GERARD SZATVANYI

IMPROVING QUALITY AND COMBUSTION CONTROL IN PYROMETALLURGICAL PROCESSES USING MULTIVARIATE IMAGE ANALYSIS OF FLAMES

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en génie chimique
pour l'obtention du grade de Maître ès Sciences (M. Sc.)

DÉPARTEMENT DE GÉNIE CHIMIQUE
FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2006

Résumé

La combustion est utilisé dans l'industrie chimique et du traitement des minéraux dans le but de produire de la vapeur dans les chaudières, de sécher les concentrés dans les fours rotatifs, et d'appliquer des traitements thermiques dans les fours pyrométallurgiques. Le contrôle serré de la combustion dans ces fours est très important parce que les conditions de combustion affectent directement la qualité du produit fini. Arriver à un contrôle serré de la combustion n'est pas facile à cause du fait que les flammes qu'on retrouve dans ces industries sont obtenues avec des combustibles non pré-mélangés et aussi parce que la combustion est affectée par des perturbations non-mesurées comme l'utilisation fréquente de plusieurs combustibles, certains étant des sous-produits de l'usine, et de débit et composition variables. Une nouvelle méthode est proposée dans cette étude afin d'améliorer le contrôle de la qualité des produits de ces fours tout en réduisant la consommation de carburants. Cette méthode s'appuie sur l'extraction d'information provenant d'images de flammes. La méthode d'analyse et de régression sur les images multivariées est utilisée pour l'extraction des caractéristiques de couleur de la flamme qui sont ensuite utilisées pour prédire la température de décharge des solides d'un four rotatif (qualité). Cette étude démontre que cette méthode est capable de très bien prédire la température de décharge du solide 20, 40, et jusqu'à 80 minutes dans le futur. Ceci devrait permettre une réduction substantielle de la variabilité de la qualité du produit et de la consommation de combustible.

Abstract

Combustion is used throughout the mineral processing industry to produce steam in boilers, to dry concentrates in rotary dryers, and to apply heat treatments in pyrometallurgical furnaces. Tight combustion control is very important in the latter type of furnace since the combustion conditions directly affect final ore quality. However, achieving tight combustion control is not straightforward since most of the flames encountered in industry are turbulent non-premixed flames, they are affected by several unmeasured disturbances, various flow rates, continuous variation in the mix between fuels since they are often produced by simultaneously burning several types of fuel, some of them coming from other parts of the plant. A novel method is proposed in this study to improve process and product quality control as well as to optimize the combustion conditions based on digital flame color images. Multivariate Image Analysis and Regression is used to extract the flame color characteristics from images to predict the solids discharge temperature of an industrial rotary kiln related to product quality. It is shown that this method yield extremely good 20 minutes, 40 minutes as well as 80 minutes ahead forecasts of the discharge temperature of mineral ore. This should lead to a substantial reduction in product quality variability as well as in fuel consumption.

Foreword

Destiny or free choice? Is everything already decided or are we able to change our life? In any case my life has been changed greatly by the people I've met and I wish to thank them here for their support and patience with my journey.

First, I wish to thank Crina, my wife for her steady support and patience, for her love that provided me with the energy and the desire to live my life to the max.

I wish to thank my family for being supportive, to Momo, my grandmother for laying the grounds of my personality, my sister Adela that is an example of perseverance, Mariana and Radu that have always been an inspiration.

I owe big thanks to Carl, my professor, who with his guidance, friendship and help, allowed me to progress in the right direction both scientifically and as a person.

Also, I wish to thank Laurentiu, my best friend for his help, for the time we spent together in endless discussion about why the sky is blue.

Last but not least, I wish to thank COREM (and especially Mr. Gianni Bartolacci) and NSERC for their financial support.

Für Crina, meine Frau

Table of contents

List of tables

List of figures

Chapter 1 - Introduction

Rotary Kilns and Fuel Fired Furnaces

Rotary kilns are frequently used in the chemical and mineral processing industries since they can accommodate the production of various kinds of products over a wide range of operating conditions. For example, by adjusting some operating conditions, minerals coming from different sources and thus having slightly different characteristics can be treated in the same process. Another advantage is their ease of operation as rotary kilns are rather simple installations.

These very versatile process equipments are used for the calcinations of lime (Järvensivu et al., 2001) and coke (Martens et al., 2001), the pyrolysis of various kinds of wastes, (Rovaglio et al., 1998), and for ore roasting and sintering (Cross et al., 1999). They are also used to dry a wide variety of products, such as fish and soy meal (Alvarez et al., 1994), minerals (Duchesne et al., 1996), sawdust (Alvarez et al., 1994), grain, bark, coal, fertilizer, and other aggregates (Kamke et al., 1986). A typical rotary kiln is depicted in Figure 1. Although their dimension might change and they can become somewhat particular to their specific application, their basic principles of functioning remain the same across the industries that use them.

Figure 1: A typical rotary kiln (Taken from J.J. del Coz Diaz et al., 2001)

Despite the fact that rotary kilns are pretty common, they are, however, very complex systems involving simultaneous solid-gas heat and mass transfer coupled to chemical reactions and solids transportation problems. Indeed, rotary kilns bring solids into close contact with a hot gas within large, inclined and rotating cylinders. Modeling of rotary kilns to improve understanding and to optimize their operation is still a very active research area.

Available models range from simple empirical correlations (Friedman and Marshall, 1949; Saeman and Mitchell, 1954) to very detailed fundamental models. To represent the dynamic behavior of a rotary dryer, Duchesne et al. (1996) built a simulator consisting of three models, a furnace model, a solids transportation model and a gas model. J.J. del Coz Diaz et al. (2001) developed a structural model that uses finite elements for a cement rotary kiln. Martins et al. (2001) have modeled the calcination of petroleum coke using differential equations as well as energy conservation principles. Finnie et al. (2005) have modeled longitudinal and transverse mixing in rotary kilns using the Discrete Element Method. The focus was on the effect of the main operating conditions on solids transport, (i.e. the filling degree and the rotational speed of the rotary kiln). Finally, Marias et al. (2005) have been

using three types of models (a bed model, a kiln model and a gas model) to create a simulator for the aluminum waste pyrolysis in a rotary kiln. Most of these models predict the temperature profiles of the bed of particles, the gas phase and the kiln internal walls in the axial direction of the rotary kiln, as well as the composition profiles for the gas and solid phases. Most simulators also include some combustion model.

One very important issue with rotary kilns is that for most of them, the hot gas is produced using combustion of various types of fossil fuels. Although the use of combustion is justified, in general, by the very high gas temperatures required to carry out solid phase reactions or for drying of solids, these systems are very energy intensive and contribute to the environmental problem of gaseous pollutant emissions. Constantly increasing fuel prices and the recent adoption of the Kyoto Agreement put enormous pressure for reducing fuel consumption and pollutant emissions while maintaining high final product quality. There is therefore a need for developing new automatic control and/or optimization strategies for fuel fired rotary kilns, to help the industry achieve these goals and operate combustion processes more efficiently.

However, in current practice, the combustion process within rotary kilns is not directly controlled. The heat transfer effectiveness can only be evaluated using solids and gas temperature measurements obtained from the input and output streams of the kiln. Very few measurements are available within the kiln itself. Some rotary kilns are equipped with thermocouples inserted through the rotating shell, but these are often unreliable due to the harsh environment. This is particularly true for the pulp and paper industry (lime kilns), the cement industry (clinker production), and the pyrometallurgical industry, three of the most important users of rotary kilns, since fouling occurs on the thermocouples located in the high temperature region (most interesting region), to a point where the readings obtained from these become useless until maintenance is performed during a complete shutdown of the kiln.

Having access to internal state measurements within a rotary kiln, such as from the combustion process itself (i.e. the flame and surroundings) would, however, be very helpful to reduce fuel consumption for a constant or better product quality. Indeed, as will be shown in this work, the combustion process often introduces disturbances within the kiln,

which affect both the quality of the product and the fuel consumption. In most rotary kiln applications, turbulent non-premixed combustion is used, which means that combustion and mixing of the fuel and the oxidizer (e.g. air) occur simultaneously, at the burner tip. This type of combustion process is more chaotic and difficult to control than when the fuel and air are premixed and then burned (Yu and MacGregor, 2004). The secondary air flow rate is usually changed using fans but is not always measured. Moreover, several rotary kilns are operated using multiple sources of fuel having different heats of combustion, some produced within the plant itself and the others obtained from suppliers. All these modifiy the heat released by the combustion process and this, eventually affects final product quality. In practice, the variability of final product quality is often such that a certain amount of overheating is required to meet the specifications of the product. Reducing the variability of the combustion process is the key to reducing product quality variations and, in turn, the amount of overheating, fuel consumption and pollutant emissions.

Another advantage of using information from the combustion process is the fact that rotary kilns typically have long residence times and slow dynamics. When disturbances introduced by the combustion process propagates to the solids within the kiln (via heat transfer), it might take several minutes to detect this event and apply a corrective action, and even more time before this corrective action brings results. When combustion disturbances are such that the temperature of the gas and the solids rise to a certain high level, this may trigger automatic kiln shutdowns, which are very costly events (loss of production time). On the other hand, when the temperature decreases by too much, than product quality starts degrading. Having access to on-line information about the combustion process would allow developing predictive models to help avoid these production losses and improve overall productivity.

Objectives

The main objectives of this study are therefore to develop a methodology to quantify actual variations in the combustion process taking place within rotary kilns, and to use this information to build predictive models for final product quality. Another objective consists of investigating ways to integrate such models into quality control and combustion control strategies in order to reduce quality variations and, subsequently, fuel consumption via a reduction in the amount of overheating.

Such an internal state sensor to monitor variations in the combustion process will be developed using digital imaging of combustion flames and of the interior of rotary kilns (i.e. walls, solids, etc.). Multivariate Image Analysis and Regression techniques will be used to extract the relevant features of these images and to build regression models between final product quality and these image features as regressors. The methodology will be illustrated using images and data collected from an industrial rotary kiln used for ore roasting. This rotary kiln is operated by the QIT Fer & Titane company (Sorel-Tracy, Québec, Canada).

The rest of this thesis is divided as follows. Previous contributions on rotary kiln control and combustion imaging will be presented in Chapter 2. Following the literature review, the industrial rotary kiln used in this study as well as specific issues, operation and control problems will be discussed in Chapter 3. In Chapter 4, the Multivariate Image Analysis and Regression techniques will be described whereas the implementation of the methodology will be the subject of Chapter 5. The Multivariate Image Regression models as well as their predictive performance of final product quality will be discussed in Chapter 6. Finally, the main conclusions drawn from this study as well as future work appear in Chapter 7.

Chapter 2 - Literature Review

This study combines two subjects that are relevant to this particular work, the automatic control of rotary kilns and combustion imaging. The most relevant contributions made in these two areas will be reviewed in this section. This review shows that all previously published control strategies are based solely on measurements external to the rotary kiln (i.e. taken on input and output streams) and that, to the author's knowledge, the use of combustion related information or other such internal state measurements has never been attempted on rotary kilns. It will also show that combustion imaging was never applied for quality control or combustion control in rotary kilns.

Automatic Control of Fuel Fired Rotary Kilns

Control strategies range from a simple feedback to complex mathematical model based strategies. The simplest model has been reported by Douglas et al. (1992a) where two PI controllers (proportional and integrative actions) were used to manipulate respectively the air flow rate and the rotation speed of a rotary sugar dryer to reach the desired product moisture content and temperature.

Myklestad (1963) used a non-linear model based control strategy. He developed phenomenological relationships relating respectively the air flow rate, the air temperature and the feed-mass flow rate to the product moisture content in order to keep the later one at the desired level.

Otomo et al. (1972) used a statistical approach to Computer Control of Cement Rotary Kilns. The goal is to control the kiln rotation speed as well as the fuel that is consumed in

the process using a completely automated controller. For this they used a combination between a classical linear controller and an empirical designed kiln rotation controller.

Robinson (1989a, 1989b, 1992) proposed an inferential control strategy. He developed an empirical model to infer the moisture content of solid, using reliable process measurements. These are the fuel rate, kiln drive power, exit gas temperature, intermediate gas temperature, burning zone temperature, secondary air temperature, cooler grate speed, percent oxygen at the exhaust, draft damper position, cooler under great pressure, kiln speed and feed rate of material. Of these variables fuel rate, cooler grate speed, draft damper position, and kiln speed are considered to be the manipulated variables whereas the others are the controlled variables.

Najim (1989) used the most complex model by implementing a learning control strategy to manipulate the oil flow rate of a rotary phosphate dryer in order to maintain constant the degree of moisture in the output phosphate.

Duchesne et al. (1997) used a dynamic rotary dryer simulator for mineral concentrate to compare a few moisture control strategies: a simple PID controller, a PID plus feedforward control, and a Neural Network controller. The rotary dryer has four major inputs: the concentrate feed flow rate, the feed moisture content, the oil mass flow rate and the secondary air flow rate to the furnace. In this study the oil flow rate is used to control the product moisture content and to compensate for disturbances. This study has shown that a PID controller supplemented by a feedfoward adjustment of oil flow rate based on a feed moisture measurement provided the best control results.

Jarvensivu et al. (2001) used a Neural Network model that takes the most significant parameters (lime mud flow rate, lime mud density, lime mud temperature, cold-end temperature, hot-end temperature, burning zone temperature, cold-end pressure, kiln drive torque, excess oxygen and TRS emissions) as inputs to control the energy consumption of a lime kiln. They also achieved good results with the reduction of sulfur emissions.

For Didriksen et al. (2002), the most important control loop for the rotary dryer was the control of product moisture. The feed of sugar beet pulp was used as a manipulated variable to control the product moisture in a feedback control loop.

Although the most obvious logical connection to energy consumption is the inner temperature profile of the rotary kiln, in all the above mentioned control strategies there is no attempt made to use internal process data as input data. Only external process data is used such as pressures, fuel flows, external temperatures, etc.

The biggest problem with all the above mentioned control strategies is that they only compute control actions after the changes in the process have occurred. This common drawback to all feedback only control systems is such that process adjustment (by the way of changing the manipulated variables) comes in with a certain delay that causes final product loss.

