Paper II

JOURNAL OF CHEMOMETRICS
J. Chemometrics 2000; **14**: 585–598

Multivariate image regression (MIR): implementation of image PLSR—first forays

Thorbjørn T. Lied[1]*, Paul Geladi[2] and Kim H. Esbensen[1]

[1]*Telemark University College, Porsgrunn, Norway*
[2]*Umeå University, Umeå, Sweden*

SUMMARY

In the effort of analysing multivariate images, image PLS has been considered interesting. In this paper, image PLS (MIR) is compared with image PCA (MIA) by studying a comparison data set. While MIA has been commercially available for some time, image PLS has not. The kernel PLS algorithm of Lindgren has been implemented in a development environment which is a combination of G (LabVIEW) and MATLAB. In this presentation the power of this environment, as well as an early example in image regression, will be demonstrated. With kernel PLS, all PLS vectors (eigenvectors and eigenvalues) can be calculated from the joint variance–covariance ($\mathbf{X}'\mathbf{Y}$ and $\mathbf{Y}'\mathbf{X}$) and association ($\mathbf{Y}'\mathbf{Y}$ and $\mathbf{X}'\mathbf{X}$) matrices. The dimensions of the kernel matrices $\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}$ and $\mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{Y}$ are $K \times K$ (K is the number of **X**-variables) and $M \times M$ (M is the number of **Y**-variables) respectively. Hence their size is dependent only on the number of **X** and **Y**-variables and not on the number of observations (pixels), which is crucial in image analysis. The choice of LabVIEW as development platform has been based on our experience of a very short implementation time combined with user-friendly interface possibilities. Integrating LabVIEW with MATLAB has speeded up the decomposition calculations, which otherwise are slow. Also, algorithms for matrix calculations are easier to formulate in MATLAB than in LabVIEW. Applying this algorithm on a representative test image which shows many of the typical features found in technical imagery, we have shown that image PLS (MIR) decomposes the data differently than image PCA (MIA), in accordance with chemometric experience from ordinary two-way matrices. In the present example the **Y**-reference texture-related image used turned out to be able to force a rather significant 'tilting' compared with an 'ordinary MIA' of the primary structures in the original, spectral R/G image. Copyright © 2000 John Wiley & Sons, Ltd.

KEY WORDS: multivariate image regression; MIR implementations; multivariate image analysis; MIA; kernel PLS

INTRODUCTION

Since the introduction of multivariate image analysis (MIA) in 1989 [1], multivariate image regression (MIR) has not been developed to the extent one would have perhaps expected. The reasons for this might be low interest within scientific society, few inspiring MIR applications and/or lack of the required computing power. With the presentation of kernel PLS, however, Lindgren [2] has shown that it is possible to reduce this last factor significantly. Computing the PLS loadings using

* Correspondence to: T. T. Lied, Telemark University College, Kjølnes Ring 56, N-3914 Porsgrunn, Norway.
E-mail: thorbjorn@lied.no

Received 15 November 1999
Accepted 3 April 2000

only small covariance matrices instead of large multivariate images reduces the number of calculations tremendously.

In traditional two-way multivariate image analysis each pixel is looked upon as an object. In image analysis the number of pixels (N) is often large, and as technology develops, constantly increasing. Thus having e.g. two million objects is not unusual today. The number of variables (K), e.g. image channels, is usually very much lower, representing e.g. wavelength (colour), polarizing angle or frequency. When these types of multivariate images are unfolded [3], we tend to get very long and narrow matrices. In MIA the loadings are usually calculated using SVD (singular value decomposition) on the covariance matrix $\mathbf{X}'\mathbf{X}$ [4], which is a $K \times K$ matrix. In kernel PLS the loadings are calculated from the $\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}$ matrix, which is also a $K \times K$ matrix. Using only small matrices in the updating of this kernel means that one does not have to carry around the large $\mathbf{X}$ and $\mathbf{Y}$ and long latent variable vectors in the numerical calculations.

MIA is first of all intended for explorative image analysis purposes. Transforming multivariate images to their most important structures (latent variables) enables a dynamic segmentation approach with problem-dependent interpretation of similar objects in the entire image [1,4]. However, in situations where external knowledge ($\mathbf{Y}$-image) is available, image PLSR can now also be considered, based on its power in guiding the decomposition of the multivariate $\mathbf{X}$-image. For predictive purposes the use of some kind of regression model is required. Some very meaningful candidates are PCR [5] and PLSR [5–8]. In this paper an implementation of multivariate image PLSR, some considerations of the method and an early application example are presented. Other application examples are available [9]. Comparison of detailed results from PCR and PLSR will be presented in a future paper.

METHOD

Traditional algorithms [10] for calculating PLS scores and loading weights for a given PC carry around the large $\mathbf{X}$ and $\mathbf{Y}$ residual matrices and corresponding parameter vectors. Because multivariate images consist of very large matrices, typically two million pixels by K variables plus one or more $\mathbf{Y}$-variable(s), these algorithms consume enormous amounts of computer memory and processing time. Thus a different approach is desired for multivariate image data.

In 1994, Lindgren [2] introduced a method designed to reduce the matrix sizes during calculation. This method initially calculates three small kernel matrices, $\mathbf{X}'\mathbf{X}$, $\mathbf{X}'\mathbf{Y}$ and $\mathbf{Y}'\mathbf{Y}$, and the master kernel $\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}$. Loadings and weights are calculated using the master kernel, which in turn is updated for each component calculated, using $\mathbf{X}'\mathbf{X}$ and $\mathbf{X}'\mathbf{Y}$. Compared with the traditional approach, which needs to update the large $\mathbf{X}$ and $\mathbf{Y}$ residual matrices, the kernel algorithm can save tremendous amounts of memory, as illustrated in Figure 1.

