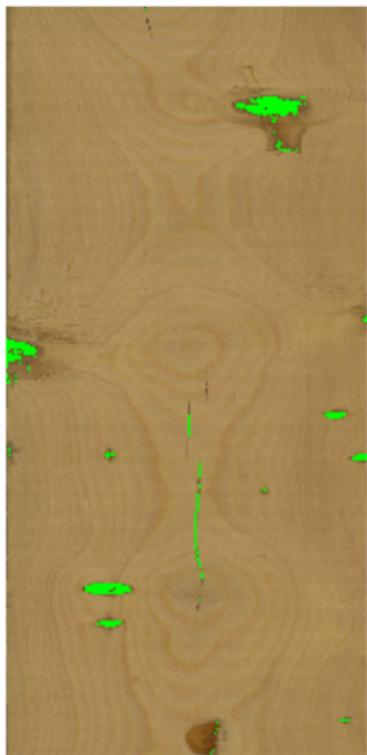


Score space density histogram

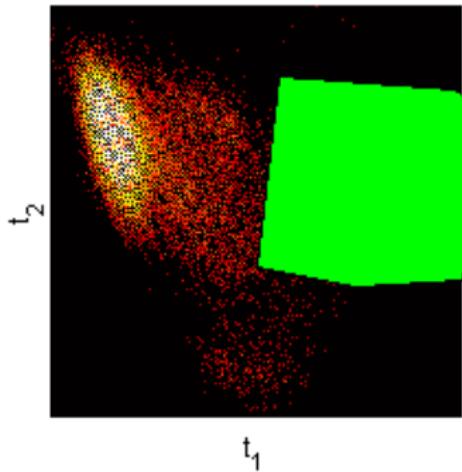


Masking in the score space

Image, with mask mapped back



Score space density histogram



Using the PCA model on a new image pixel

We have an existing PCA model; consists of

- ▶ loadings \mathbf{P} , an $K \times A$ matrix
- ▶ K = number of wavelengths (channels)
- ▶ t_a^{\min} and t_a^{\max}
- ▶ one or more score-space masks (store the vertices)

A new image (of different size, even as small as a single pixel):

- ▶ Unfold the image as \mathbf{X}_{new}
- ▶ \mathbf{X}_{new} has as many rows as there are pixels; has K columns
- ▶ $\mathbf{T}_{\text{new}} = \mathbf{X}_{\text{new}} \mathbf{P}$ = scores (has A columns)
- ▶ For each column in \mathbf{T}_{new} :
 - ▶ $\mathbf{t}_{a,\text{new}}^{\text{sc}} = \left[255 \cdot \frac{\mathbf{t}_{a,\text{new}} - t_a^{\min}}{t_a^{\max} - t_a^{\min}} \right]$
- ▶ For each row in $\mathbf{T}_{a,\text{new}}^{\text{sc}}$ check whether the new score values lie under a mask

Other hints and techniques

We have a problem if image space objects are spread across more than 2 different scores:

- ▶ Resolved using **automatic masking with support vector machines**

Building a model from multiple images?

1. Just paste images side-by-side to create a big, single image
2. More elegantly:

$$(\mathbf{X}^T \mathbf{X})_{\text{composite}} = (\mathbf{X}^T \mathbf{X})_1 + (\mathbf{X}^T \mathbf{X})_2 + \dots + (\mathbf{X}^T \mathbf{X})_{\text{last image}}$$

Calculate eigenvectors from $(\mathbf{X}^T \mathbf{X})_{\text{composite}}$

Principle of Multivariate Image Analysis

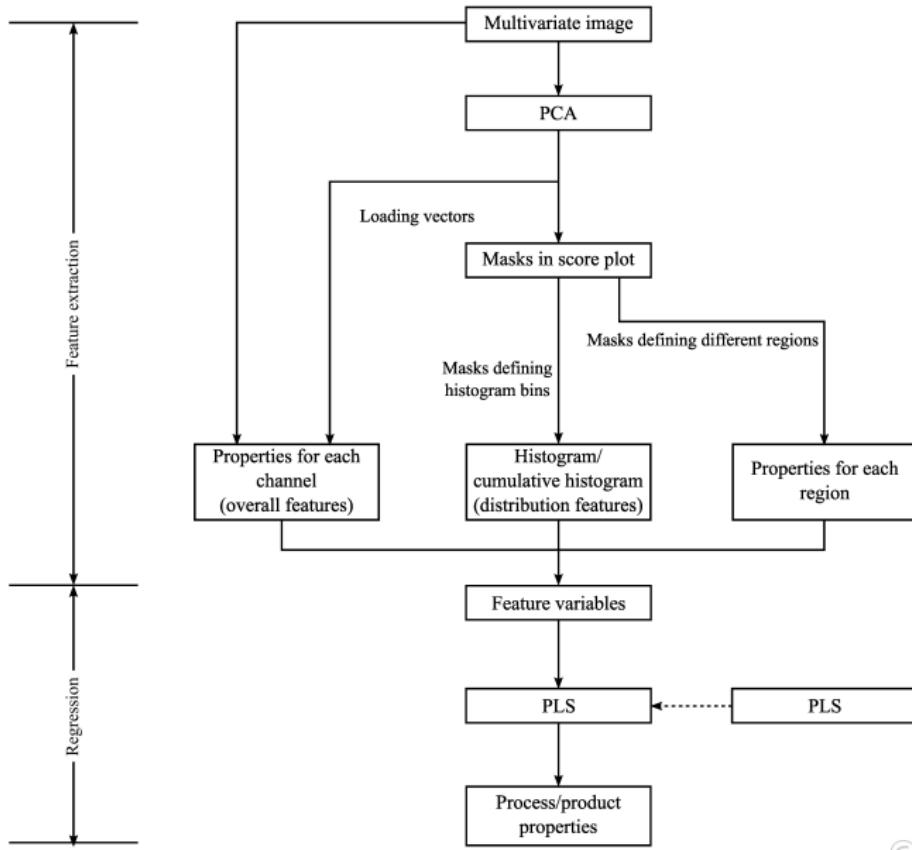
Multivariate Image Analysis (MIA)

Objects in the image with similar *spectral* signature, no matter where they are *spatially*, will appear clustered in the scores.