Combustion imaging

Using flame images that have as background the entire interior rotary kiln for process control is an interesting and a logical approach as the color of the flame provides a lot of information about combustion and the background areas of images (with the interior of the kiln) provide information about the temperature profile within the kiln along its length.

Flame imaging has already been investigated in the past, but most of these contributions were performed on laboratory scale combustion systems and on premixed flames. Also flame images have never been used for forecasting product quality in an industrial rotary kiln. Past studies concentrate on laboratory scale furnaces.

Flames using spetrometers

Some researchers have focused on using spectrometers in studying the flames. That's the case of Allen et al. (1993) that use IR Spectrometers and Neural Networks to estimate flame temperature and Yamaguchi et al. (1997), who to detect air ratio, used an IR spectrometer.

Flames using grayscale images

Victor et al. (1991) used flame images from a glass furnace in order to classify and characterize them. This study uses grayscale images as well.

In another study, Huang et al. (1999), used flame images to predict flickering rate while others, Shimoda et al. (1990), Lu et al. (1999), Lu et al. (2000), and Yan et al. (2002) predicted the unburned carbon, CO_2 and NOx emissions. They all used grayscale images and they would compute geometrical and luminous properties of the flames to classify the flames.

Bertucco et al. (2000) used cellular neural networks to analyze the flame image and to classify it. Their study is laboratory scale and it only uses grey images.

Flames using color images

However, in the last years a few researchers have investigated the use of color flames, RGB flame images (Red Green Blue image, or simply, a color image) and have extracted flames features from RGB color images of flames. This is the case of Wang et al. (2002), who estimated the NOx concentration by analyzing color flame features using segmentation techniques. It is also the case of Keyvan et al. (2003), which used Neural Networks on a laboratory scale furnace to extract flame features and to optimize the burning process.

The shortcomings of the studies done before are that most of them used only grayscale images, which carry less information than color images, and all previous work employed a complicated image segmentation technique in order to delimit regions of interest (i.e. the flame) that is not the most efficient technique to extract combustion information. Using only grayscale images means that all the color information of a flame is lost, information that provides an insight into the phenomenon that take place in the rotary furnace.

Also, segmentation is very difficult due to the fact that segmentation is working with spatial coordinates and as flames are moving a lot it becomes highly problematic to spatially locate them. Since flames are turbulent, they bounce around continuously and hence, extracting the flame's visual characteristics requires finding the location of the flames boundaries for each image. The problem becomes even more difficult when the flames are obtained by burning a mixture of two fuels, especially if the fuels are not premixed in which case the flame movement kinetics is very high. The flame image segmentation increases computation time and might cause difficulties for on-line monitoring of highly turbulent flames.

This problem has recently been addressed by Yu and MacGregor (2004) who applied the Multivariate Image Analysis (MIA) technique to RGB images of non-premixed turbulent flames from an industrial boiler. As will be further discussed in Chapter 4, MIA basically classifies the pixels of an image according to their spectral characteristics, regardless of their spatial organization within the image. This way, all pixels belonging to a flame can be classified together and quantified without having to locate the flame using segmentation techniques. The application published by Yu and MacGregor (2004) is somewhat simpler to the rotary kiln problem since the boiler dynamics is negligiblem which is not the case for rotary kilns. This work will have to deal with the slow dynamics of rotary kilns as well.

Chapter 3 - The industrial rotary kiln of QIT, its particularity and problems

Quebec Iron and Titanium Inc. (QIT) is an iron ore processor that is extensively using rotary kilns. A total of four of these kilns are currently operated by QIT (Sorel-Tracy, Québec, Canada), each of them are about 70 m long and have a diameter of about 4m. The kiln is used to apply a heat treatment to raw ore by a direct contact with a hot gas flowing in countercurrent with the solids. The residence time of the solids within the kiln is in the order of 75 minutes, determined by the rotational speed and the inclination of the kiln as well as the solids throughput. A number of thermocouples are inserted through the kiln shell at different locations along the kiln length, however, as mentioned previously, these are considered unreliable by operators and engineers. They are located in an extremely harsh environment and to ensure reliable readings, they would require frequent maintenance due to fouling. Maintenance can only be performed during shutdowns since the thermocouples rotate with the kiln.

As shown in Figure 2, the combustion takes place at the solids discharge end of the kiln. Two types of fuel as well as primary air are fed to the burner tip from three concentric pipes without premixing. The flow rates of primary air and the two fuel types are measured on-line. Process fuel A is produced in another part of the plant and both its flow rate and composition vary. The flow rate of fuel B (supplied to the plant) is adjusted to maintain heat released by the combustion, based on a relationship involving the ratio of standard heat of combustion of each fuel. Finally, secondary air is also blown in the kiln to support complete combustion and to maintain a certain amount of excess oxygen in the off-gas for safety considerations. Secondary air flow rate is not currently measured, but is changed using fans.

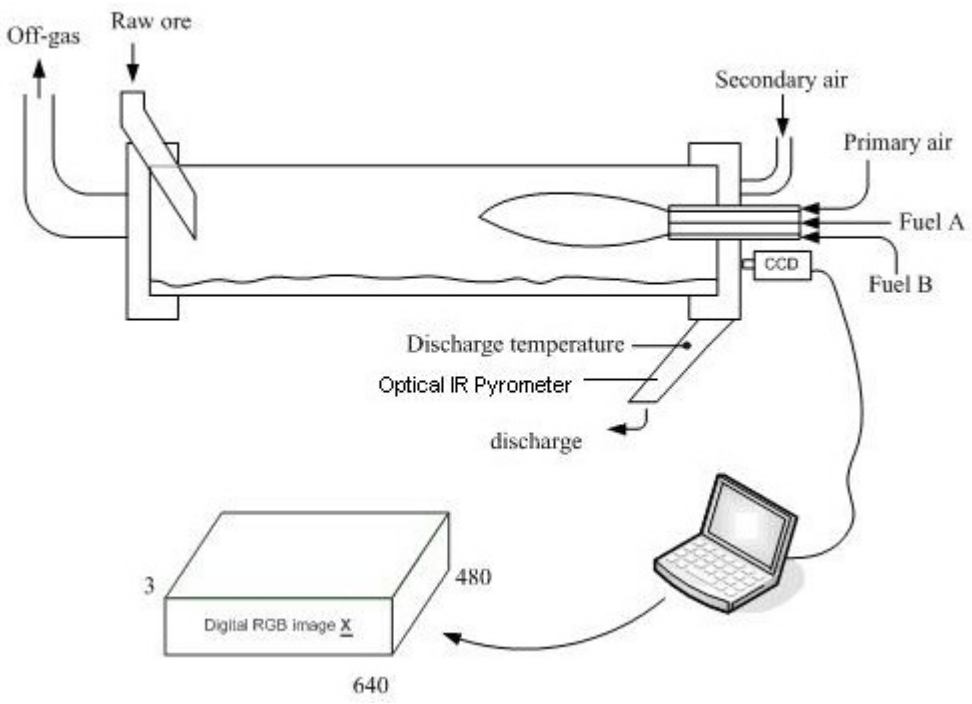


Figure 2: QIT's rotary kiln and imaging system setup

This kiln is used for ore roasting. The quality of the final solids product is determined by the extent of this roasting as well as some physical properties of the material. Quality variables are therefore measured in the laboratory. Two issues arise from laboratory measurements: 1) the actual quality analyses are performed on a composite sample obtained from all four kilns, and 2) these analyses are only available a few times per day. On-line quality control therefore cannot be performed directly using laboratory analyses. It is rather achieved in practice using the solids discharge temperature, which is measured using an Infra-Red pyrometer (see Figure 2 for the location of this sensor). The rational behind that is threefold: 1) the solids discharge temperature is highly correlated to product quality measured in the laboratory, 2) this temperature is easier to measure with good accuracy and reliability, and 3) the IR pyrometer provides more frequent measurements than laboratory quality analyses (in the order of seconds instead of a few times per day).

The quality control problem, which is depicted in Figure 3, is to maintain the solids discharge temperature above a lower limit, below which product quality starts degrading (production loss), and below an upper limit, above which the kiln automatically shuts down for safety reasons. In the latter case, several minutes are required to restart the kiln and also

involve production losses. Achieving a solids discharge temperature within these limits is, however, no longer sufficient due to constantly increasing fuel costs and the pressure to reduce CO_2 emissions (e.g. Kyoto agreement) and other combustion pollutants such as CO, NO_x and SO_2. Note that the quality control problem discussed above is specific to QIT. However, rotary kiln control problems in other companies and in other areas than roasting, such as for lime kilns, clinker kilns and even for rotary dryers, can, in general, also be formulated in a very similar way since heat transfer from a gas produced by combustion to solids is most of the time the mean by which the transformation is performed on the solids.

Quality control is currently performed manually by the plant operators. Adjustments to total fuel flow rate (manipulated variable) are made when solids throughput is changed (on a weekly basis) and whenever solids discharge temperature approaches the lower or the upper temperature limits. However, to avoid producing off-specification materials or automatic kiln shutdowns due to high temperatures, operators have very little time to react to drifts in the solids discharge temperature due to the process dead-time and slow process dynamics. Step change tests made by control engineers show that a dead-time of about 20 minutes exists between total fuel flow rate and discharge temperature whereas the time constant of that transfer function is about 40 minutes. To help operators in decision making, plant personnel have tried different methods to predict the future behaviour of solids discharge temperature over the dead-time period, such as building empirical models using process data (including thermocouples inserted through the shell) and measuring solids temperature a certain distance from the discharge point using an IR pyrometer. However, little success was obtained. As will be shown in the next sections, flame imaging is very successful in solving that problem.

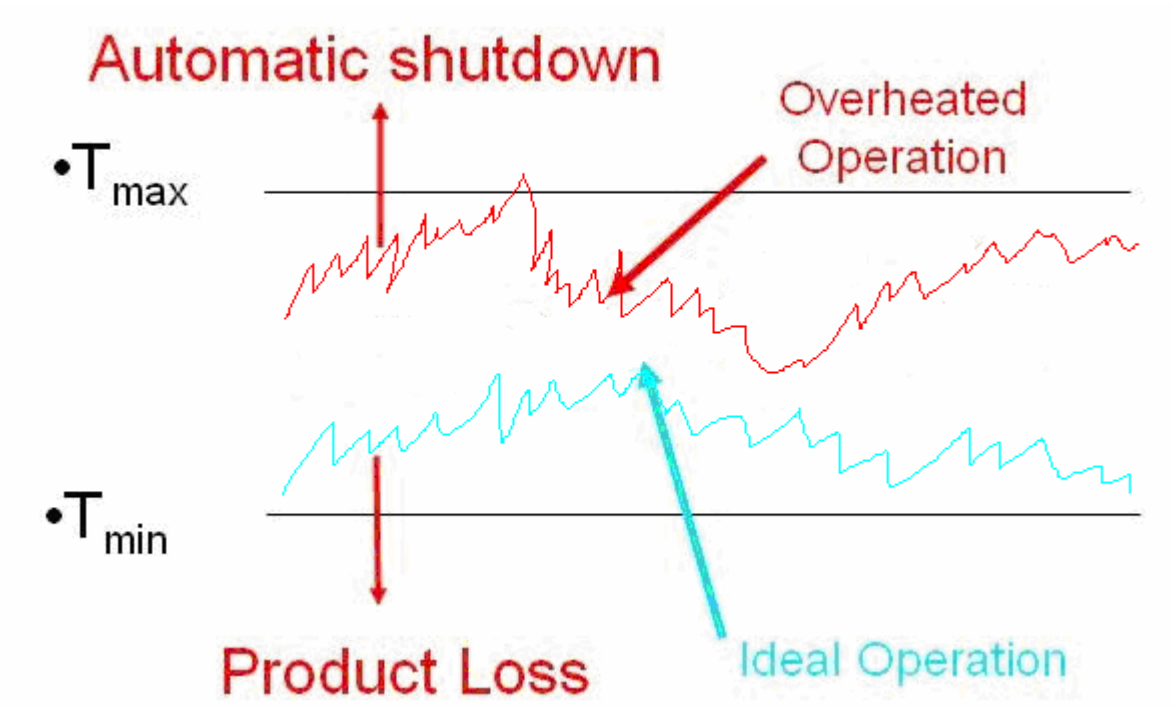


Figure 3: QIT's quality control problem.

QIT Installation Setup

The color CCD video camera (JVC TK-C1380) is installed in a small opening just behind the burner (see Figure 2). An air-cooling device is used to protect the CCD from damage caused by high temperatures. The output signal of the camera is sent to a computer located in the control room, where the frames are digitized using a frame grabber card. Each of the resulting digital image forms a three way array or a "cube" of data consisting of 640 x 480 pixels (spatial dimensions) and, for each of these pixels, the spectral information in the red (R), the green (G), and the blue (B) colors are stored in the third dimension of the cube (i.e. the spectral dimension is 3). The RGB light intensities vary between 0 and 255 with a resolution of 24 BPP (Bits Per Pixel).

To develop prediction models for solids discharge temperature (quality), a total of 80458 such images were collected over time at a rate of 1 frame every 10 seconds. Images were intentionally gathered at different periods during year 2004-2005 to capture any seasonal

variations and to test the robustness of the prediction models. Table 1 summarizes the five image collection periods.

Table 1: Summary of collected images

Periods	**Date**	**Number of frames**
1	May 2004	2526
2	June-July 2004	6730
3	August 2004	1839
4	November 2004	23134
5	February 2005	46229
Total:		**80458**

On-line kiln operation data was also collected and synchronized with the flame images. About 50 measurements currently are available around the kilns. In total around 140000 process data points were acquired from the process and have been correlated with the images. However, the most relevant measurements for this work were solids throughput and discharge temperature, as well as the following combustion related variables: fuel (A and B) and primary air flow rates, fuel ratio and total fuel flow rate, and the shutter position of the secondary air blower. A sample of data acquired in this study is shown in Figure 4. Note that the data shown in this thesis are not presented in their absolute values but rather in relative units due to their proprietary nature.