This approach is based on the fact that scores and loadings can be calculated as eigenvectors using square kernel matrices:

$$\mathbf{w}\lambda_1 = (\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X})\mathbf{w}, \quad \mathbf{w} : \text{PLS } \mathbf{X}\text{-weights}$$

$$\mathbf{q}\lambda_2 = (\mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{Y})\mathbf{q}, \quad \mathbf{q} : \text{PLS } \mathbf{Y}\text{-weights}$$

$$\mathbf{t}\lambda_3 = (\mathbf{X}\mathbf{X}'\mathbf{Y}\mathbf{Y}')\mathbf{t}, \quad \mathbf{t} : \text{PLS } \mathbf{X}\text{-scores}$$

$$\mathbf{u}\lambda_4 = (\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{X}')\mathbf{u}, \quad \mathbf{u} : \text{PLS } \mathbf{Y}\text{-scores}$$

Because MIA and MIR operate on vectorized images where $N \gg K$, $\mathbf{w}$ is a preferred starting point in the calibration procedure. In situations where $K \gg N$, this is not the case, because $\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}$ becomes

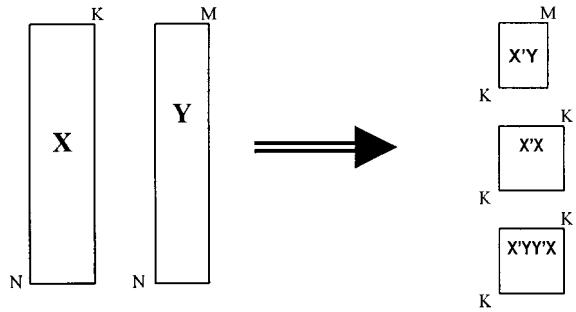

Figure 1. The kernel approach saves lots of computer memory required for calculating the weights and loadings. The actual amount saved is dependent on the *N/K* ratio.

very large. Instead, $\mathbf{XX}'\mathbf{YY}'$ is used [11] for this purpose. In situations where $N \approx K$, kernel PLS does not give much improvement. In Reference [12], kernel PLS is compared with a similar algorithm for the singular value decomposition of $\mathbf{X}'\mathbf{Y}$.

IMPLEMENTATION

It was found convenient to use LabVIEW as a programming environment for MIA/MIR. LabVIEW is mainly used for user interactions and file management, while MATLAB takes care of the actual number crunching. Our choice was made based on prior knowledge of LabVIEW and MATLAB as cost-efficient with regard to development time. The price we have to pay is a slightly slower algorithm than would be possible to obtain using C/C++. Especially the link between LabVIEW and MATLAB is slow when passing large matrices. The speed obtained is quite adequate for R&D as well as routine MIR, however.

LabVIEW (National Instruments website: www.ni.com/labview) is a graphical programming environment, written in C, which in the last few years has gained popularity and usability in numerous fields of applications. As the environment itself is becoming more stable and debugged, different toolboxes pop up around the world, introducing more and more pre-programmed functions, or VIs (virtual instruments) as they are called in LabVIEW. Because LabVIEW uses a graphical

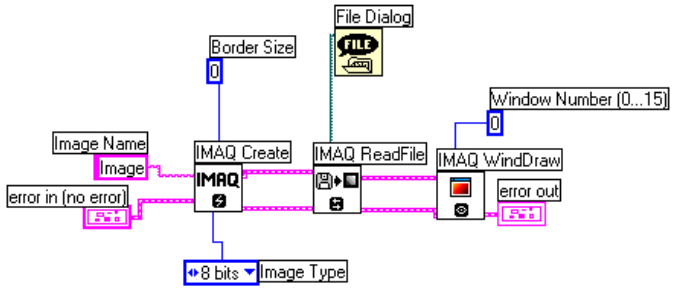

Figure 2. LabVIEW uses graphical symbols for different functions and sub-VIs, and the programmer connects these together using wires. User controls and indicators also show up as symbols in the diagram.

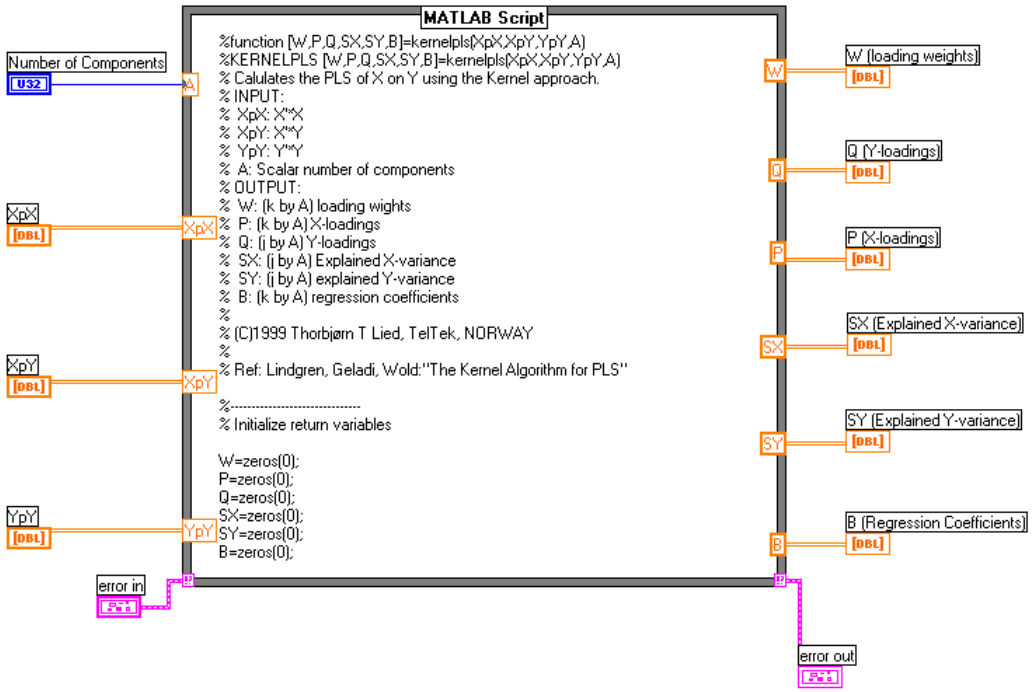

Figure 3. The actual diagram for the kernel PLS implementation. Note that the entire script is not shown as it is inside a scrollable box. The algorithm is found in Reference [2].

programming language, called G, and user interfaces are drawn directly in panels, LabVIEW truly is a visual programming system (Figure 2).

MATLAB (MathWorks Inc. website: www.mathworks.com) was used for the core numerical calculations.

There are two possibilities when combining LabVIEW and MATLAB. One is to put MATLAB scripts directly in the LabVIEW diagram, the other is to call external scripts (m-files) from the diagram. The first alternative was chosen here.

To reduce the amount of data passed between LabVIEW and MATLAB, it was decided to calculate the initial kernels in LabVIEW and pass these to MATLAB, which in turn returns loadings, loading weights and regression components.