- ▶ There is a many-to-one mapping between the image space and score space
 - ▶ One location in the score space can appear in many spatial locations in the image space.

As a result, MIA is most useful when spatial information is not too important.

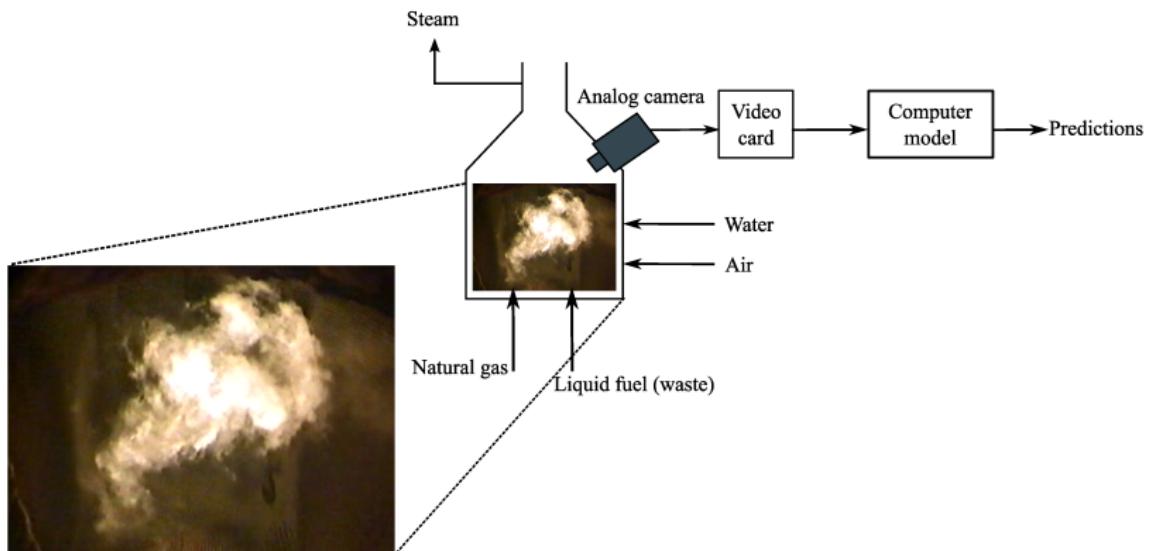
General strategy for MIA



Applications of image analysis

1. Flame monitoring and prediction (in depth)
 - ▶ Credit: thanks to Honglu Yu for permission to use her slides
 - ▶ **PhD thesis**
2. Snack food seasoning prediction, and monitoring (brief)
 - ▶ Credit: Honglu Yu
3. Automated judging steel sheets
 - ▶ Credit: thanks to J. Jay Liu for permission to use his slides
 - ▶ **PhD thesis**

Flame: system overview

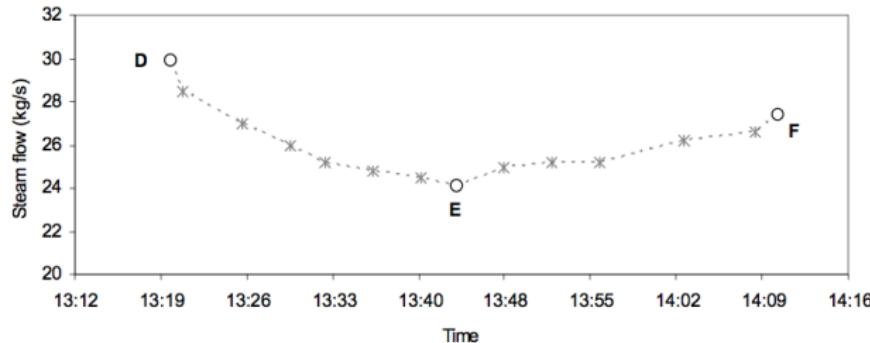
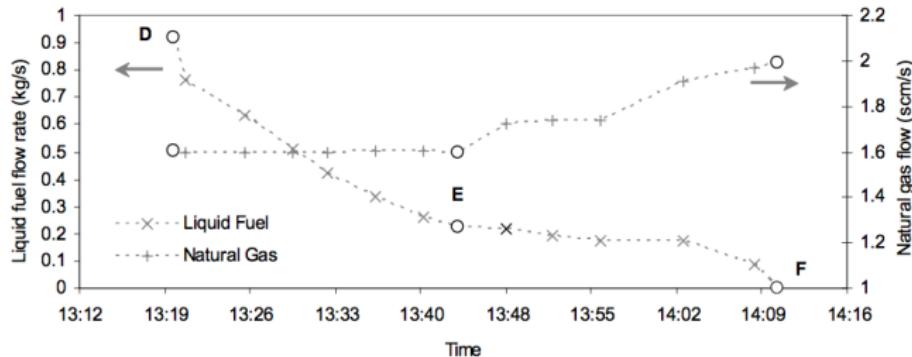


- ▶ Liquid waste stream has energy content
- ▶ Perhaps predict steam flow rate from image?