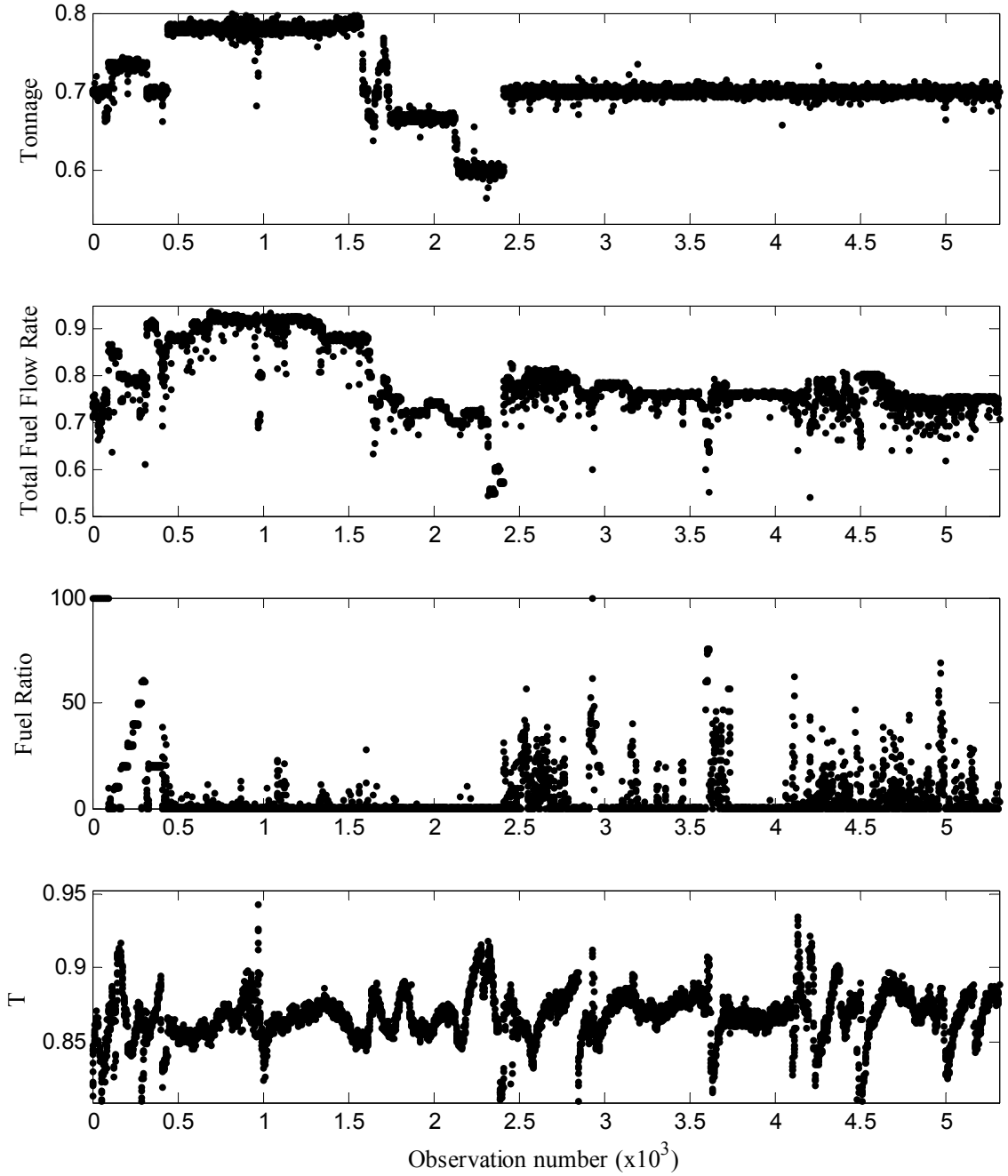


Figure 4: A sample of the data collected on the industrial rotary kiln

Chapter 4 – Methodology

Explanation of MIA

As previously already stated, MIA is being used because of its suitability to deal with the highly turbulent nature of flames; flames will change shape, will bounce up and down and will do all this at a high frequency and therefore making this technique so relevant.

Prior to developing models between flame images and solids discharge temperature, one needs to extract the features of flame images in order to formulate the regression problem. It has been shown by Yu and MacGregor (2004) that flame color features can be efficiently extracted using a MIA technique, since it classifies the image pixels according to their spectral characteristics (e.g. combinations of RGB intensities) without considering their spatial position. This means that pixels having a similar coloration (i.e. similar heat release) will be projected in a similar region of the MIA low dimensional feature space even if they are located differently in the image. This is a very useful characteristic since the objective of this work is to develop a model between heat released by the flames (i.e. flame color) and product quality rather than tracking the position of highly turbulent flames that bounce around constantly. MIA therefore allows extracting flame information without first locating the flame within the image. This is a major advantage compared to conventional flame image analysis techniques used in previous research working directly in the image space, hence requiring additional computing time.

The MIA technique was originally introduced by Esbensen and Geladi (1989). A concise overview of both the method and its history is provided in Geladi and Grahn (1996). In

addition, the following papers also provide some recent developments for quality control applications:

- Bharati and MacGregor, (1998), deals with quality issues regarding the Real Time monitoring using color flame images and MIA;
- Yu et al. (2003) use MIA and color images to monitor quality of a snack food process;
- Bharati et al. (2003) use MIA and color images for Softwood Lumber grading;

A schematic description of MIA is presented in the Figure 5.

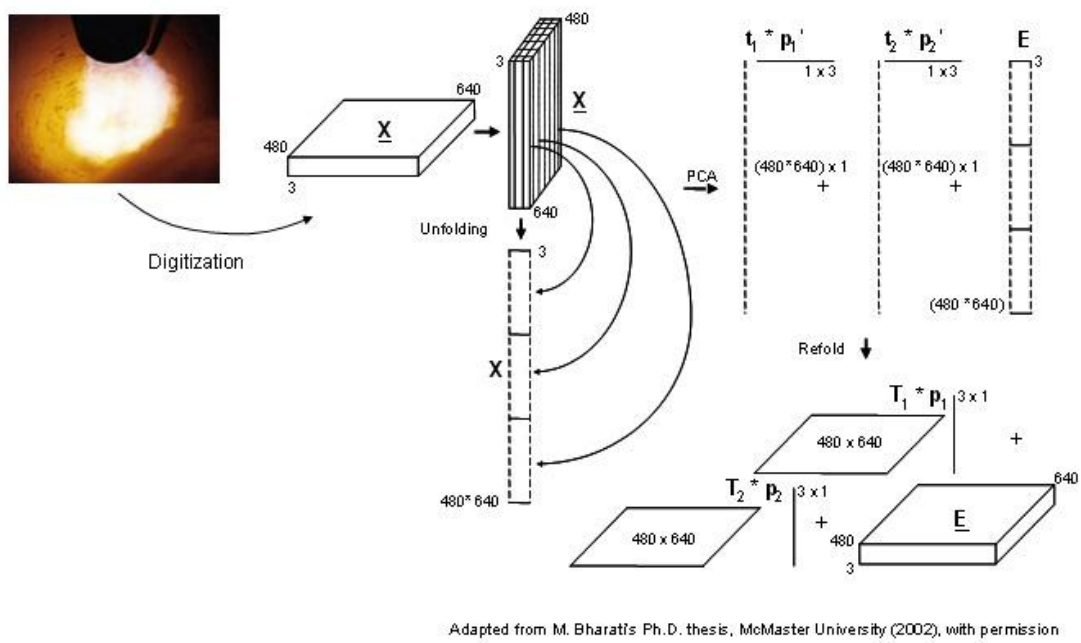


Figure 5: Schematic description of MIA

MIA essentially consists in performing a Multi-Way Principal Component Analysis (MPCA) on a digital multivariate image. This involves two steps. First, the digital image $\underline{\mathbf{X}}$ is unfolded from a three-way array to a two-way matrix $\mathbf{X}$:

$$\underline{\mathbf{X}}_{(\mathbf{Nrow},\mathbf{Ncol},\mathbf{Nspect})} \xrightarrow{\text{unfold}} \mathbf{X}_{(\mathbf{Nrow\ x\ Ncol},\mathbf{Nspect})} \tag{1}$$

where N_{row} and N_{col} correspond to the spatial dimensions of the image (640 and 480 in this study), whereas the third dimension or spectral dimension is identified by N_{spect}. Since the images have three spectral channels (R, G, and B), N_{spect} equals 3. This unfolding operation collects the RGB intensities of each pixel row wise in matrix **X** independently of their spatial location, but the identity of each pixel is kept in memory for future correspondence, as will be discussed later**.** The spatial information is therefore not being used from this stage on, only spectral intensity will be important and **this is how MIA can deal with bouncing flames, by classifying the pixels according to their spectral information only, regardless of their location in the image.** Second, PCA is performed on the unfolded digital image **X**:

$$\mathbf{X} = \sum_{a=1}^{K} \mathbf{t}_a \, \mathbf{p}_a^T + \mathbf{E} \tag{2}$$

where K is the number of principal components, the $\mathbf{t_a}$ vectors are the score vectors, and the corresponding $\mathbf{p_a}$ vectors are the loading vectors. For RGB images, the maximum number of components is 3. If K < 3, then **E** contains the residuals of the PCA decomposition. A kernel algorithm (Geladi and Grahn, 1996) is typically used to compute this decomposition since **X** has a very large number of rows (N = 640 × 480 = 307200) and a small number of columns (3). In this algorithm, the loadings vectors ($\mathbf{p_a}$) are obtained from a singular value decomposition (SVD) of the very low dimensional kernel matrix $\mathbf{X}^T\mathbf{X}$ (only 3 × 3 for an RGB image). The score vectors are then computed using $\mathbf{t_a} = \mathbf{X}\,\mathbf{p_a}$. The MIA technique as described previously is used for the analysis of a single image. When MIA is to be used for the analysis of a set of J images, then the kernel matrix is calculated as $\sum_{i=1}^{J} \mathbf{X}_i^T \mathbf{X}_i$ and then SVD is performed on that summation matrix to calculate the loading vectors.

As for the analysis of data matrices using PCA, the interpretation of image features is performed using score plots, and particularly t_1-t_2 score plots since in most MIA applications using RGB images, the first two principal components explain most of the variance. However, due to the very large number of score values typically encountered in image analysis (total number of pixels or 307200 for a 640 × 480 image), the score plots are

usually displayed as 2-D density histograms and shown as images themselves to enhance their visual appearance. To obtain such a score histogram, denoted as **TT**, the t_1-t_2 score plots is first divided into a number of bins, usually 256 × 256, and the pixels falling into each bin are then counted and stored in matrix **TT** at the corresponding bin location. After selecting a proper color map proportional to the pixel density in each bin, the 2-D score density histogram **TT** (256 × 256 × 1) can be displayed as an image (see Figure 6). Every point of the 2-D score density histogram corresponds to a pixel of the image. Pixels having similar spectral (i.e. color) characteristics will fall within the same region of the score histogram whereas those pixels having different coloration will project in a different regions. One can therefore use segmentation techniques to identify relevant features such as the flame itself in the score histogram instead of directly in the image and, therefore, avoid locating the flame in the orginal image.

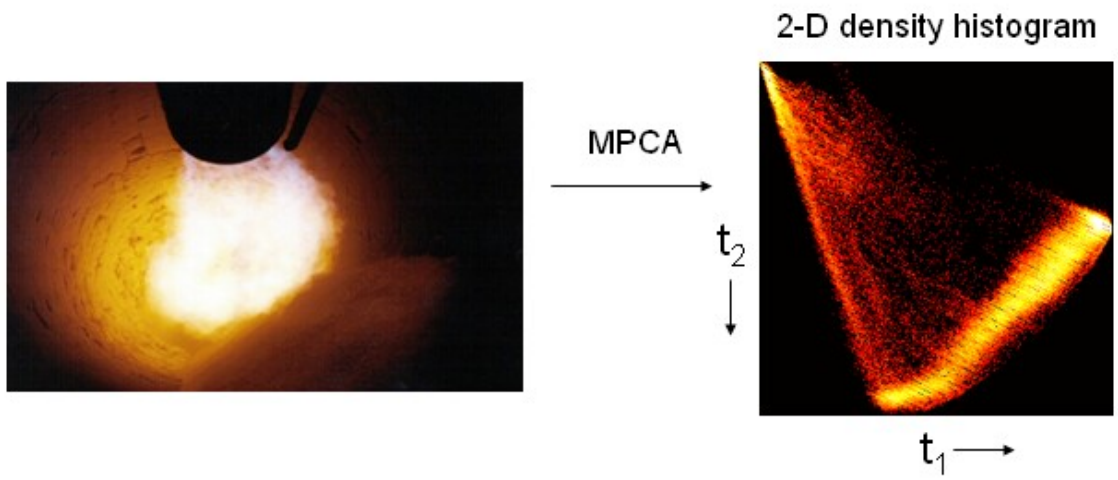


Figure 6: The visualization of the score histogram

When a set of images are analyzed using MIA, a common scaling range is used for the scores prior to compute the density histograms. This scaling range corresponds to the minimum and maximum values of all t_1 and t_2 score vectors of the set of images.

The score histograms are useful to visually identify the information extracted from the original image by each principal component. By masking a region in this score histogram, one can identify all pixels falling under that mask (since the identity of each pixel is kept in

memory) and highlight them in the original image. This operation is shown in Figure 7, where a blue mask is drawn on the score histogram (image on the left). The pixels falling under this mask are shown using the same blue color in the orginal flame image (image on the right). This ability to retrace a region of the score plot in the original image has been previously presented in the paper of Yu and MacGregor (2004).

An alternative way to interpret the image information is to refold the $\mathbf{t_a}$ (N_{row} $N_{col} \times 1$) score vectors into a three-way array $\mathbf{T_a}$ ($N_{row} \times N_{col} \times 1$) according to the same spatial coordinates as in the original image $\underline{\mathbf{X}}$ (i.e. pixel locations), and then show each $\mathbf{T_a}$ as a univariate image.