As in most modern programming environments, in LabVIEW it is desirable to build each program as a collection of reusable sub-programs, or sub-VIs. This makes the code, or diagrams, easier to read and debug, and is of course indispensable for building complete dedicated software packages. Figure 3 shows how LabVIEW passes the initial kernels to MATLAB and calls upon MATLAB to perform the PLS calculation.

One level higher in the program, this VI is called upon with the kernel matrices as parameters. The implementation of this is shown in Figure 4.

Prior to the VI shown in Figure 4, scaling and centring of **X** and **Y** can be applied, if necessary. Following this VI, **X**- and **Y**-scores are calculated by projecting **X** and **Y** on their corresponding loadings. After this, we are ready to display score plots and score images, as well as loading plots,

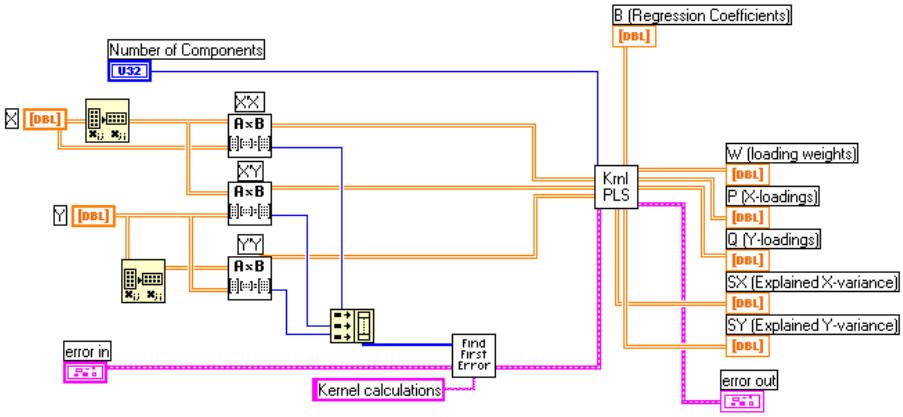

Figure 4. How the kernels are calculated and passed to the kernel PLS VI. The current VI is typically called upon after scaling and/or centring of **X** and **Y**.

and/or to carry out further calculations, etc., very much in the same tradition as with MIA [1,4–8].

While score images are shown in their original size and geometry, score plots are not. A score plot is a 2D histogram, or a scatter plot between two score vectors. Treating each pixel in the scene space as an object, and thus plotting each object individually, this scatter plot becomes unreadable without using an intensity colour-slicing map [1]. Score plots are used for object classification (**X**-scores, **T**), while **T** vs **U** (**Y**-scores) plots are used for evaluating the prediction performance of the image regression model [5–8]. The difference between scatter plots and images will be more evident when looking at an example.

APPLICATIONS

Because most effort has been put into the present new software development, a large range of applications is not yet available. Nevertheless, an early example with three spectral channels in the raw image is presented here. Although this is only a very modest multivariate image, it serves the purpose of showing the principles of MIR completely.

The raw image was captured with SILVACAM (VTT Automation website: http://www2.vtt.fi:82/aut/rs/prod/silvacam.html), which is a modified RGB video camera where the blue channel has been replaced with an NIR (near-infrared) channel. The composite raw image (R/G/NIR) is shown in Plate 1. In the present example, however, the NIR channel did not contribute much to the decomposition and was therefore removed from the data matrix for the texture derivations to be presented below (in order to give more room for the latter).

This image has been specifically designed to highlight both spectral as well as different textural and structural differences between the different objects in the image. Thus we have constructed an image with only three principal objects present:

- highly textured cloth as background (Canadian lumber-jacket);
- flat plastic fragment (‘prison window bars’);
- eight lead pencils in four colours.

The idea behind this image is that there are important differences between the spectral objects (which

can be discriminated by a standard MIA spectral decomposition [1,4]) and the textural objects which will be the main focus of concern in this application example (texturally there is e.g. only one type of pencil, while there are four spectral classes corresponding to the four colours).

Observe how the green image apparently conveys more detail and focus than the red image, especially regarding the definition of the highly textured background (Figure 5). The image is also representative of various forms of specular reflectance. This latter is directly dependent upon illumination angles, etc. For this constructed image a partly asymmetrical illumination was used, producing a clear light/shadow contrast primarily in the N–S direction.

Thus, while very simple in the number of object types present, this image in fact catches many of the principal image analysis elements and features of technological imagery, a number of different spectral classes, many or all with individual texture, illumination (light/dark/shadow) differences, etc.

The goal of this example is twofold:

(1) to discriminate between these different types of classes specifically with help from the textural information;
(2) to compare MIA vs MIR.

In order to do this, a new MIX (multivariate image texture analysis) concept is introduced whereby a series of textural image derivatives is directly added onto the series of spectral variables (from the perspective of both MIA and MIR, this simply results in a set of added **X**-channels). This will be done in three different ways in the present case.

Thus, for each of the two spectral channels (red and green), three relevant textural derivatives have been calculated, giving a total of $K = 2 \times (1 + 3) = 8$ channels (see Figure 5). The following texture filters were applied:

- median filter;
- Laplace filter;
- compound filter (sculpture + variance + median + inversion).

A reference **Y**-image is of course required for image PLSR. A 'texture index' **Y**-image (**TI**) is devised (Figure 6) which expresses the basic texture differences between the three texture classes in a quantitative manner. Texturally the piece of plastic is almost completely 'flat' (**TI** = 0–10); the pencils are slightly more complex texturally speaking (octagonal cross-section), resulting in **TI** = 20–40; while the highly textured Canadian lumber-jacket cloth displays a very high texture index, **TI** = 225–255. Figure 6 shows these texture relationships very clearly. This is the type of information that will be used in order to introduce textural relationships in the image decompositions, but exclusively as **Y**-information.

The **TI** image was constructed in Image Pro Plus from Media Cybernetics, applying a combination of texture-sensitive filters to the red channel in **X**. The combination consisted of 'sculpt', 'Sobel', '5 × median 5 × 5' and contrast enhancement, which, when applied in the mentioned order, gave the result shown in Figure 6.

Application MIA vs MIR—objectives

In order to see how image PLSR (MIR) decomposes differently than image PCA (MIA), three cases will be studied, in which the PCA and PLSR algorithms will be applied essentially to the same data set but in three different ways:

- case 1—MIA_0 (without **Y**-reference in **X**);
- case 2—MIA_Y (**Y**-reference included in **X**);
- case 3—MIR (**Y**-reference used in **Y**-block).