Data available

Process data: average value over ~ 3 minute interval

- ▶ steam flow (y variable)
- ▶ natural gas and waste stream flows



Data available

Video data: images digitized 1 second apart. Examples at D, E, F:



D



Liquid fuel: 0.917 kg/s
Natural Gas: 1.61 scm/s
Steam: 30.6 kg/s



E



Liquid fuel: 0.225 kg/s
Natural Gas: 1.58 scm/s
Steam: 24.1 kg/s



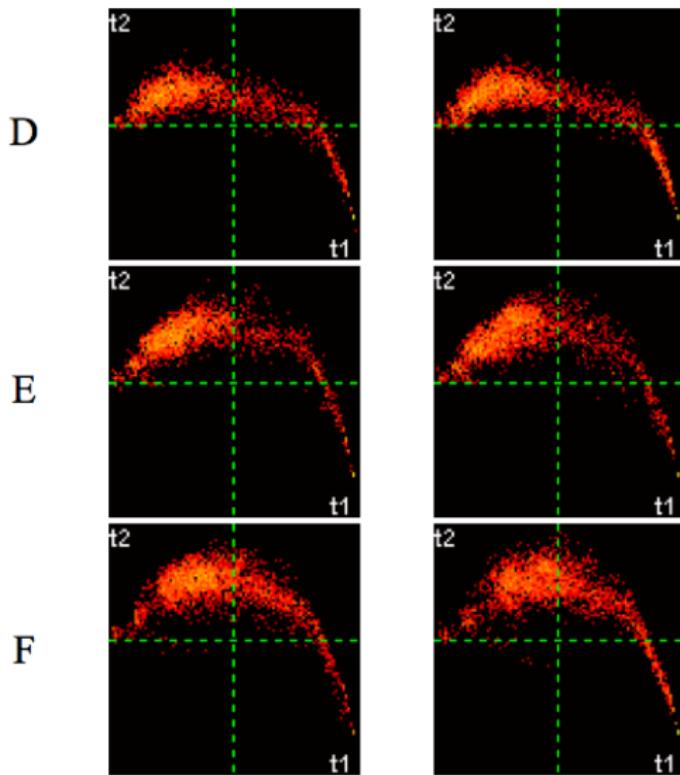
F



Liquid fuel: 0 kg/s
Natural Gas: 2 scm/s
Steam: 27.2 kg/s

- ▶ Each frame is $120 \times 160 \times 3 = 56$ kilobytes

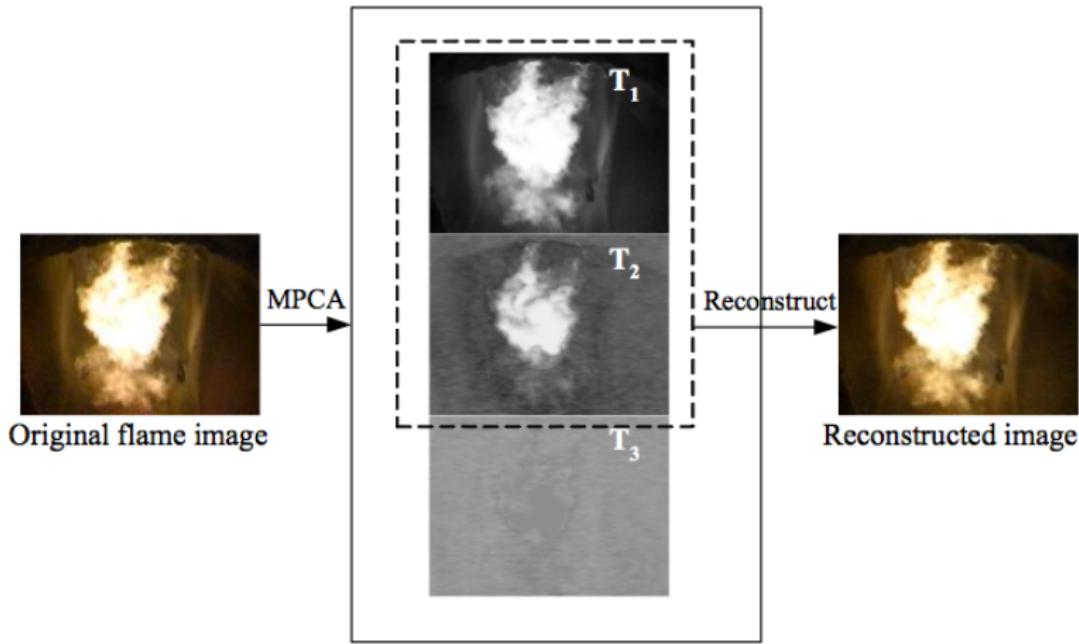
PCA on the flame images



Show video

PCA notes

- ▶ Use only 2 components: third PC mainly noise
- ▶ Reconstruct the denoised image: $\hat{\mathbf{X}} = \mathbf{t}_1\mathbf{p}_1^T + \mathbf{t}_2\mathbf{p}_2^T$



Important note

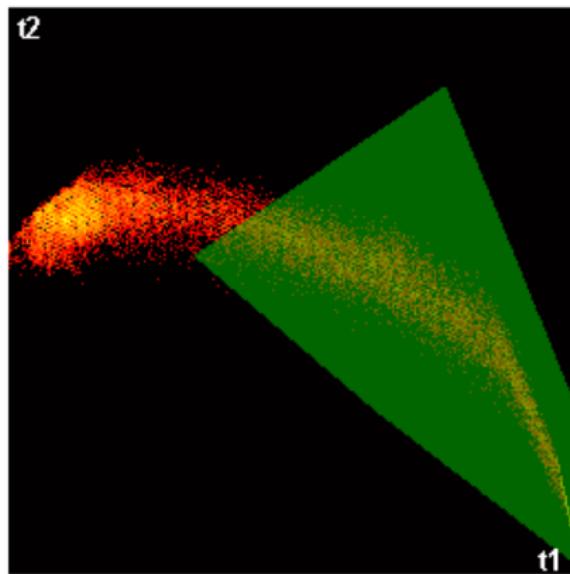
Based on the video and previous images:

- ▶ The image space *changes rapidly* from frame-to-frame
- ▶ Classical approaches to flame monitoring extract features directly from the image. For example:
 - ▶ perimeter
 - ▶ area
 - ▶ sphericity
 - ▶ luminosity
 - ▶ entropy (sum of absolute values)
 - ▶ maximum and minimum intensity
- ▶ The score space is much more stable
 - ▶ Use the score space to extract features

First segment the images

Def: *segmentation* – to divide the image into 2 or more regions

- ▶ Separate “flame” from “non-flame”
- ▶ Hard and slow to do in the image space (try it!)
- ▶ Much easier in the score space



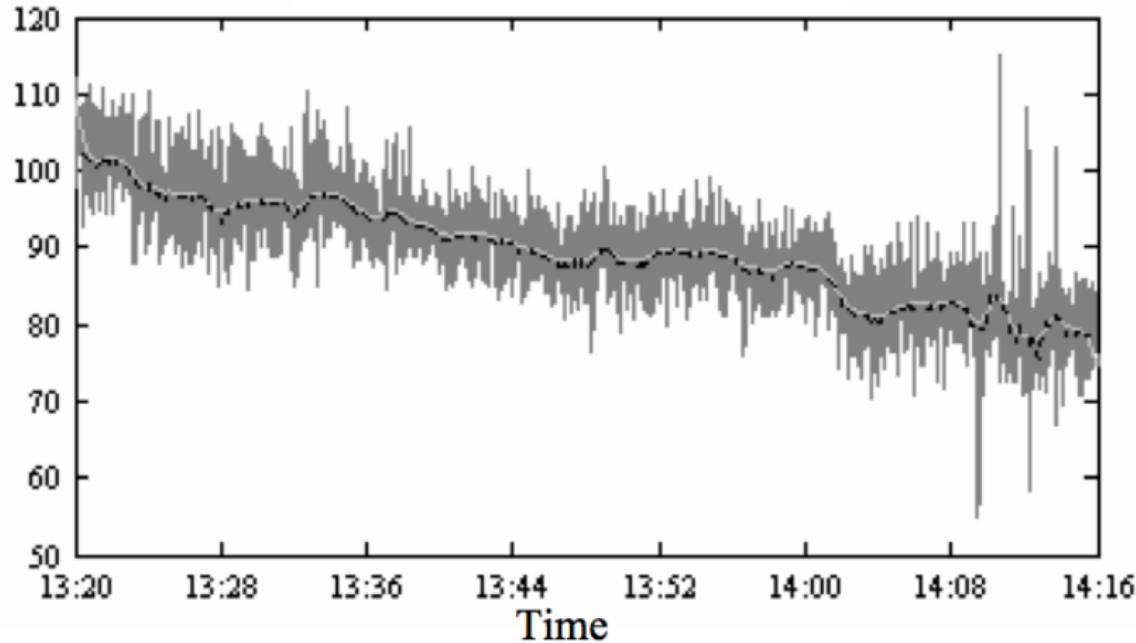
Features extracted by Honglu

Extract these features from the **entire image**:

- ▶ **A**: number pixels in flame region (area)
- ▶ **B**: average intensity of flame pixels after converting to grayscale (brightness)
- ▶ **U**: standard deviation of grayscale flame pixels (uniformity)
- ▶ **W**: average intensity of background pixels
- ▶ $\mathbf{s}_{1,m}, \mathbf{s}_{2,m} = [\overline{R}, \overline{G}, \overline{B}]$ $\mathbf{P} =$: average colour of entire image projected onto the PCA model
- ▶ $\mathbf{s}_{1,f}, \mathbf{s}_{2,f}$: as above, except for average flame colour
- ▶ **N_c**: number of unique colours in the flame region

i.e. generate a feature vector once per second

Stability of features?

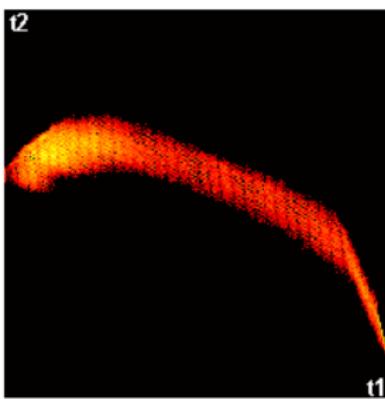


The features are not quite as stable as hoped for. Use smoothing

- ▶ dark-gray: feature from each image (noisy)
- ▶ light-gray: smooth the features after calculating
- ▶ black: smooth in the score space ... described next

Smoothing

1. Average 60 consecutive images, then calculate features (poor)
2. Calculate features, then use the average (e.g. MA, or EWMA)
3. Calculate 60 consecutive score images, *average in the score space*, then reconstruct image: $\hat{\mathbf{X}} = \mathbf{t}_1\mathbf{p}_1^T + \mathbf{t}_2\mathbf{p}_2^T$ and calculate features on reconstructed image.



Option 2 and 3 give similar performance (previous slide)

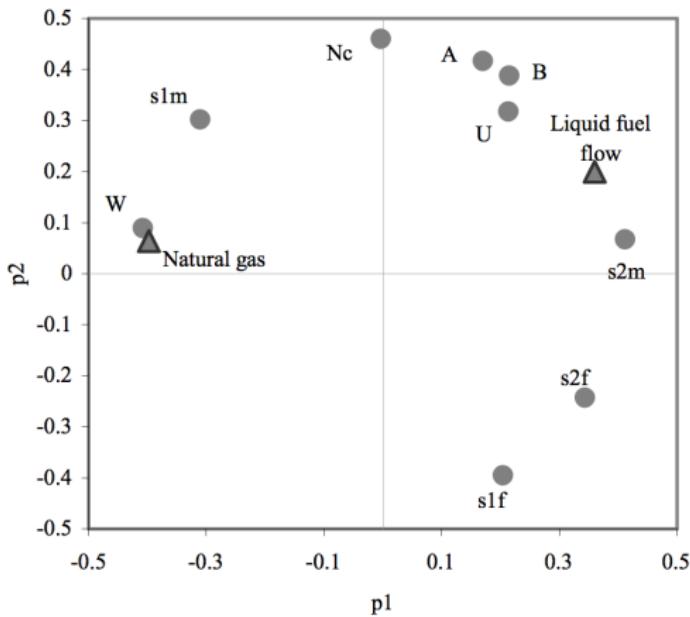
PCA on the features

Waste fuel:

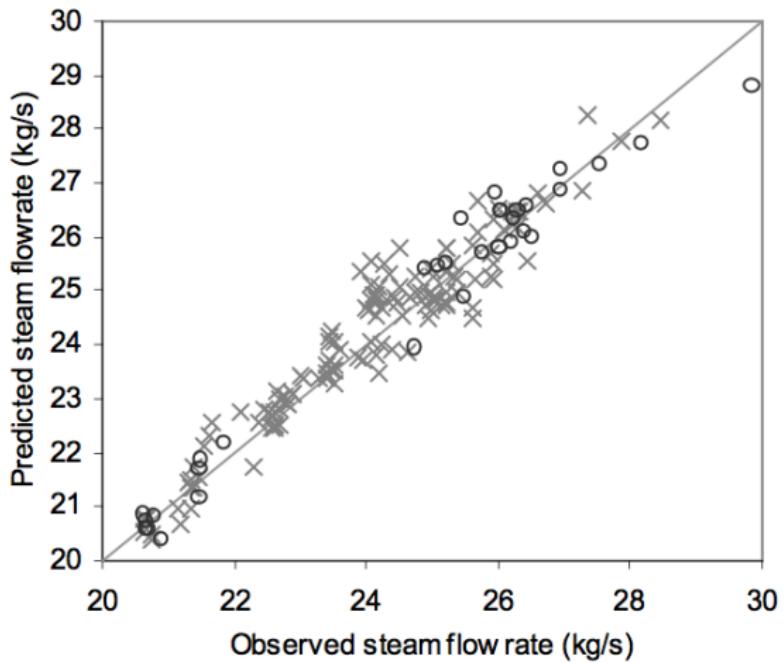
- ▶ bigger flame, brighter, and higher standard deviation (more variable)
- ▶ fewer non-flame pixels
- ▶ higher t_2 values for flame and non-flame regions (shift up-down)

Natural gas:

- ▶ more reflectance off boiler walls (more non-flame pixels)
- ▶ higher t_1 values (shifts left-right)



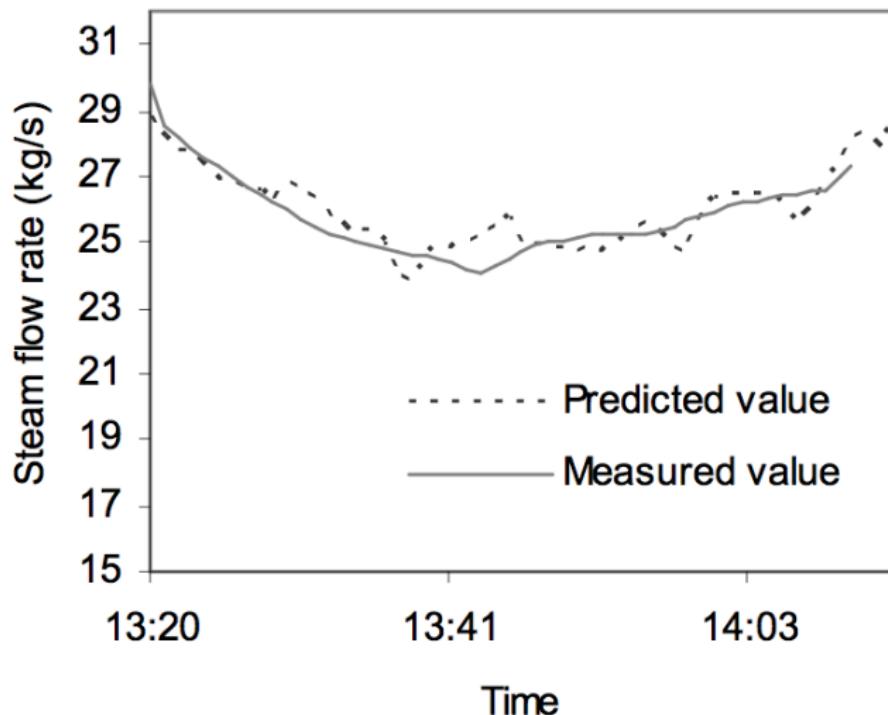
Prediction results



\circ = training data ($\text{RMSEP} = 0.4 \text{ kg/s}$)

\times = testing data ($\text{RMSEE} = 0.5 \text{ kg/s}$)

Prediction results



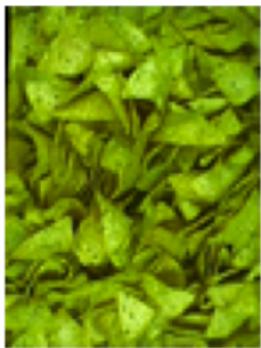
Definitely would require more in-depth study to prove long-term reliability, but shows good promise.