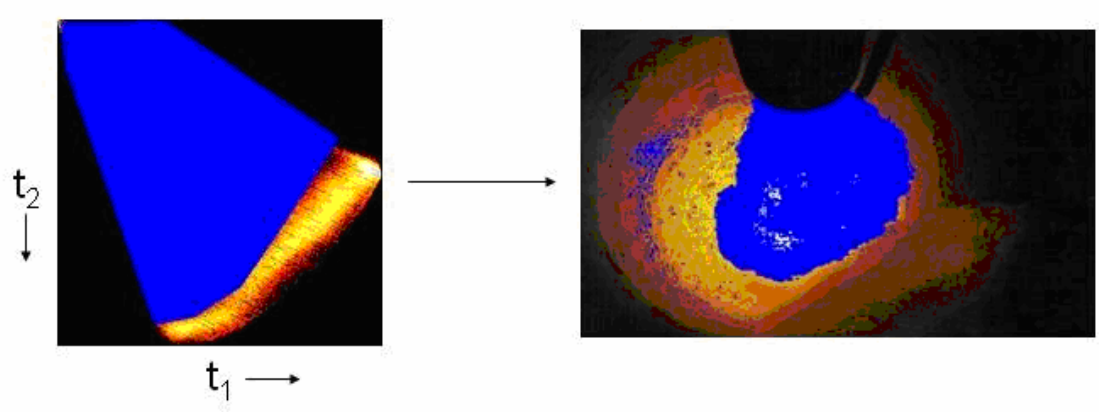


Figure 7: Correspondence between Real Image and the 2D Density Histogram

Explanation of MIR/PLS

The aim of this work is to build a dynamic model between the flame color features extracted from each image and the corresponding solids discharge temperature measurement (i.e. quality variable). This can be accomplished using Multivariate Image Regression (MIR), which refers to a family of techniques used for regressing quality or response variables on image features. Image regression problems can be formulated in several ways, depending on the image feature extraction method. In this study, solids discharge temperature is regressed on the t_1-t_2 score density histograms (i.e. distribution features) obtained from flame images using MIA (see previous section). However, prior to build regression models, one needs to solve the dimensionality issue arising from the fact that for each score histogram (i.e. a matrix of dimension $N_{bt1} \times N_{bt2}$) correspond a single temperature measurement (i.e. a scalar). Dimensions N_{bt1} and N_{bt2} are the number of bins dividing the score plot along the t_1 and t_2 score axes respectively. This problem was addressed by storing the elements of each score histogram matrix (i.e. number of pixels falling into each bin) row wise in a new matrix $\mathbf{X_{MIR}}$ as follows:

$$\mathbf{X_{MIR}}(i,1{:}\,N_{bt1}\times N_{bt2}) = [\mathbf{TT}_i(1,1{:}\,N_{bt1})\ \mathbf{TT}_i(2,1{:}\,N_{bt1})\ \ldots\ \mathbf{TT}_i(N_{bt2},1{:}\,N_{bt1})] \quad i = 1,2,\ldots,J \qquad (3)$$

This procedure is schematically shown in the Figure 8, for the score histogram of flame image I ($\mathbf{TT}_i$) with $N_{bt1} = N_{bt2} = 29$ (i.e. score histogram divided using a grid of 29×29). In this way, the t_1-t_2 score density histogram information obtained for image I is all contained in row I of $\mathbf{X_{MIR}}$ whereas the corresponding quality measurement is stored in the ith row of the quality matrix $\mathbf{Y}$. Any appropriate regression method can then be used to build a model between $\mathbf{X_{MIR}}$ and $\mathbf{Y}$, such as Ordinary Least Squares (OLS), Partial Least Squares (PLS), etc., and even non-linear regression models such as Neural Networks. For this regression to be meaningful, however, one must absolutely make sure that a common scaling range has been applied to the t_1-t_2 score density histograms before storing their elements in $\mathbf{X_{MIR}}$,

and that these elements are always stored in the same order to preserve information congruency.

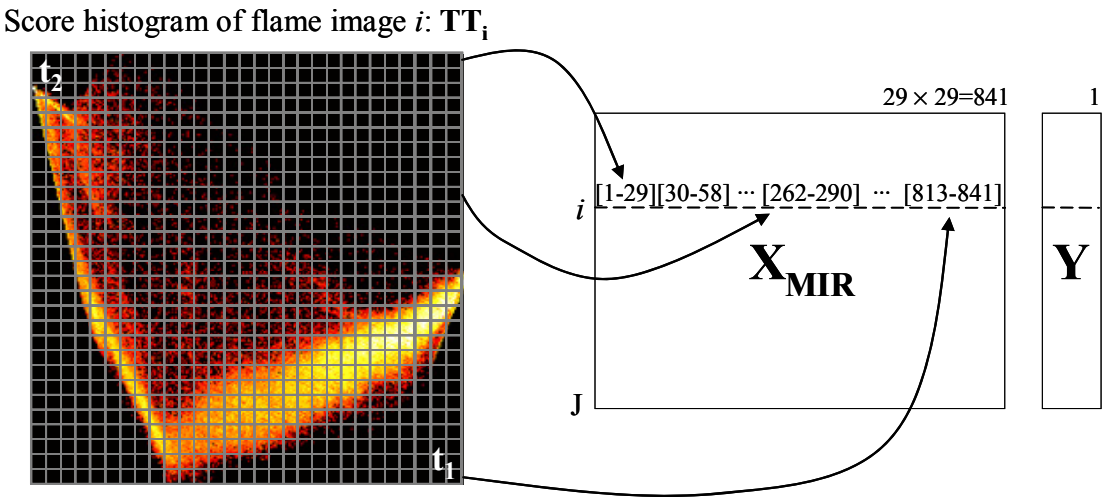


Figure 8: Multivariate Image Regression (MIR) problem formulation

Finally, dividing score plots using a grid of 256×256 is appropriate for image interpretation and visualization using MIA. For image regression purposes, however, such a resolution is often unnecessarily high, and would unduly increase computing time and memory requirements. Most previous work on MIR (Yu and MacGregor, 2003) make use of a 32×32 grid. After some testing, a grid of 29×29 bins was selected for convenience, but no further improvements in the results were obtained by increasing the grid resolution.

Explanation of Neural Network Models

As alternative to MIR/PLS, a non-linear neural network model has been used in order to verify if the performances of the linear PLS model could be improved. The neural network model used to approximate the relationship between the discharge temperature (denoted with y) and the image feature vector (denoted with $\mathbf{x}$) is a multi-layer perceptron neural network (MLP) (Rumelhart et al., 1986). This is by far the most common artificial neural network type used in function approximation applications. A brief description of the neural network modeling procedure as given in Tarca (2004) follows.

Considered as "black-box" modeling tools, the neural networks have gained enormous popularity in many other engineering fields, perhaps due not only to their appealing "learning ability," but also because of versatility and performance with respect to classical statistical methods. Without assuming a particular equational form (e.g. linear), a neural network function is able to mimic complex nonlinear relationships between the input feature vector $\mathbf{x}$ and a dependent (output) variable y by consuming the information in a set of training samples of known input and output values. The parameterized neural network fitting function with a single output (approximating thus a scalar) has the form:

$$f(\mathbf{x},\mathbf{w}) = \sigma^{(2)}\left(\sum_{j=1}^{J+1}\left(w_j \cdot \sigma_j^{(1)}\left(\sum_{i=1}^{I+1}(x_i \cdot w_{i,j})\right)\right)\right) \tag{4}$$

where $\mathbf{w}$ are the free parameters called weights and I is the dimensionality of the input vector. The J activation functions in the first layer $\sigma_j^{(1)}$ and the single one in the output layer $\sigma^{(2)}$ are sigmoid functions, $\sigma(z) = \frac{1}{1+e^{-z}}$, while the I+1 component of the feature vector, x_{I+1}, and the J+1 activation function, $\sigma_{J+1}^{(1)}$, are set to a constant value of 1. Training of the neural network means determining the parameters $\mathbf{w}$ in such a way that the estimate produced by the trained neural network $\hat{y}(\mathbf{x}) = f(\mathbf{x},\hat{\mathbf{w}})$ closely approaches the true value $y(\mathbf{x})$ on a set of training samples (the design set) $D = \{(\mathbf{x}_r, y(\mathbf{x}_r)), r = 1...n\}$. In our case the

number of training samples was 5000. The training algorithm minimizes the sum of squared prediction errors on the training samples using a gradient-based technique. The training of the neural network function was done in this study using the Levenberg-Marquardt algorithm implemented in MATLAB® software. After the parameters of the model, **w,** were estimated, the neural network function (Eq. 4) is used to predict the discharge temperature for new data samples. For this study a monotonic network has been used (see below Fig. 9).

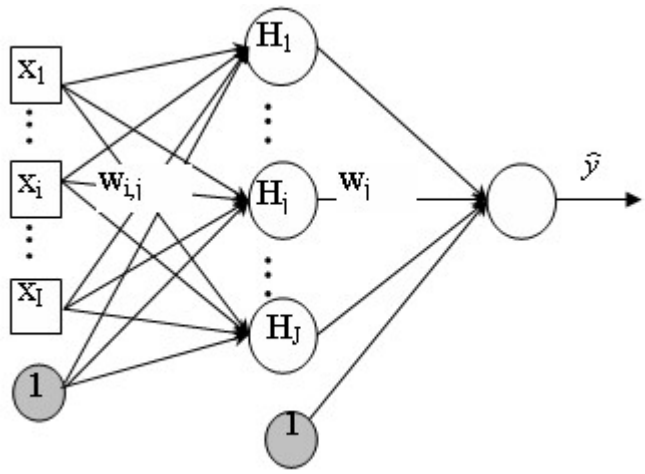


Fig. 9 Typical monotonic network (with permission from Tarca et al. 2004)

Chapter 5 - Methodology Implementation

Technology

To build and validate all prediction models, the Matlab software, a MS SQL database as well as the Java programming environment were used according to the functional diagram below (Figure 10).

Matlab has been used to create the model and launch all the model functions. Matlab has been chosen due to the powerful mathematical functions as well as the Image Processing Toolbox. The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB® numeric computing environment. The toolbox supports a wide range of image processing operations, including spatial image transformations, morphological operations, neighborhood and block operations, linear filtering and filter design, transforms, image analysis and enhancement, image registration, deblurring as well as region of interest operations.

The MS SQL database has been used to store all the data available whether it is process data or image information data. This has been chosen in order to improve the productivity in working with data, due to the fact that an important amount of data is involved. MS SQL database enhances the productivity and provides easier access to data in various formats. Also the data can be viewed and extracted according to any desired criteria.

Several productivity tools have been created and they were programmed using the Java programming language. Java programming language has the advantage of being cross platform as well as the fact that Matlab integrates with it in a seamless way. Also the bridge between Java and MS SQL database is a popular one, fact that also improves productivity.

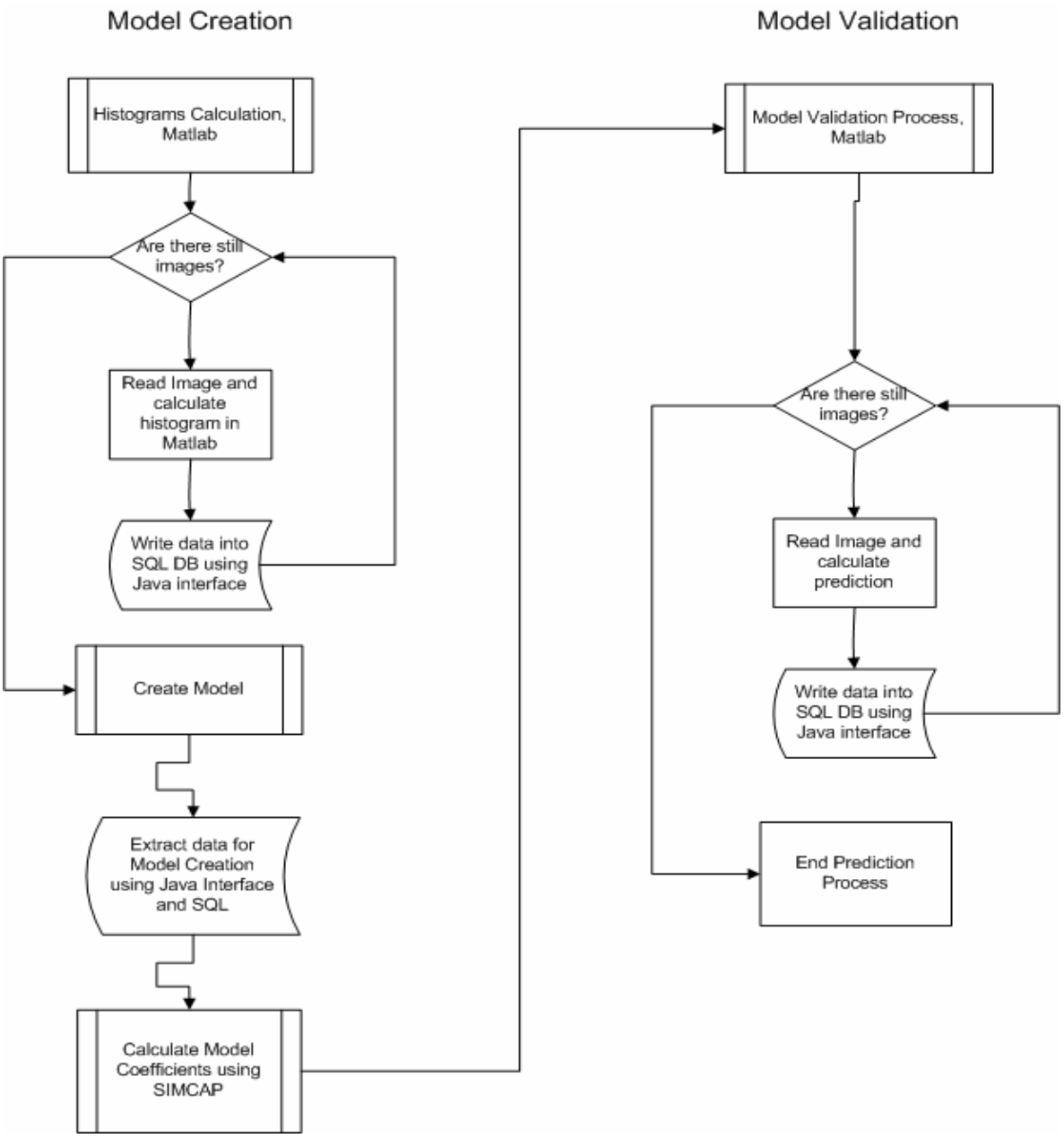


Figure 10: Technology Diagram.