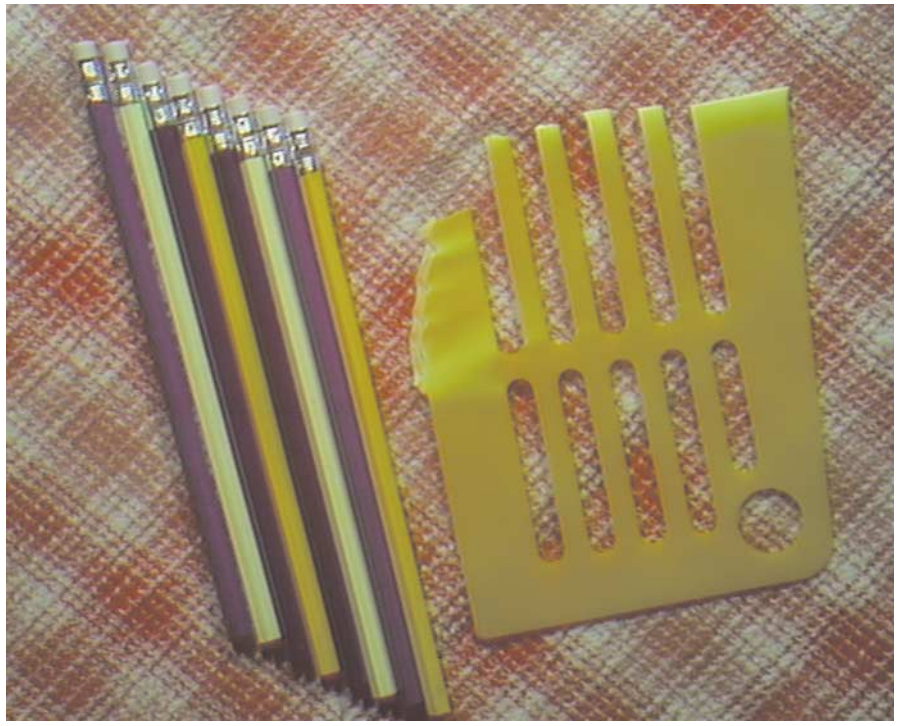

Plate 1. Raw image in three spectral channels, Red, Green, and NIR.

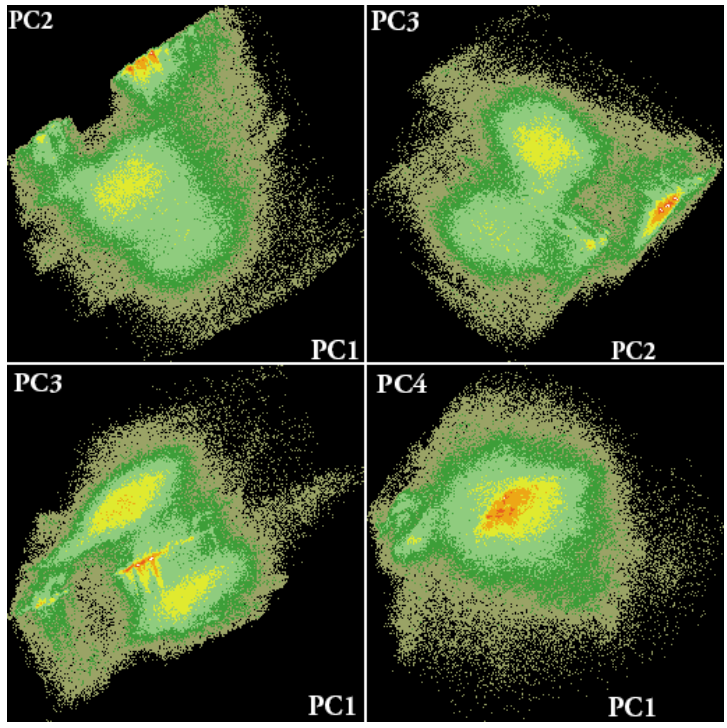

Plate 2. MIA_0 Scoreplots: 1-2, 2-3, 1-3 and 1-4.

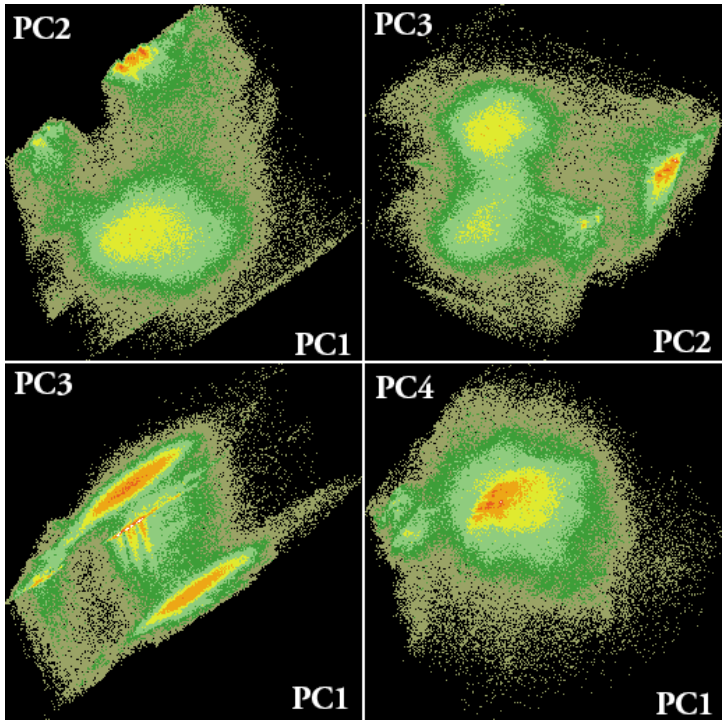

Plate 3. MIA_y Scoreplots: 1-2, 2-3, 1-3 and 1-4.