What was learned

- ▶ score space has no concept of *spatial* relationships
- ▶ scores only capture *spectral similarities*
- ▶ masking is effective at spectral segmentation (very rapid)
- ▶ hardest part: **good feature extraction**
 - ▶ extract features relevant to goal
 - ▶ features should be as stable as the underlying process

Seasoning application

We can see a change in product appearance with more seasoning: ²



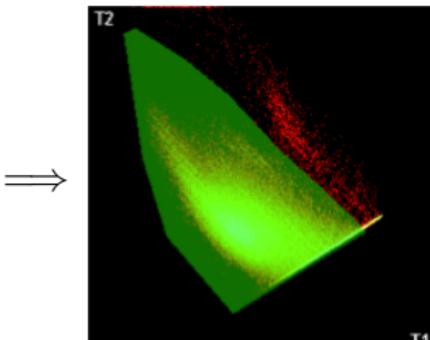
Increased seasoning from left to right

²From Honglu Yu's PhD work (used with permission)

Background: segment “background” vs “product” pixels



Training image



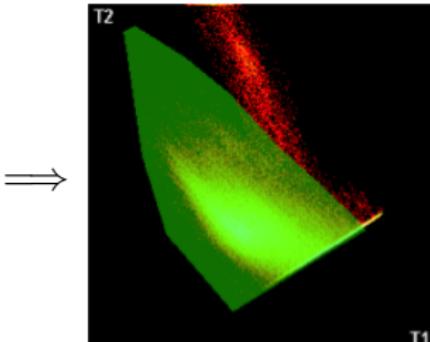
Find suitable mask



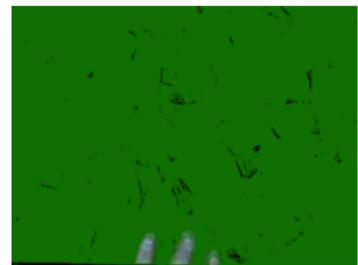
Segmented result



Future testing
image

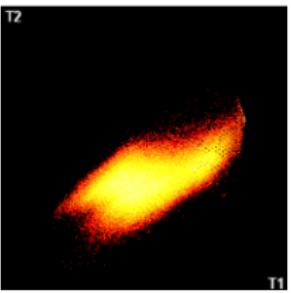
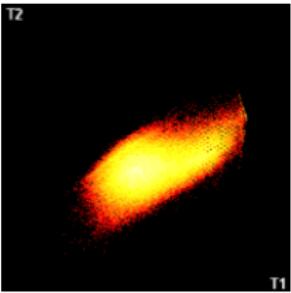
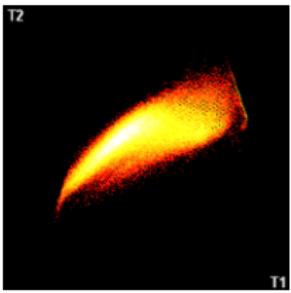
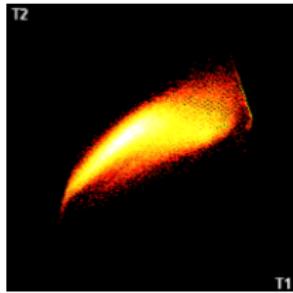


Use the same mask



Segmented result
on test image

Score images



No seasoning

No seasoning

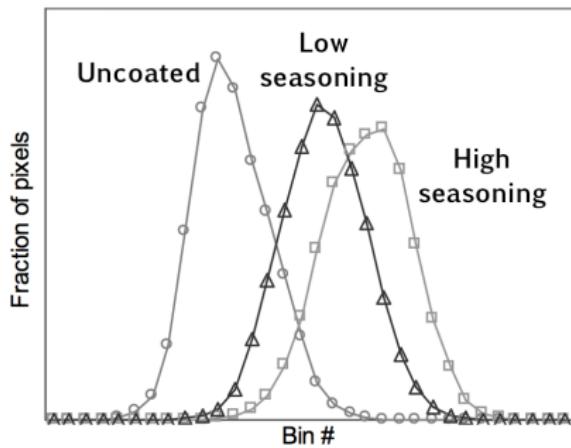
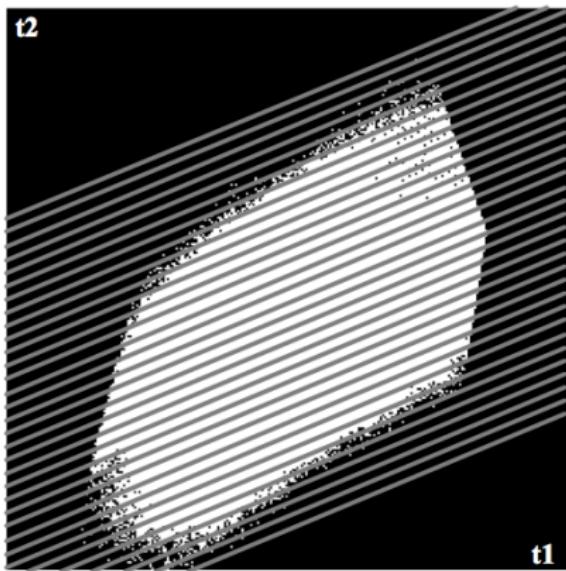
Low seasoning

High seasoning

- ▶ Image space changes, while score space remains stable
- ▶ Scores change for different seasoning (spectral features)

Feature extraction in the score space

Create bins in the score space and count pixels



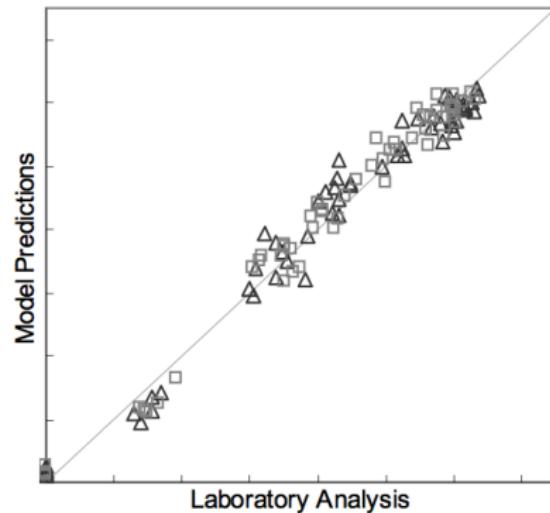
Direction of bins: direction in which seasoning level shifts the score plot ³