For the Multivariate Image Regression models built between $\mathbf{X_{MIR}}$ and $\mathbf{Y}$ using PLS, the SIMCA-P V10® (Umetrics Inc.) software has been employed.

Prediction Model Building Procedure

The procedure consists of three main steps:

- Image and process data selection
- Image processing
- Model construction
- Model validation using predictions

See Annex 1 for the Matlab, MS SQL as well as Java code.

Image and process data selection

The first step in the model construction was to remove the outliers, mostly caused by sudden kiln shutdowns and also when the lower detectability limit of the optical IR pyrometer was reached. For that purpose, only the data for which no operation disruption (kiln shutdown) occurred within an 80 minutes window from current time was kept for model development. Kiln shutdowns can be detected in a very straightforward manner just by looking at the measured discharge temperature which drops drastically after the the fuel flow rate is cut-off. From the 80458 original images, only 53300 satisfied the above criteria and from this amount, 5300 images were used for model building and 48000 images were kept for model validation.

Image Processing

After selecting images to incorporate into the model they were processed using MIA, sliced in 29x29 histograms and inserted in the database along with image related information.

Images are synchronized at this stage with process data information so that there is a one to one relationship between an image and process data records.

This processing is done in two steps, first the global Variance Covariance matrix was computed using all the images used for model construction. In order to do this all the variance covariance matrices particular to every image processed are summed up, thus obtaining the Global Variance Covariance matrix. The global minimum and maximum values of the t_1 and t_2 score vectors were obtained simultaneously. This is being done by processing all the images that were selected for model construction. At this stage it is very important to have images that reflect all the operating condition of the process, the inability to comply with this will result in out of the range values. This later situation is taken into account by the software developed for predicting discharge temperatures.

The saved global Variance Covariance matrix along with minimum and maximum score values were used to reprocess each image again. At this stage a 2D density histogram is obtained and is reorganized in a 29 * 29 matrix that is saved in the database.

Model construction

The histograms that are associated to each image are exported from the database along with corresponding process data and imported in SIMCA-P V10®. This software is then used to construct a PLS model, and then extract the regression coefficients that are associated to each histogram bin. The models were constructed to forecast the discharge temperature over a certain period of time, such as 20, 40, 60, and 80 minutes in the future.

Having the image information along with process data allowed the development of various ways to extract interesting information and use it for model creation.

Apart for the PLS model that has been created using SIMCAP® a separate Neural Network Model has been developed as well in order to compare results and test the Plug and Play model creation methodology. For the NN model, a multi-layer perceptron (MLP) neural network model has been used.

Model validation using predictions

Using the models that have been obtained as described at the point above, the remainder images have been used to predict the discharge temperature. The predicted discharge temperature has been then compared to the actual measured temperature in order to evaluate the model's performance.

Plug and Play Methodology

An interesting approach has been to create the methodology in such a way so that is allows the ability to change the type of models used for prediction. That is why in this study two types of models have been used (PLS and Neural Networks). Other type of regression techniques could be used as well just by replacing the mode and changing the coefficients that would enter the prediction model.

Chapter 6 - Results

Initial Tests

Before starting the actual work using images a series of tests have been carried out in order to verify whether or not it was possible to build an empirical prediction model using data only, which is the conventional approach. Therefore a model has been constructed using process data, from which only the most reliable inputs (discussed in Chapter 3) have been used and the results have been compared with a basic model constructed with images. In these initial tests we tried to build multi-input one output (MISO) finite impulse response models between process variables and discharge temperature. No autoregressive components were used in those models in order to make a fair comparison with image based models which are not using autoregressive components on discharge temperature neither. The prediction results of both models are shown in Figure 11 and Figure 12. For the data used in this work it is clear that images contain more information to predict discharge temperature compared to process data only. The main reason why this comparison is not totally fair is that the process data does not contain designed experiments on the main manipulated variables which are required to perform model identification properly. However the purpose of this study is not so much to compare image based predictions with data based models, but more to investigate the information content of images. In order to perform a fair comparison of the prediction ability of both classes of models additional experiments and modeling would be required.

The results shown in Figure 11 were obtained using a MISO model constructed with a history of 200 past samples, which roughly corresponds to a 30 minutes history. This can be compared to the results shown in Figure 12 produced using an image based model

predicting discharge temperature using a single image collected at time t, without any history.

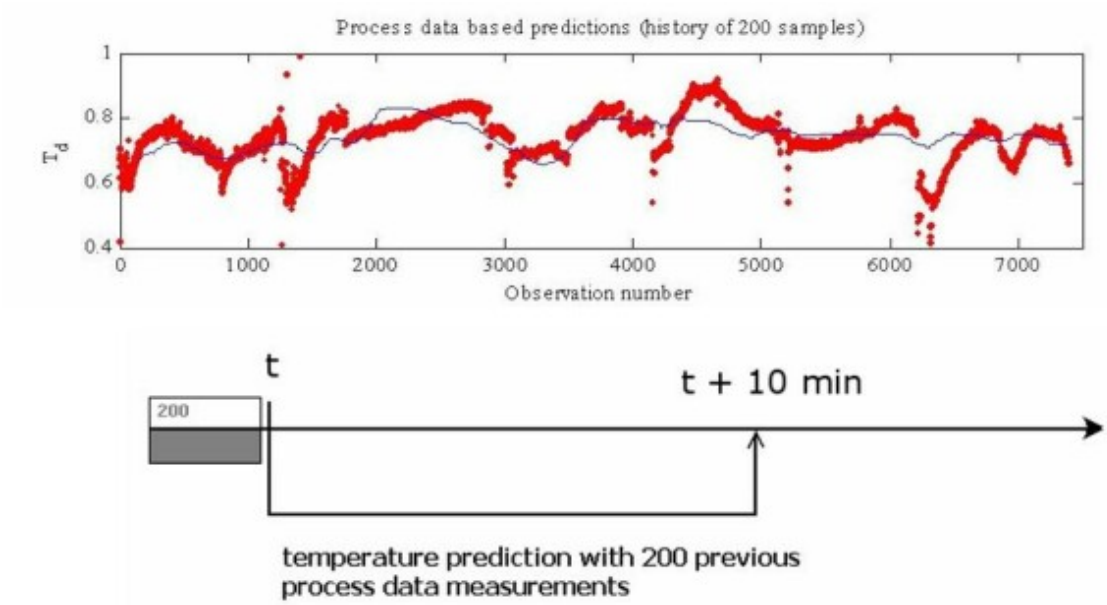


Figure 11: Process based model, 10 min in advance

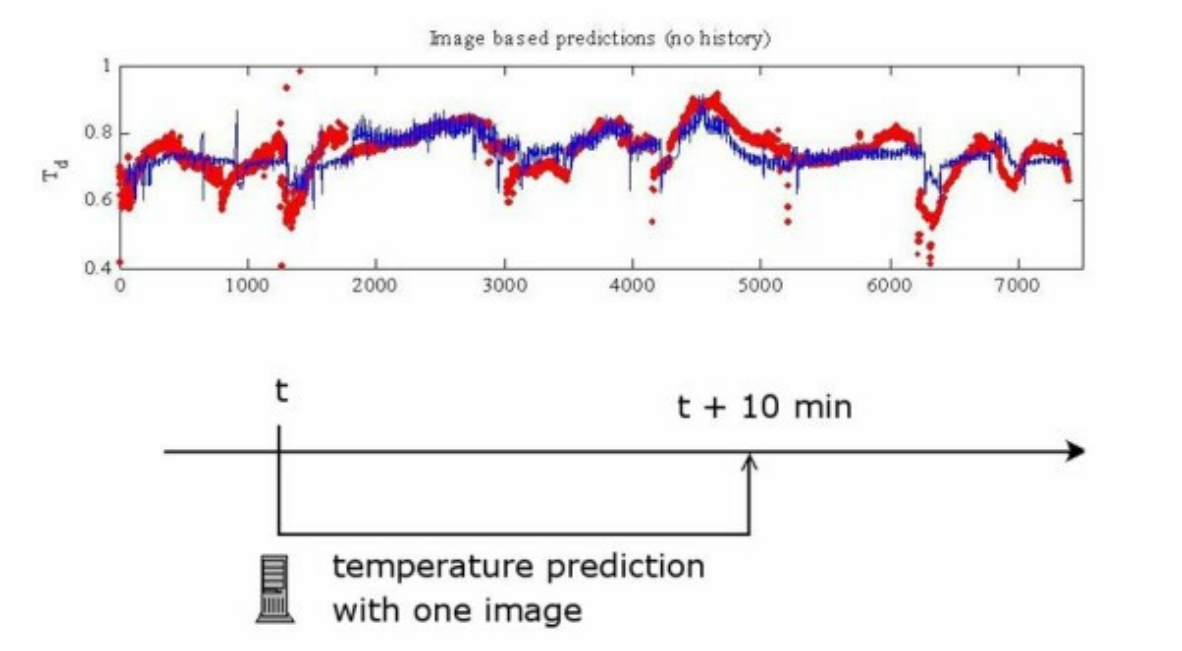


Figure 12: Image based model, 10 min in advance, model constructed without past history data

Prediction of solids discharge temperature from flame images

A total of eight MIR prediction models were built as shown in Table 2 to evaluate the forecast ability of flame images. Each model uses the same predictor matrix $\mathbf{X_{MIR}}$ but a different response matrix **Y**, corresponding to solids discharge temperature collected at different times from the corresponding images. For example, model A uses the discharge temperature collected at the same time as the corresponding image, T(t), whereas model E uses the discharge temperature collected 20 minutes later than the corresponding image, T(t+20 min). The summary statistics for each prediction model are also presented in Table 2. It shows the number of cross-validated PLS components (A) used for each MIR model, three cumulative multiple correlation coefficients ($R^2_{X,cum}$, $R^2_{Y,cum}$, and $Q^2_{Y,cum}$) calculated from model building data as well as the root mean square prediction errors (RMSEP) obtained from validation data. The $R^2_{X,cum}$ statistics correspond to the percentage of the

total variance in the image information ($\mathbf{X_{MIR}}$) used to explain **Y** whereas $R^2_{Y,cum}$ gives the percentage of the total variance of **Y** explained by the model. The cumulative $Q^2_{Y,cum}$ value is the percentage of the total variance of **Y** that can be predicted by the models using a cross validation procedure. These statistics are computed as follows: $Q^2_{Y,cum}(a)=1-PRESS(a)/SS_E(a-1)$. PRESS(a) is the total prediction error sum of squares obtained by cross-validating a model with *a* latent variables and $SS_E(a-1)$ the sum of squares of the residuals of the model using *a*-1 components. As long as Q^2 is greater than zero, the *a*-th dimension is improving the predictive power of model. The $R^2_{Y,cum}$ statistic is the fit multiple correlation coefficient, or alternatively, the explained variance for a model with *a* latent variables and is computed as follows $R^2_{Y,cum}(a)=1-SS_r(a)/SS_{tot}$, where $SS_r(a)$ is the sum of squares explained by the model using *a* components whereas SS_{tot} is the total sum of squares of **Y**. Finally, the $R^2_{X,cum}$ statistic is computed similarly as for $R^2_{Y,cum}$, but is applied to the regressor matrix ($\mathbf{X_{MIR}}$) instead of to the response matrix (**Y**).

Model A shows that as much as 80% of the discharge temperature variations can be explained by the flame color features, and hence confirm that most of the temperature variations are introduced by the combustion process. The remaining 20% may be caused by feed disturbances, such as changes in solids moisture content, composition and feed rate as well as measurement noise. This result also supports that stabilizing the combustion process could significantly reduce temperature variations and therefore shows the importance of monitoring the combustion process using flame images. The fact that 80% of the temperature variations are explained by a single image can be surprising, but it will be shown later that the collected images not only show the flame itself, but also show the kiln walls and the solids. The coloration of these high thermal inertia elements carries information about the thermal history of the solids within the kiln. This will be referred to as the slow dynamics of the kiln, the fast dynamics being the flame itself (combustion). As the forecast horizon increases from 0 to 80 minutes in the future the RMSEP degrades as expected, but at a rate that is slower than linear. Predicting discharge temperature t+80 minutes in the future using the color features of the image collected at time t as the only source of information still allow to explain as much as 30% of the variations.

Table 2: Several model summary statistics comparison

Model	**Y**	A	$R^2_{X,cum}$ (%)	$R^2_{Y,cum}$ (%)	$Q^2_{Y,cum}$ (%)	RMSEP
A	$T(t)$	10	59.8	79.7	77.8	16.22
B	$T(t+5\,\text{min})$	10	59.9	76.4	74.4	17.35
C	$T(t+10\,\text{min})$	10	59.9	71.9	69.4	18.20
D	$T(t+15\,\text{min})$	10	59.8	68.3	66.0	19.10
E	$T(t+20\,\text{min})$	7	52.9	61.8	60.3	20.13
F	$T(t+40\,\text{min})$	7	53.3	49.1	47.1	22.82
G	$T(t+60\,\text{min})$	7	53.3	37.2	34.8	25.30
H	$T(t+80\,\text{min})$	7	53.2	28.3	25.3	27.01

The prediction performance of models A, E, F, and H on the validation data set is shown in Figure 13. A very good agreement between the time series is obtained, even for model H. This shows that flame images contain very useful information to predict the future behaviour of solids discharge temperature and should therefore help improve quality control and reduce fuel consumption.

As shown in Figure 13, very good prediction results can be obtained using a single flame image. However, as mentioned earlier in this paper, a dead-time of about 20 minutes exists between changes in the state of the combustion process and the solids discharge temperature. Considering this dead-time, one would expect that predicting current discharge temperature using the current flame image (Model A) would result in poor performances. However, since very good prediction results were obtained with Model A,

one might therefore suspect that flame images must carry information about the solids thermal history, over some time window, in addition to providing information about the current state of the combustion process. In order to validate this assumption and to provide more insight into the results, an investigation of the most important image features for solids discharge temperature predictions is proposed in the next section.