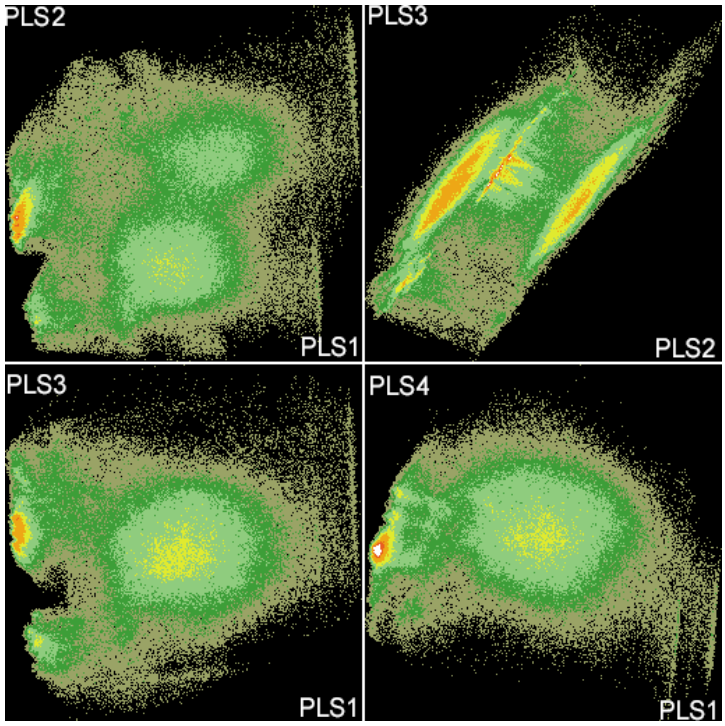

Plate 4. MIR Scoreplots: 1-2, 2-3, 1-3 and 1-4.

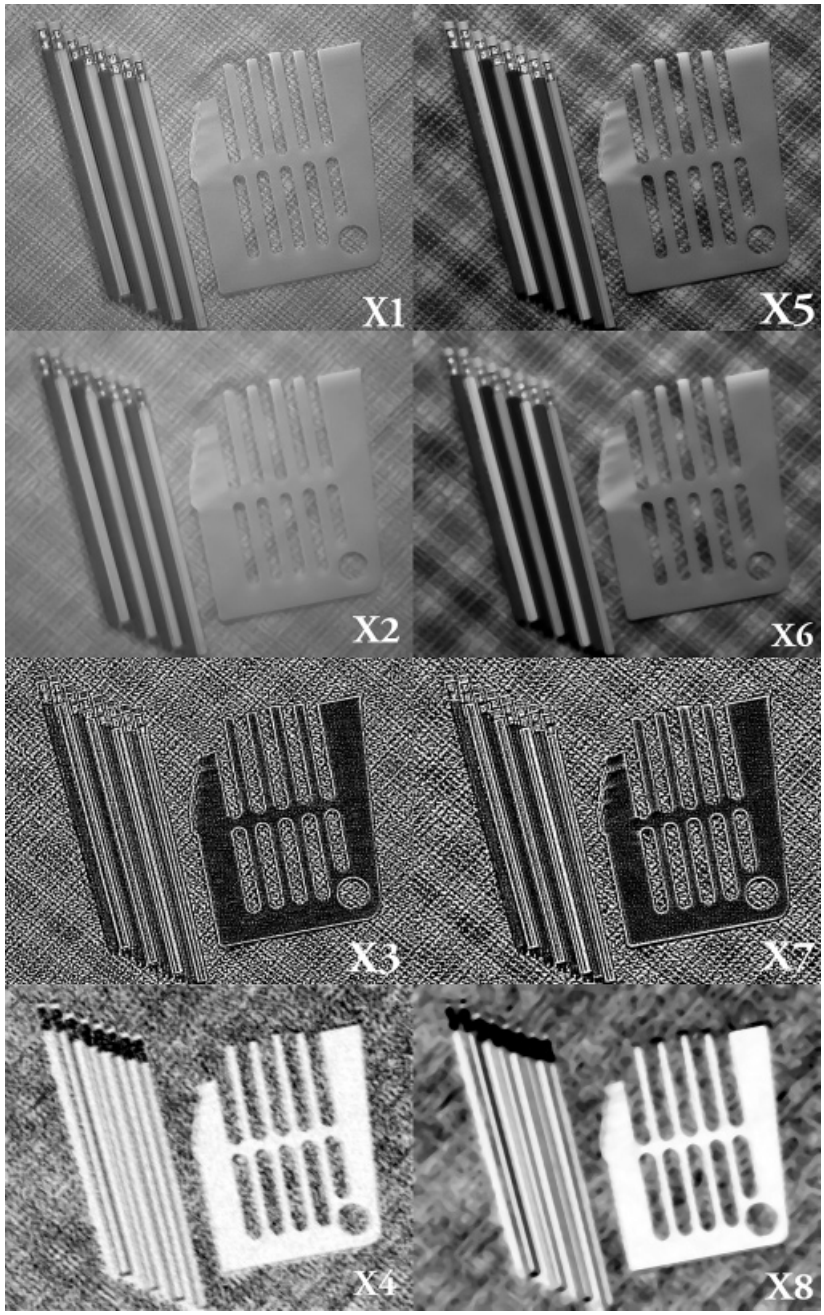

Figure 5. Two spectral channels (red, top left; green, top right) and three textural derivatives of each.

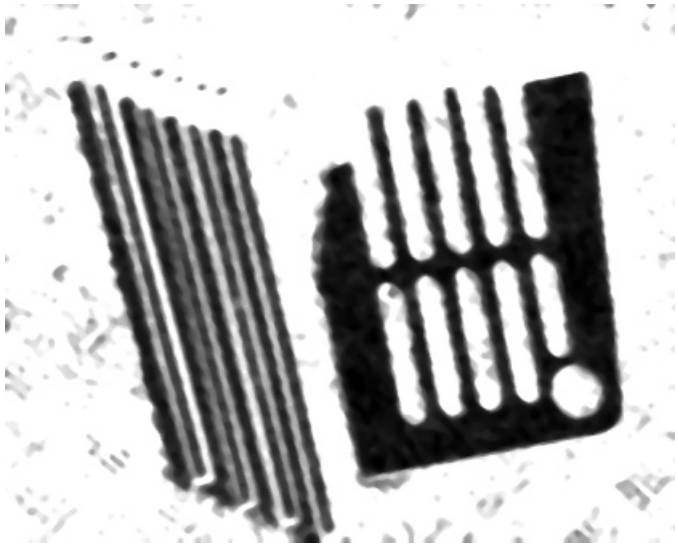

Figure 6. Reference **Y**-image expressing **TI** of the principal 'texture objects' ≠ spectral objects.

Table I shows the contents of **X** and **Y** in the three cases.

When comparing the three cases, the preprocessing must of course be identical. Thus, prior to the calculation in this example, all pertinent images were autoscaled.