³Other binning methods described in [journal publications](#) and thesis

Model building and predictions

PLS model:

- ▶ **X**-space: cumulative histogram of *fractional* bin counts
 - ▶ Note the **X**-space will be very collinear
- ▶ *y*-variable: seasoning level

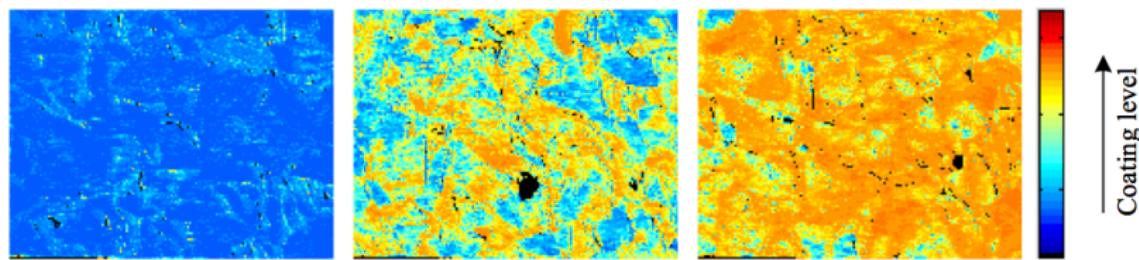


triangles = training data; squares = testing data

Apply the procedure to new images

Notice that none of the steps taken are a function of the number of pixels.

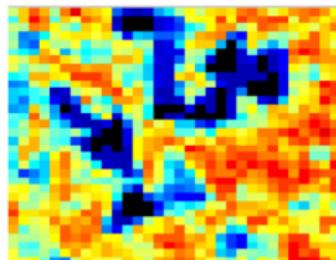
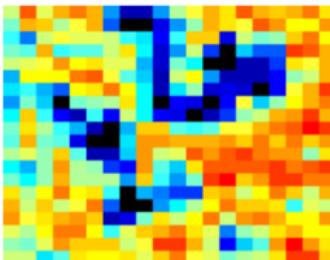
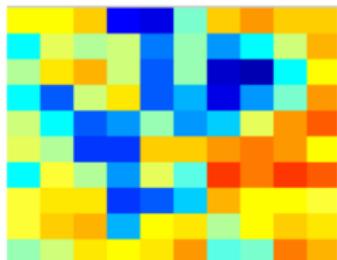
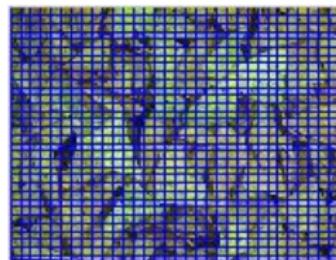
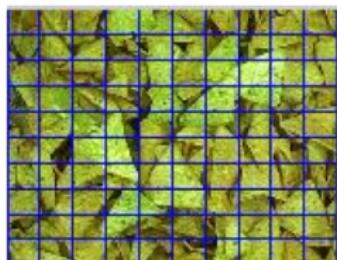
- ▶ Can apply the procedure to every pixel in an image



- ▶ Used for visual display at the manufacturing line

Apply the procedure to new images

- ▶ Apply procedure to subset of pixels: “**small-window strategy**”



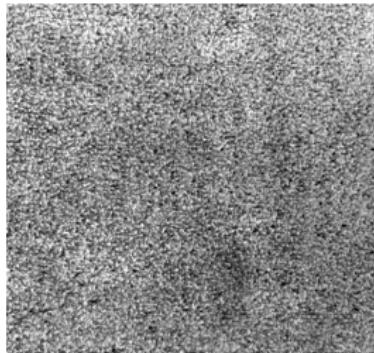
- ▶ Calculate seasoning variance: monitoring chart

Seasoning application: summary

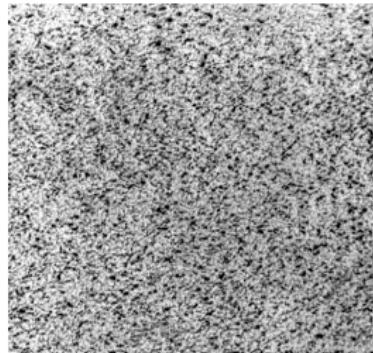
- ▶ Image segmentation again: to remove background
 - ▶ background pixels have different spectral signature
- ▶ Features extracted in the score space this time
 - ▶ score space is stable, while the image space varies
- ▶ Use PLS to regress $y = \text{seasoning}$ onto $\mathbf{X} = \text{features}$
 - ▶ the features are extremely collinear (MLR would fail)
- ▶ Prediction method is not dependent on the image size
 - ▶ use the “small-window strategy” to predict seasoning variability
- ▶ The seasoning prediction is now used for feedback control at many of the company’s manufacturing facilities
- ▶ The seasoning variance can be used in monitoring charts

Intro to appearance monitoring and control

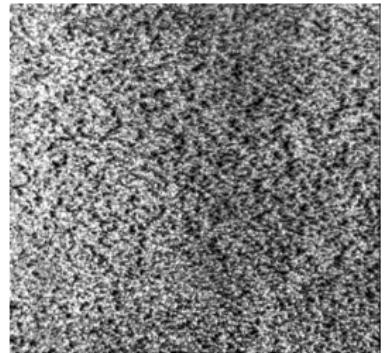
Surface quality of rolled steel sheets ⁶