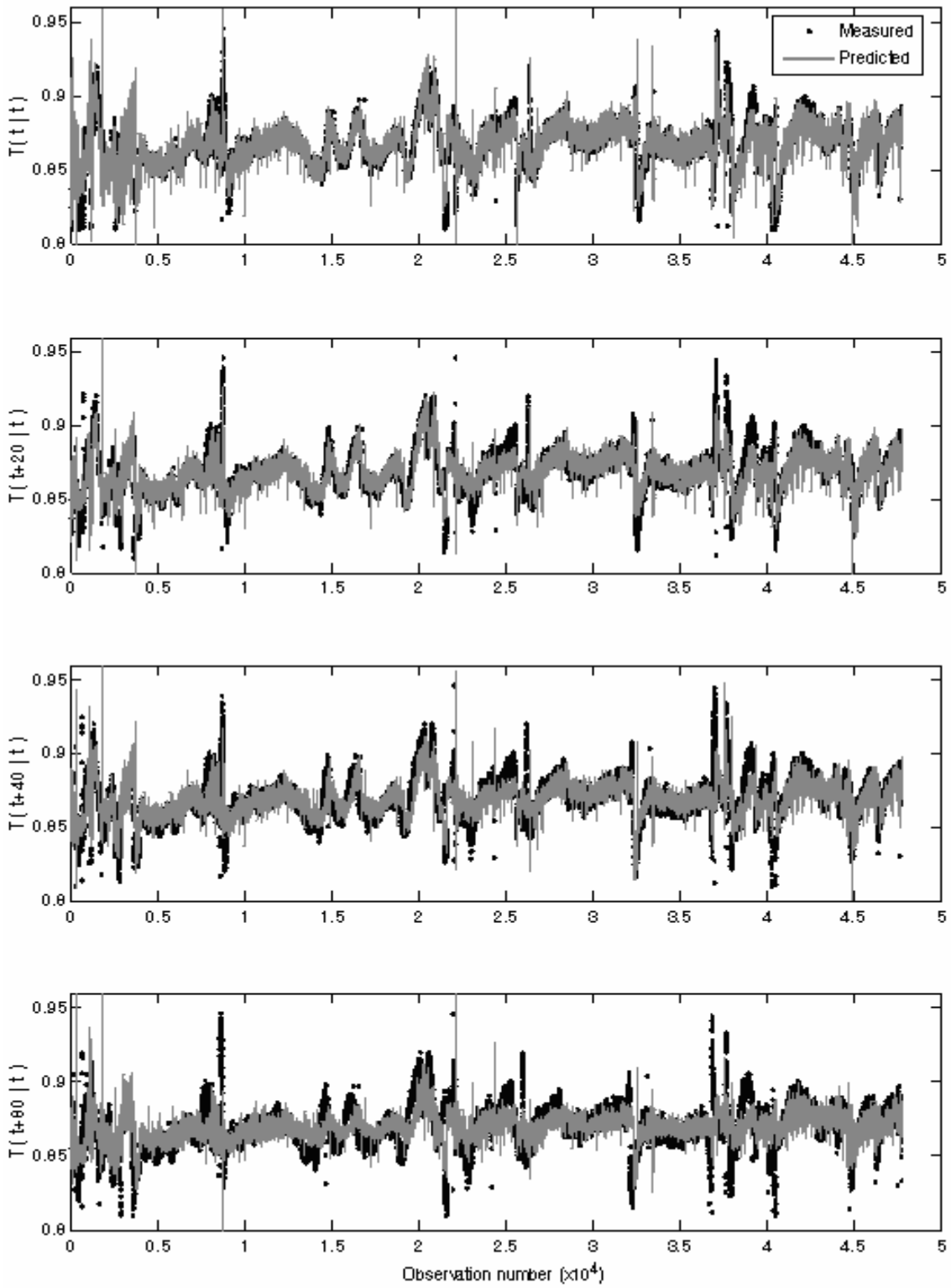


Figure 13: Prediction results for MIR models A, E, F, and H on validation data. Black dots correspond to measured solids discharge temperatures and gray lines to model predictions.

Interpretation of MIR model predictions

Assessing the importance of image features for solids discharge temperature predictions based on MIR models requires two steps: 1) identifying which bins of the score histograms are the most important for predicting **Y**, and 2) mapping these bins in the original image space. The first step consists in quantifying the importance of each regressor variable in $\mathbf{X_{MIR}}$ (e.g. bins of score histograms) in explaining variations in **Y**. This can be achieved using the Variable Importance (VIP) metric commonly used in the chemometrics literature:

$$\mathrm{VIP_k}(\mathrm{a}) = \sqrt{\mathrm{K} \sum_{\mathrm{a}} \mathrm{w}_{\mathrm{ak}}^2 \left(\mathrm{SSY}(\mathrm{a}) / \mathrm{SSY_{tot}} \right)} \tag{5}$$

where $\mathrm{VIP_k}(a)$ is the importance of the kth regressor variable based on a model with a components, w_{ak} is the corresponding loading weight of the kth variable in the ath component obtained using the PLS decomposition algorithm, SSY(a) is the explained sum of squares of **Y** by a PLS model with a component, $\mathrm{SSY_{tot}}$ is the total sum of squares of **Y**, and finally, K is the total number of regressor variables (e.g. total number of bins, 29x29 in this study). Contour plots can be used to visualise the importance of the various bins since each has its own VIP value and correspond to a specific location in the t_1-t_2 score plot. Figure 14 shows two such VIP contour plots for model A and model H respectively. A color map indicates the magnitude of the VIP values for each bins, where the black color means no importance whereas the white color correspond to the highest VIP values of the model (e.g. most important bins or image features). From Figure 14, it is clear the most important image features (e.g. regions of the t_1-t_2 score plots) for predicting solids discharge temperature are similar over the investigated forecast window, from time t to time t+80 minutes. The differences between these two contour plots come from the fact that the VIP values are lower for model H since it explains less **Y** variance.

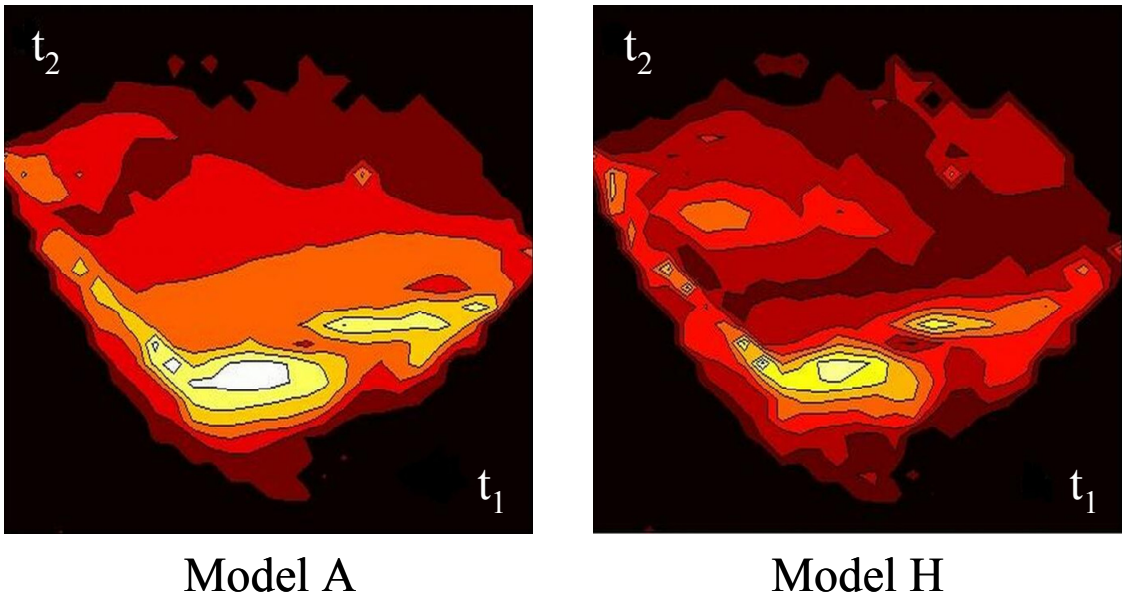


Figure 14: VIP contour plots for MIR models A and H, where orange-white are the highest VIP regions of the model.

The second step consists in mapping the most important t_1-t_2 score regions (e.g. bins), as determined using the VIP plots, onto the original image space to show what aspects of the scene are the most important for discharge temperature predictions. Mapping back and forth between the feature space (t_1-t_2 score plots or histograms) and the original image space is the key strength of MIA (Geladi and Grahn, 1996). Figure 15 shows a mapping of the most important regions of the score space onto a specific flame image. The selected flame image and its corresponding t_1-t_2 score density histogram are shown in Figure 15 a) and b), respectively. Figure 15 d) shows an overlay of the two highest VIP regions based on Model A (see Figure 14) using masks of different colors. The mapping of the pixels falling under these masks onto the original flame image is shown in Figure 15 c) using the same color coding. Note that the shape of the masks is not optimal in any way, but has been selected for illustration purposes only.

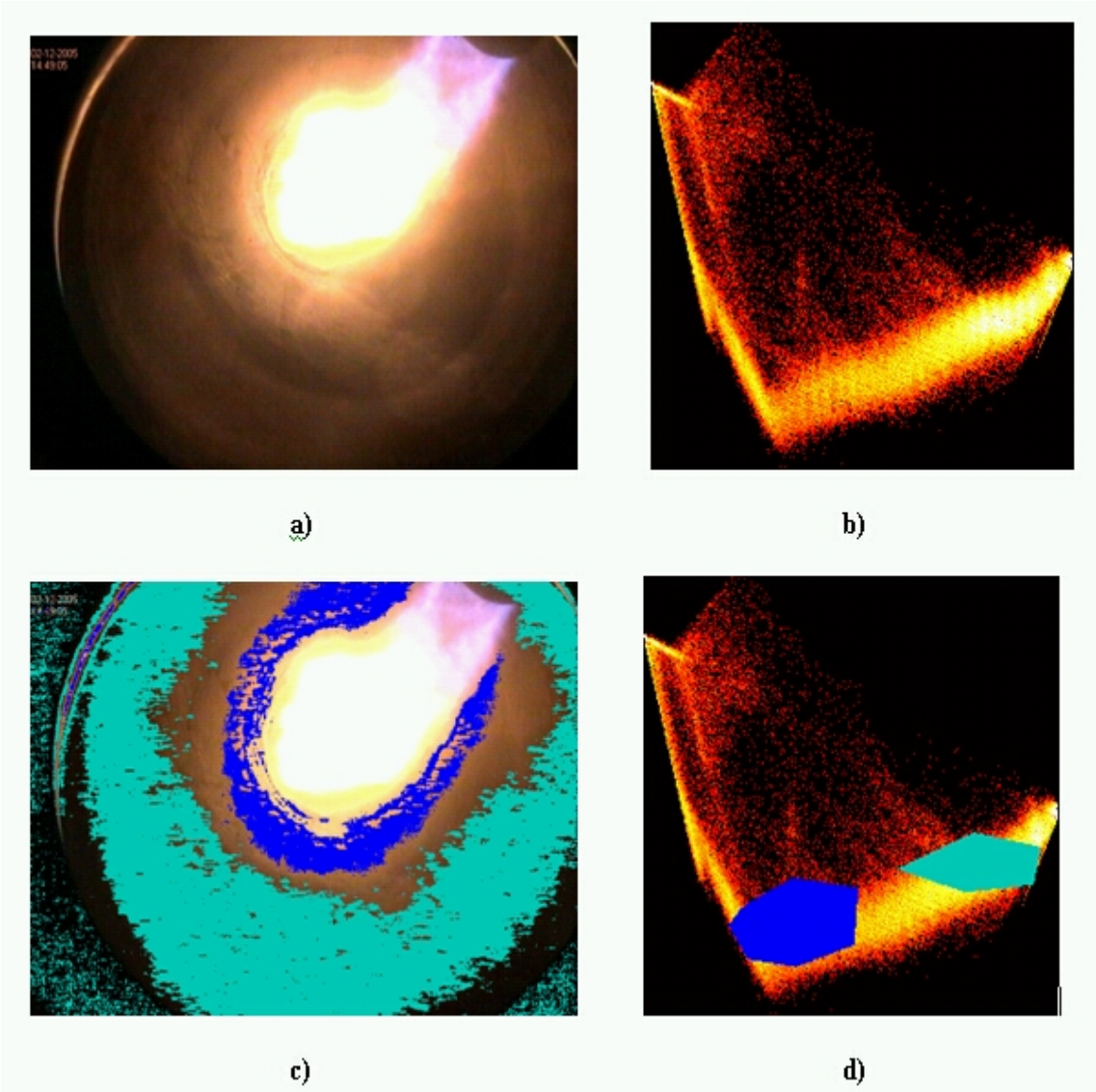


Figure 15: Mapping of the highest VIP regions of the $t_1 - t_2$ scores on a selected flame image based on MIR model A used for solids discharge temperature predictions.

The main conclusion drawn from Figure 15 is that the most important aspects of flame images for predicting solids discharge temperature are the coloration of the solids and of the kiln's walls. This result is very sensible based on chemical engineering knowledge since these internal features are all related to the slow dynamics of the kiln; an increase in the heat released by the combustion achieved, for example, through changes in fuel flow rate does not have an immediate impact on solids discharge temperature since there is a

very important thermal inertia to overcome(this conclusion has been drawn after taking in consideration all the images used for this study).The solids and the kiln's walls will take some time to heat up or to cool down in response to an increase or a decrease in the heat released by the combustion process. In addition, since the flame usually covers one third of the kiln length it affects the solids temperature over a long distance within the kiln. This explains why a single flame image has very good forecast ability for the solids discharge temperature.

Neural Network Model Results

A multilayer neural network model was fitted to the discharge temperature 20 min in advance and tested in the same conditions as the linear model (model E). The input layer correspond to each of the 29x29 bins of the 2-D score histograms. The network has 8 hidden neurons, and uses log transformation for inputs spread on more than 2 decades of values. Bayesian regularization was used for training the model.

The neural network model, although more complex, provided only slightly better prediction capabilities. The comparison criteria are those shown in table 3 and 4 below. Figure 16 also show the prediction performance of the Neural Network model compared to measured data.