Cases 1 and 2—MIA_0 and MIA_Y. In the first case, **X** contains eight variables and approximately 350 000 objects (pixels). Thus $\mathbf{X}'\mathbf{X}$ is an 8×8 matrix. In the second case, **TI** will be added to **X** from MIA_0 as an extra variable. Thus **X** will have nine variables and approximately 350 000 objects and $\mathbf{X}'\mathbf{X}$ will be a 9×9 matrix. The two models prove to be very similar, so loading plots are shown only for case 2. Score plots, though, are shown for both cases.

Figure 7 shows the accumulated explained variance for case 1. The number of PCs to use in the following discussion is not obvious from this plot, but using the standard four components that the software provides seems to be a fairly good alternative.

There is a very strong pairwise correlation between variables in these two cases. This can be seen from the loading plots (Figure 8).

One can see the following variable pairs in the loading plots: 1–5, 2–6, 3–7 and 4–8. An obvious interpretation would be that the texture filter operations on both red and green are closely similar. From these observations, one could for example argue that the number of variables could be reduced to four in the **X**-matrix, e.g. variables 1–4. If the computer is low on memory or speed, this can be considered to speed up the calculations. In the following, however, all the initial variables are used, since we have a quite different purpose than variable selection with the present decompositions.

Case 3—MIR. In the last case a regression model between the **X** used in case 1 (MIA_0) and **Y** from **TI** will be built using the kernel PLS algorithm. In this case the model will be actively forced in the direction of textural information, presumably somewhat suppressing pure spectral correlations, characterising the MIA_0 and MIA_Y cases respectively.

Table I. Contents of **X** and **Y** in MIA and MIR calculations

Channel	MIA_0	MIA_Y	MIR
Red	X1	X1	X1
Red median	X2	X2	X2
Red Laplace	X3	X3	X3
Red compound	X4	X4	X4
Green	X5	X5	X5
Green median	X6	X6	X6
Green Laplace	X7	X7	X7
Green compound	X8	X8	X8
Reference **TI** image (Figure 6)	—	X9	Y

In MIR there is still a correlation between the same variable pairs (see Figure 9), but not at all as strong as in MIA. The score image (see Figure 12) shows better textural details than in the previous cases. Thus putting the **TI** image in **Y** successfully forces the algorithm primarily to enhance texture in the decomposition, as it 'should' considering the exclusive texture index nature of the **Y**-image.

In general, of course, it is to be expected that MIA and MIR will decompose the same data set (the same multivariate image) differently, provided that the pertinent **Y**-reference information indeed does add new information. It is interesting to see how these expected differences manifest themselves in the loading and score plots (Plates 2–4) of the present example. Combining the first three score images into 'false colour composites' is always a useful way to compare alternative decompositions

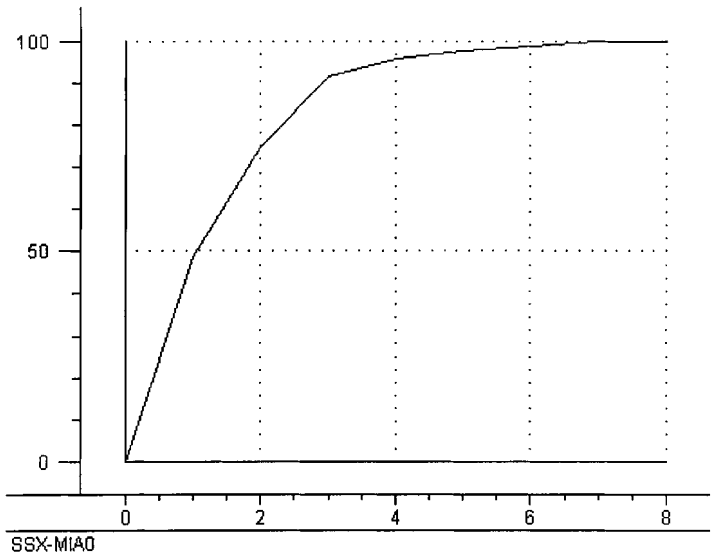

Figure 7. Explained variance for case 1, MIA without **Y**.

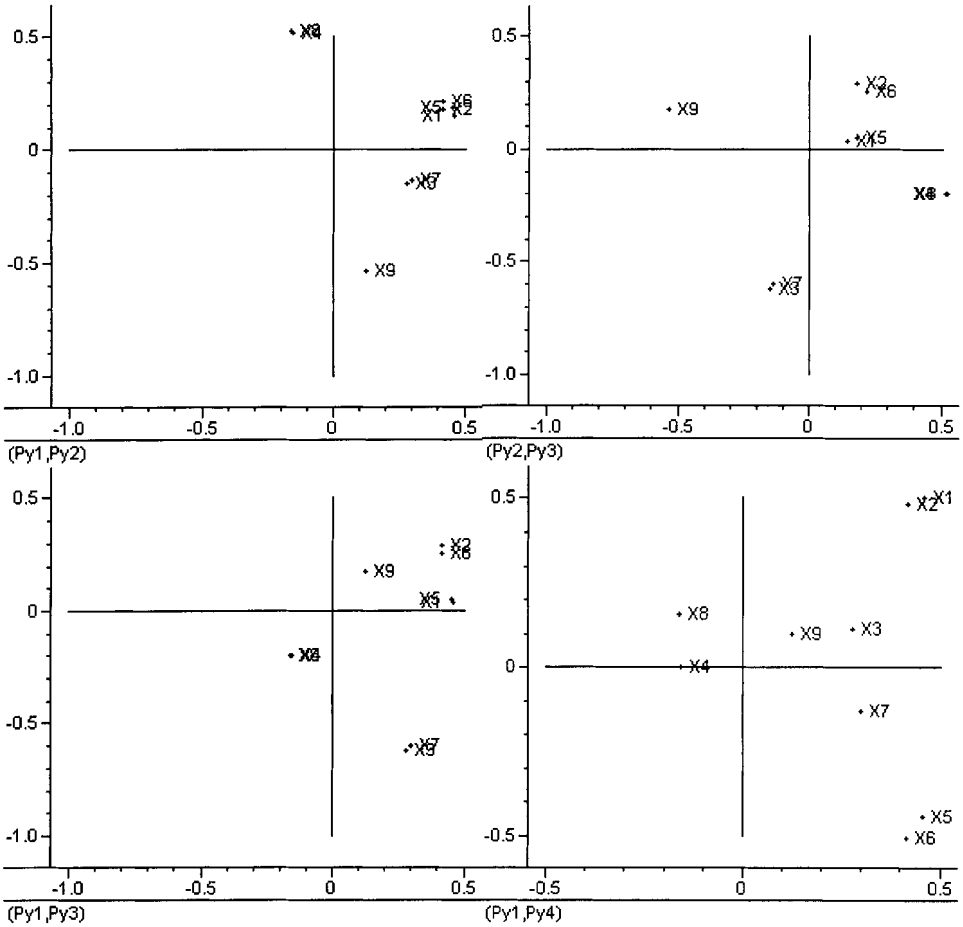

Figure 8. MIA_Y loadings 1–4.