Good



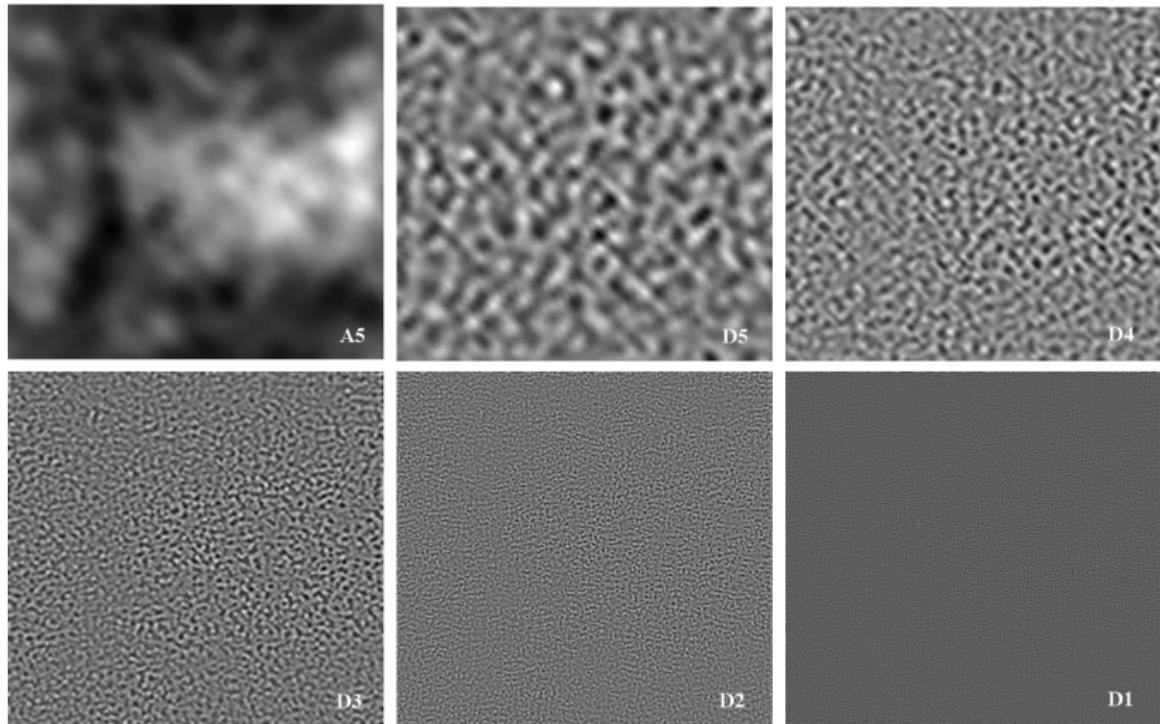
Medium



Bad

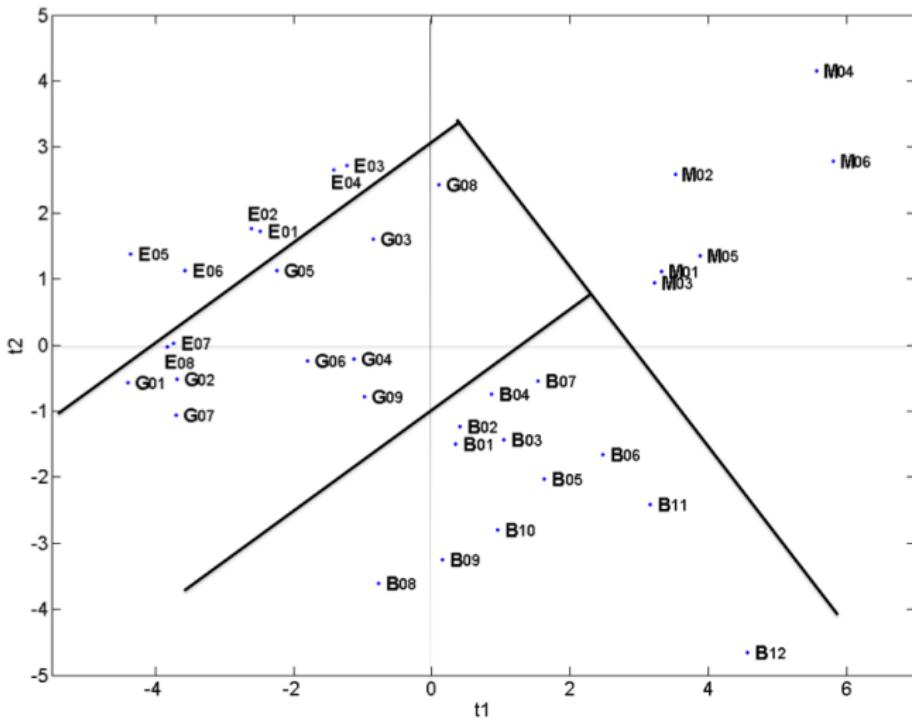
⁶From June Liu's thesis [Ch 3] (used with permission)

Wavelet texture summary



A complete summary of the original image at different frequencies.

Separating grades of steel



- ▶ PCA on the 15 features show good, natural separation into the 4 categories: **Excellent**, **Medium**, **Good** and **Bad**
- ▶ Separating lines are added manually

Practical issues

Lighting, lighting, lighting

- ▶ surface pits on steel: how to emphasize pits?
- ▶ snack food: performance degrades dramatically if ambient light changes

Practical issues

Drift in ambient lighting (day/night)

- ▶ Recalibrate with a neutral background periodically
 - ▶ background belt (e.g. seasoning application)
 - ▶ colour cards of known spectral intensity

Correct for drift. More details in [Zheng Liu's Masters thesis](#).

Similar issue: different cameras give different pixels values on same image scene

Practical Issues: Software

- MATLAB (not good for real-time use)
- OpenCV
 - Fastest option
 - Requires strong C++ skills
- EMGU CV
 - C#, Python, VB
 - Good for quick proof of concept