Table 3: Neural Network Results Comparison for model E (t+20 min)

	Linear model	**NN model**
R^2 between pred (predicted values) and true (real data)	76.1%	80.5%
Mean Square Error	573	481

Table 4: Quantiles of errors

	0%	**25%**	**50%**	**75%**	**100%**
Linear	-290.5	-11.40	0.5069	11.61	191.1
NN	-253.9	-10.42	0.1830	10.67	218.9

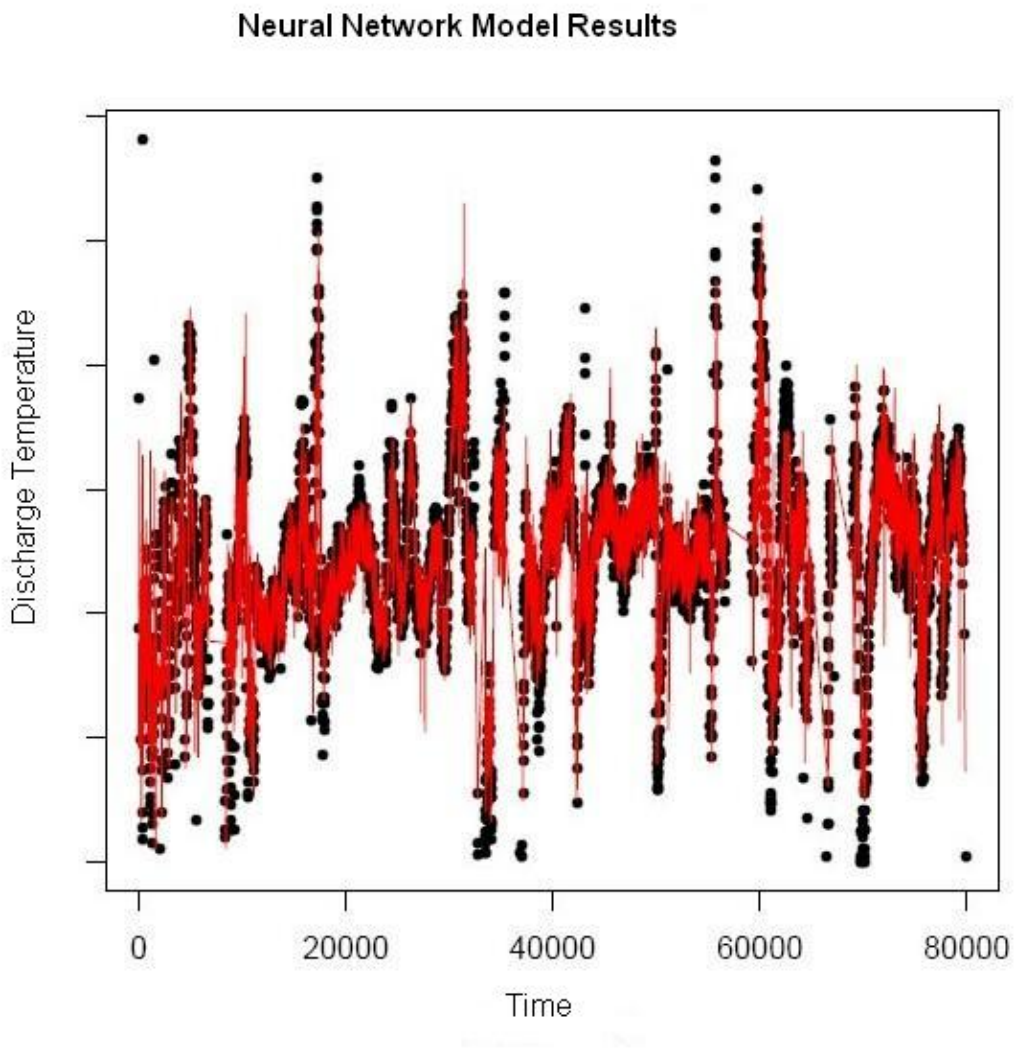


Figure 16: NN predicted vs. real points distribution

From the data presented Neural Networks give good results, however due to their complexity and to the fact that the gain is not significant, has been decided to continue with PLS which is more robust, well known and relatively easy to use. See the Annex 2 for detailed comparison of the data obtained from the Neural Network Model versus a Linear PLS Model.

Potential Quality Control Strategies based on Flame Images

It has been shown in the previous section that the proposed MIR models provide very good predictions for the current and future behavior of the solids discharge temperature, $\hat{T}(t+k|t)$ k=0, 5, …, 80 min, given a single flame image collected at time t. These models could be used as such when kilns are manually controlled, as is often the case in practice, to help kiln operators take appropriate control decisions, providing them with an early warning of the production of off-specification materials or of the onset of automatic kiln shutdown events. However, a more efficient use of these predictions would be to integrate them into automatic quality control strategies. Obviously, implementing a simple PID feedback control scheme manipulating the total fuel flow rate to control solids discharge temperature (measured with an IR pyrometer) could yield a significant reduction in temperature variability over manual control. However, this control scheme has some limitations, since it reacts upon observed changes in the controlled variable (i.e. solids discharge temperature) and fight against the process dead-time to reject disturbances. Considering the relatively tight difference between the high and low temperature limits of the industrial case study presented in this paper, as well as the 20 minutes dead-time(determined by QIT personnel, based on process data and knowledge), it is expected that a simple feedback control scheme will not be enough to prevent hitting the two temperature constraints. Further reduction in the discharge temperature variability is required which, in addition, will help in reducing overheating and fuel consumption. The aim of this section is therefore to discuss ways to enhance simple feedback quality control through an efficient use of flame image based discharge temperature predictions.

A modified Smith Predictor Control Scheme

One commonly used feedback control enhancement when important process dead-times exist is the Smith predictor. The main idea is to control the process output without dead-time, obtained using some process model, and to correct for model mismatch through the feedback loop. MIR model E provides very good forecasts of the solids discharge temperature obtained after the dead-time period $\hat{T}(t+20\,\text{min}|t)$ (see Table 2 and Figure 13) and hence, could be as the delay-free model of a Smith predictor control scheme. Figure 17 shows a modified Smith predictor control scheme using a flame image based prediction model.

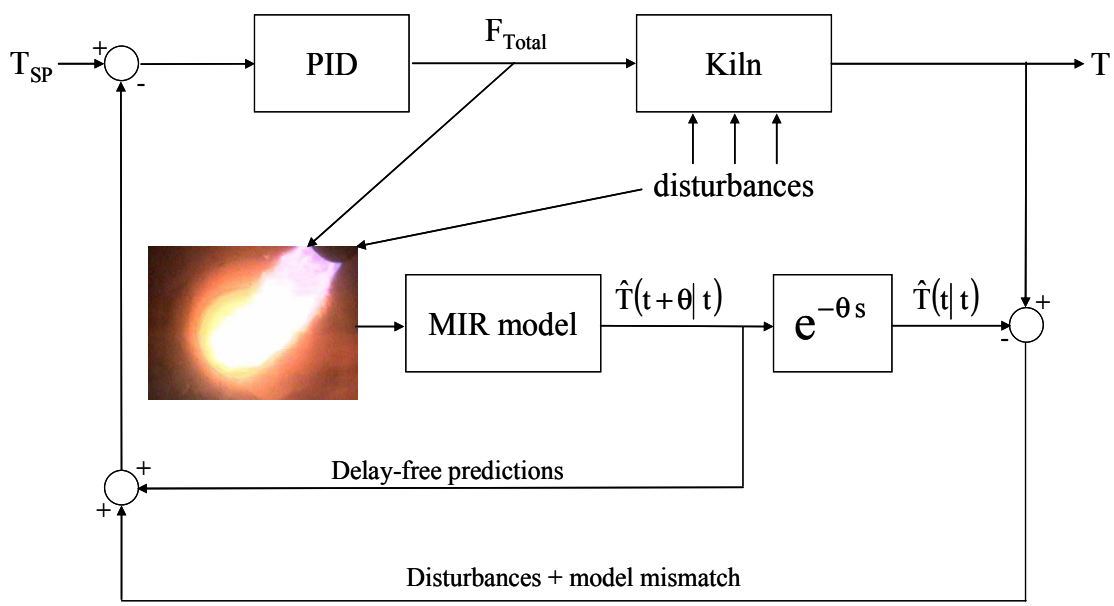


Figure 17: Modified Smith Predictor control scheme based on flame image predictions.

In Figure 17, T represents the solids discharge temperature as measured using an IR pyrometer, T_{SP} is the temperature set-point, F_{Total} is the total fuel flow rate (manipulated variable) and θ is the dead-time between F_{Total} and T (θ = 20 minutes in this work). For the industrial case study considered in this work, the dead-time of the transfer function between

F_{Total} and T is about half of the time constant. One should therefore expect some improvements with respect to feedback control only.

A new Cascade Control scheme involving image based combustion control

Cascade control is a useful enhancement to prevent fast disturbances, introduced by the manipulated variables themselves, affecting process outputs, which dynamics is usually slower. This situation arises in the present study. About 80% of the solids discharge temperature variations can be explained by the combustion process itself (i.e. manipulated variable), which dynamics is much faster than that of the solids and kiln's walls. Reducing variability in the combustion process through tighter control of the visual appearance of the flame (obtained by imaging) could lead to very significant reduction in solids discharge temperature by preventing combustion disturbances to propagate through the slower process dynamics (i.e. heat transfer to solids and walls). Such a cascade control scheme is presented in Figure 18. The inner controller would adjust the total fuel flow rate (F_{Total}) to maintain the flame visual appearance (i.e. heat released) at a desired set-point. To quantify the flame visual appearance, one could use the score values of MIR model A, as will be discussed later. As shown by Liu and MacGregor (2005) control calculations based on images can be formulated as a constrained optimization problem, where the implemented value of total fuel flow rate would be the one that minimizes some distance measure J of the appearance of the current flame image from set-point, given measured (fuel ratio) and unmeasured combustion disturbances. Finally, the outer control loop could be a simple PID feedback loop aimed at correcting for feed disturbances and throughput changes. If the flame visual appearance set-point requires more than one latent variable, for exemple two as in Figure 18, then a multivariable controller is necessary. Multi-loop PID control system could be used in this case and would be easy to tune since the latent variables are independent (i.e. equivalent to tuning single input single output PID controllers). The PID controller was mentioned here since it is simple and widely used. However, any appropriate control can be used in that loop. Finally, if necessary, the Smith predictor control scheme shown in Figure 17 could also be implemented on the cascade control of Figure 18 (not shown to simplify drawing of that Figure).

Developing such an image-based combustion control scheme (inner loop in Figure 18) requires 1) a causal relationship between flame appearance and solids discharge temperature, and 2) building a causal model between total fuel flow rate and fuel ratio (manipulated and disturbance variable of the combustion process) and the flame visual features (necessary for control calculations). It has been shown earlier in this paper that the former requirement is met; there exist a physical relationship between the combustion process and the solids discharge temperature and this has been quantified by the MIR models developed. For the latter requirement, the data available in this study did not allow to obtain such a causal model directly due to the lack of designed experiments. This will be the subject of future work. However, the available data is sufficiently informative to discuss the feasibility of image-based combustion control.

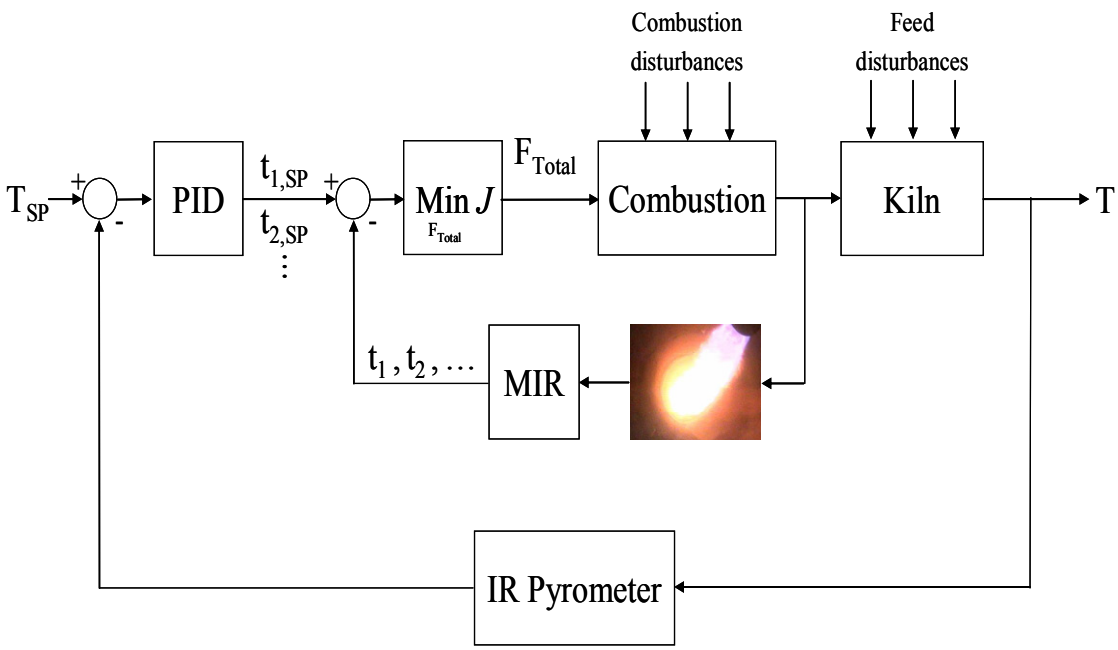


Figure 18: A new Cascade Control scheme involving an image based combustion controller (slave controller).