(Figure 10–12). The most evident difference between the score images in these three cases, looking beyond differences in colour, is the gradual increase in detail. The MIR score image looks much 'sharper', more focused, than the MIA score images, primarily because of better texture and detail description.

Figure 13 shows the calibrated, explained variance for MIA_0 vs MIR [14,15]. It shows that, in this case, MIA performs better in the first two components than MIA. The third component is not very different in the two cases, while the fourth component is a little better in MIA than MIR. Figure 14 delineates y-variance modelled.

CONCLUSION

In this paper it has been shown that image PLS adds a new dimension to the complex field of analysing multispectral images. PLS was performed using the kernel algorithm, which is now implemented in our prototype MIA/MIR software system. The programming was done in a

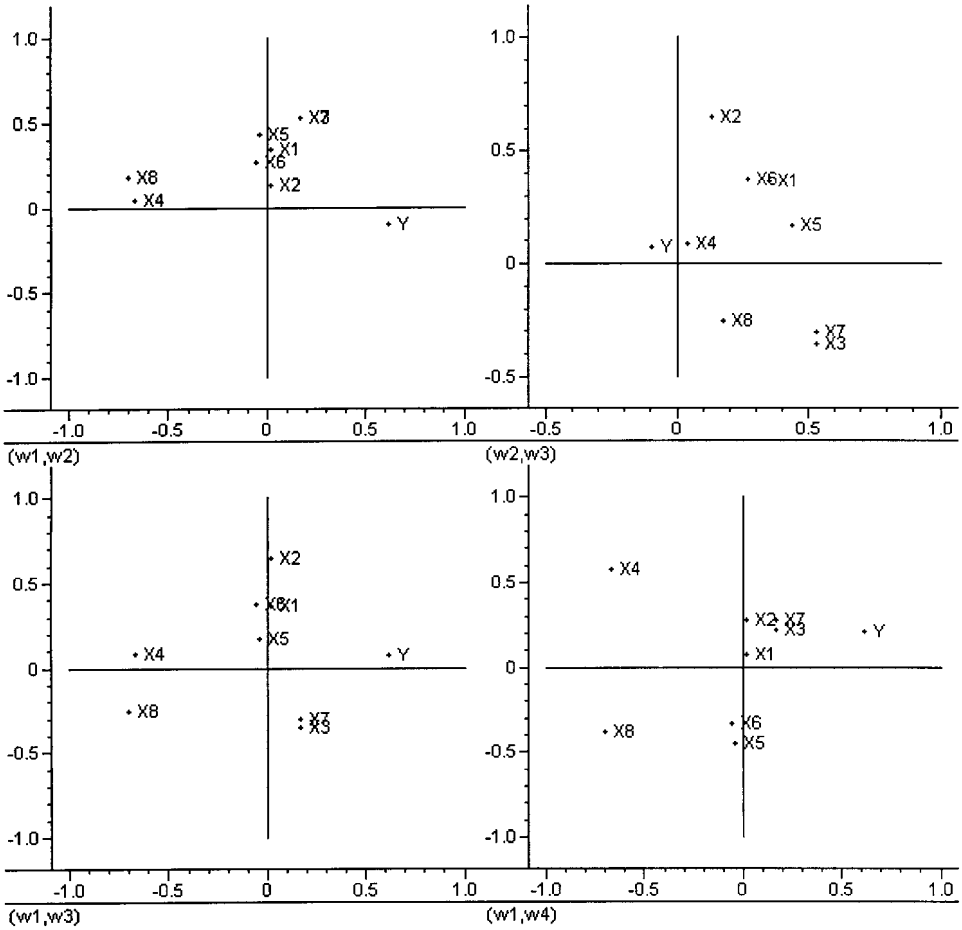

Figure 9. MIR loading weights 1–4.

combination of LabVIEW and MATLAB using the best properties of both programming environments. Using this approach, the calculations can be carried out on a standard desktop computer.

Applying this algorithm on a representative test image which shows many of the typical features found in technical imagery, we have shown that image PLS (MIR) decomposes the data differently than image PCA (MIA), in accordance with chemometric experience from ordinary two-way matrices. In the present example the **Y**-reference texture-related image used turned out to be able to force a rather significant 'tilting' compared with an 'ordinary MIA' of the primary structures in the original, spectral R/G image.

MIR requires a different validation approach than the conventional PLS approach. Much work remains, but the working prototype is now successfully implemented. We are currently also working on an extended series of representative applications.

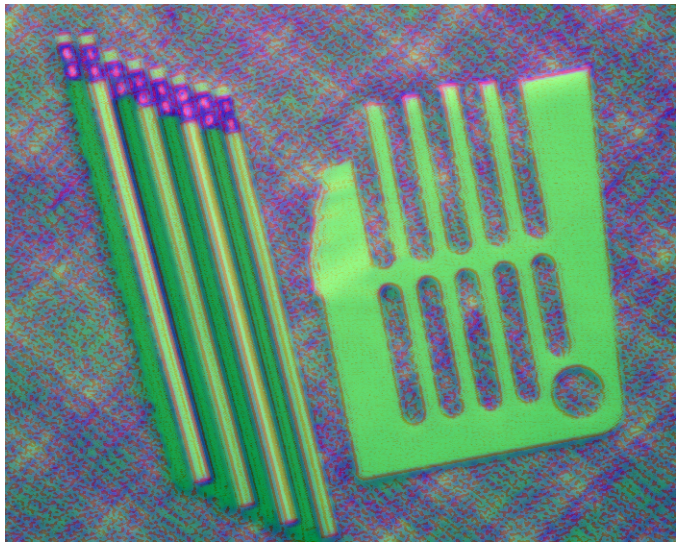

Figure 10. MIA_0 score images 1–2–3 (R–G–B).

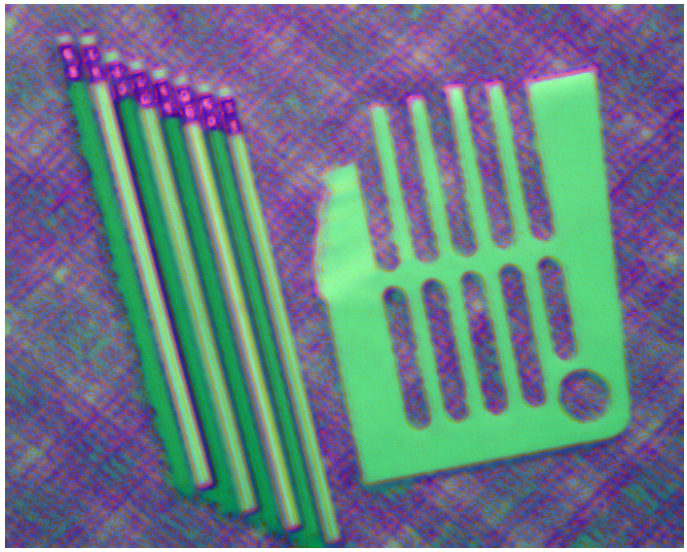

Figure 11. MIA_Y score images 1–2–3 (R–G–B).

Figure 12. MIR score images 1–2–3 (R–G–B).

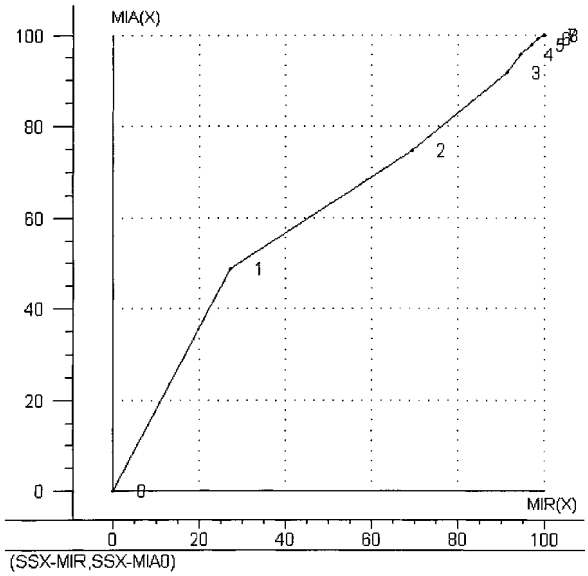

Figure 13. SSX MIA_0 vs SSX MIR.

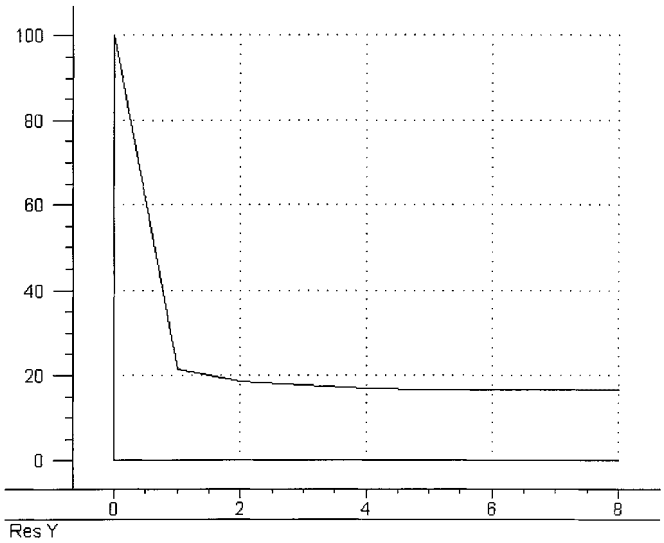

Figure 14. **Y**-residuals (SSY) from the MIR case.

REFERENCES

1. Esbensen K, Geladi P. Strategy of multivariate image analysis (MIA). *Chemometrics Intell. Lab. Syst.* 1989; **7**:67–86.
2. Lindgren F. Third generation PLS. *PhD Thesis*, Umeå University, 1994.
3. Alsberg BK, Remseth BG. Multivariate analysis of 2D surfaces—folding and interpretation. *J. Chemometrics* 1992; **6**:135–150.
4. Geladi P, Grahn H. *Multivariate Image Analysis*. Wiley: Chichester, 1996.
5. Esbensen K, Geladi P, Grahn H. Strategies for multivariate image regression (MIR). *Chemometrics Intell. Lab. Syst.* 1992; **14**:357–374.
6. Geladi P, Esbensen K. Regression on multivariate images. Principal component regression for modelling, prediction and visual diagnostic tools. *J. Chemometrics* 1991; **5**:97–111.
7. Grahn H, Szeverenyi N, Roggenbuck M, Geladi P. Tissue discrimination in magnetic resonance imaging. A predictive multivariate approach. *Chemometrics Intell. Lab. Syst.* 1989; **7**:87–93.
8. Grahn H, Sääf J. Multivariate image regression and analysis. Useful techniques for the evaluation of clinical magnetic resonance images. *Chemometrics Intell. Lab. Syst.* 1992; **14**:391–396.
9. Lindgren F, Geladi P, Wold S. Kernel-based regression: cross-validation and applications to spectral data. *J. Chemometrics* 1994; **7**:45–59.
10. Martens H, Næs T. *Multivariate Calibration*. Wiley: Chichester, 1989.
11. Rännar S, Lindgren F, Geladi P, Wold S. A PLS kernel algorithm for data sets with many variables and fewer objects. 1. Theory and algorithm. *J. Chemometrics* 1994; **8**:111–125.
12. Dejong S, Terbraak CJF. Comments on the PLS kernel algorithm. *J. Chemometrics* 1994; **8**:169–174.
13. Höskuldsson A. PLS regression methods. *J. Chemometrics* 1988; **2**:211–228.
14. Geladi P. The regression model comparison plot (REMOCOP). In *Frontiers in Analytical Spectroscopy*, Andrews D, Davies A. (eds). The Royal Society of Chemistry: Cambridge, 1995; 225–236.
15. Geladi P, Swerts J, Lindgren F. Multiwavelength microscopic image analysis of a piece of painted chinaware. *Chemometrics Intell. Lab. Syst.* 1994; **24**:145–167.