To build a causal model between total fuel flow rate and fuel ratio and the flame visual features one needs to performed designed experiments (DOE) on the former two combustion related variables. Fuel ratio is a constantly varying disturbance in normal operation. However, when the process fuel (fuel A) supply is not limited, it is possible for the plant operators to independently manipulate the fuel ratio. Such a DOE was not

performed in this study. However, to show that it is possible to detect variations in total fuel flow rate and fuel ratio using flame images, two new MIR models were developed between flame images and combustion variables. In each of these models, matrix $\mathbf{X_{MIR}}$ still contains the color features of the flame images, but matrix **Y** now corresponds to the fuel ratio (disturbance variable) and to the total fuel flow rate (manipulated variable), respectively. No dynamics was assumed in these models (i.e. flame image collected at time t predicts combustion conditions prevailing at time t) since the dynamics of the combustion process is extremely fast. Each model was built on a different data set, which are both subsets of the data used for discharge temperature predictions. The data used to build the prediction model for fuel ratio (model I) and total fuel flow rate (model J) respectively correspond to the first 300 samples and to the first 2051 samples. The former data set comes from a short design of experiments performed on fuel ratio only whereas the latter data set was chosen due to the wide range of variations in the total fuel flow. Table 5 gives the summary statistics for MIR models I and J and Figure 19 shows the prediction results.

Table 5: Summary statistics for models I and J

Model	**Y**	A	$R^2_{X,cum}$ (%)	$R^2_{Y,cum}$ (%)	$Q^2_{Y,cum}$ (%)
I	Fuel Ratio	5	54.0	97.2	96.0
J	Total Fuel Flow	6	48.2	89.6	87.9

Using PLS again, very good predictions are obtained after regressing the combustion conditions on the color features of the flame image. The explained variance of each combustion variable in fit and validation ($R^2_{Y,cum}$ and $Q^2_{Y,cum}$) are high and the predicted trends are in good agreement with the measurements. This confirms that the visual characteristics of flame images reflect the variations in the main disturbance of the combustion process (fuel ratio) as well as variations in the manipulated variable (total fuel flow rate). Therefore, building a good causal model from DOE data between the two

combustion variables and the visual features of flame images should pose no problem (i.e. flame images can be used as a sensor for combustion control).

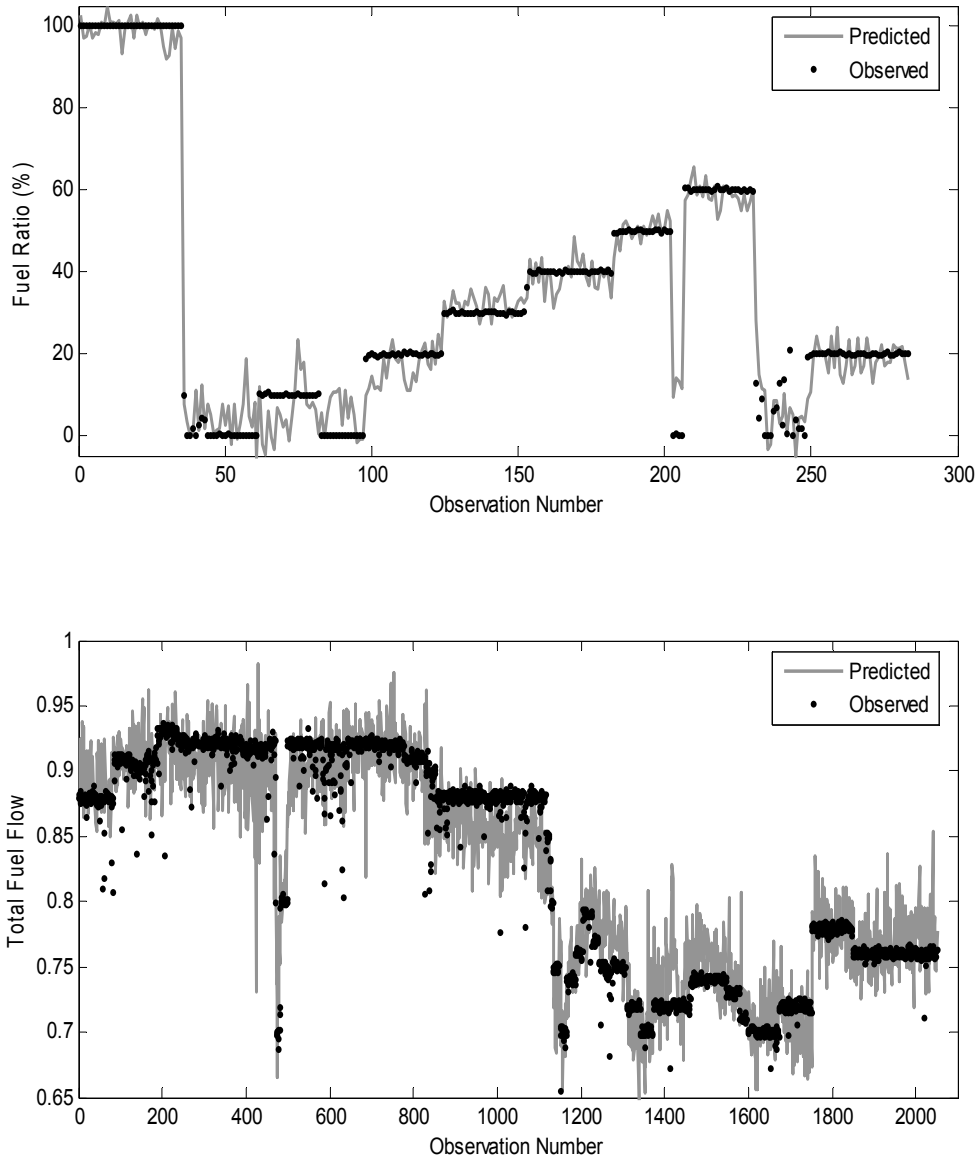


Figure 19: Prediction results of combustion variables using flame images (MIR models I and J). Black dots correspond to measured solids discharge temperatures and gray lines to model predictions.

To interpret these modeling results, two VIP contour plots are shown in Figure 20, one for model I (fuel ratio) and the other one for model J (total fuel flow rate). The VIP plot of model I show that two distinct regions are more important than others in predicting fuel ratio. These are located at the extreme left and right of the VIP plot. A mapping of these two regions onto a pre-selected flame image is given in Figure 21. This figure clearly shows that the most important image features for predicting fuel ratio are the coloration of flame, both at the boundary of the flame and at the burner tip (extreme left of the VIP and score histogram plots), as well as the projected luminosity of the flame in the area closer to the camera (extreme right of the VIP and score plots). Indeed, experienced operators at the QIT plant often have a very good idea of the fuel ratio just by looking at the overall coloration of the flame.

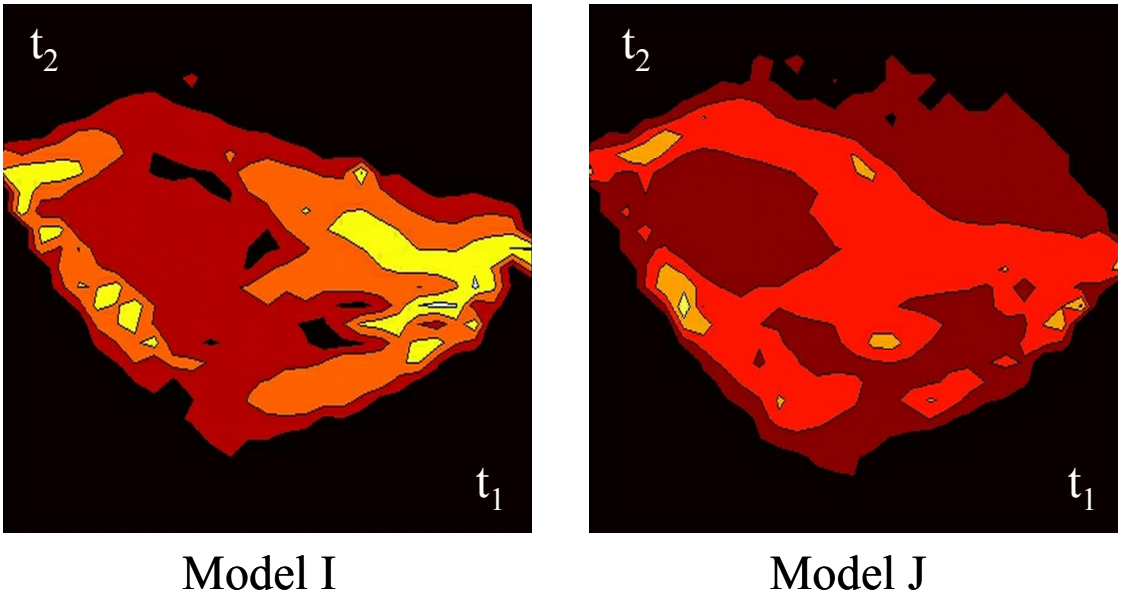


Figure 20: VIP contour plots for MIR models I and J.

On the other hand, the VIP plot of model J (total fuel flow rate) does not show any particular regions of the score density histograms being much more important than others. VIP values are rather quite homogeneous throughout the plot meaning that all bins colored

using light red (or dark red) have similar importance in predicting total fuel flow rate. This was expected since changes in the total fuel flow rate leads to important changes in heat released, which affect the color intensities of most pixels of flame images. Thus, changes in score values are similar across the score space. Total fuel flow rate therefore has an average effect on the flame color features.

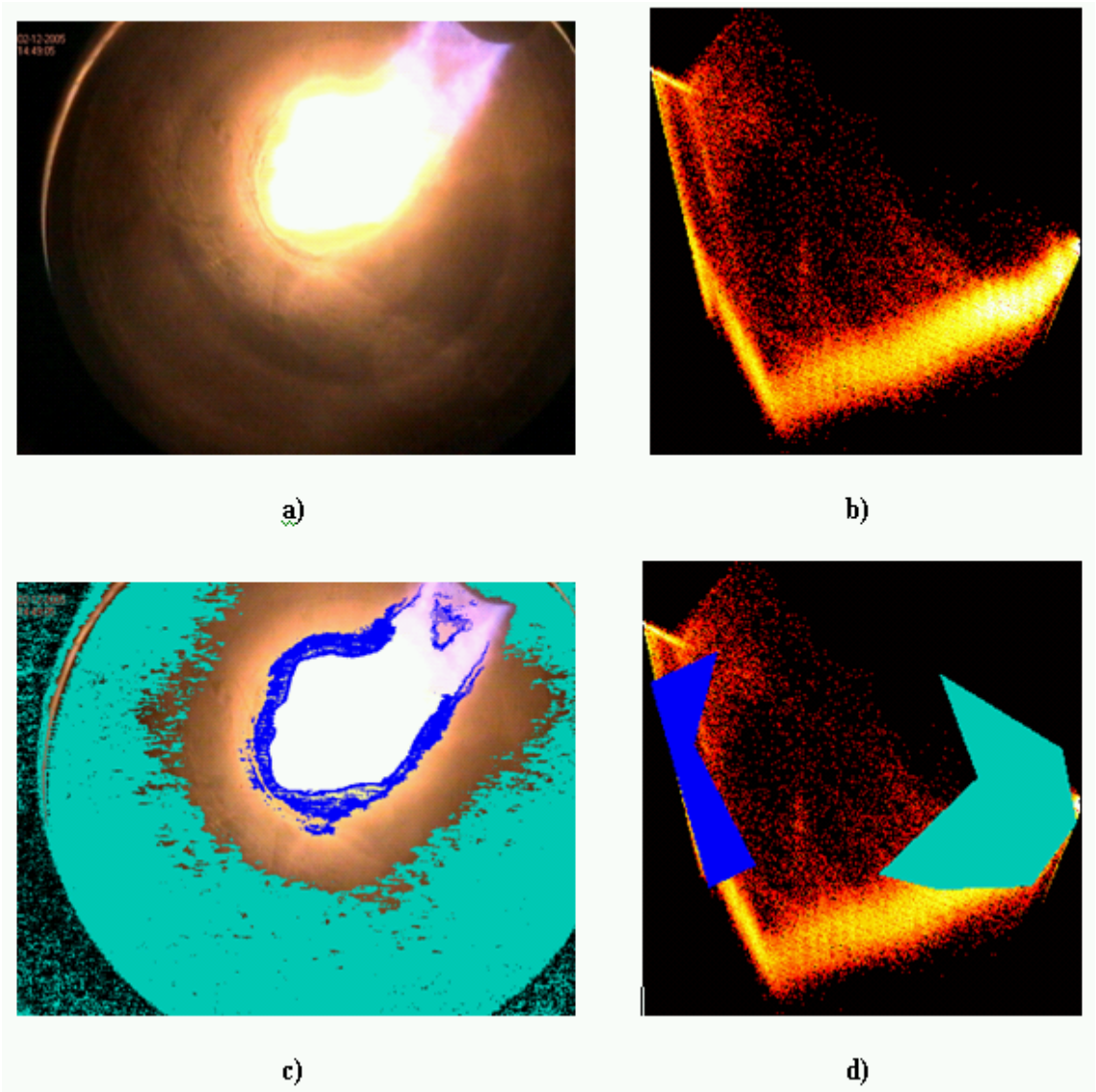


Figure 21: Mapping of the highest VIP regions of the t_1-t_2 scores on a selected flame image based on MIR model I used for fuel ratio predictions.

Finally, the choice of using the score values obtained from MIR model A to quantify flame appearance is motivated by 1) the fact that model A provides the best relationship between image color features and solids discharge temperature, and 2) the scores are multivariate measures of the color features of a flame image. Images having similar score values have a similar coloration, and hence the heat released by the combustion process should also be similar. To illustrate this, the score space (or feature space) of MIR model A is shown in Figure 21. That space has ten dimensions (see number of components of model A in Table 2), however only the first two dimensions (t_1-t_2 components) are shown for simplicity. In this t_1-t_2 score plot, each point corresponds to a multivariate summary of the color features of a flame image. Therefore, flame images having similar color features should fall in a similar region of that space. All images associated with a coded solids discharge temperature falling within the 0.85-0.86 range are shown using red dots whereas all other images are shown using blue dots. The specified range of discharge temperatures is a very narrow range around the actual set-point, according to the current QIT operation policies.

It is clear from Figure 22 that flame images associated with coded solids discharge temperature values within the range of 0.85-0.86 do not scatter over the entire score range, but cluster together in some region of the score plot. The role of the image-based combustion controller (inner controller in Figure 18) would be to find the total fuel flow rate to implement on the kiln in order to bring back or to maintain the flame appearance within the region delimited by the red dots (when a solids discharge temperature within 0.85-0.86 is desired). Controlling these score values at a specific set-point (or at a set-point region) should therefore help reduce heat transfer variability.

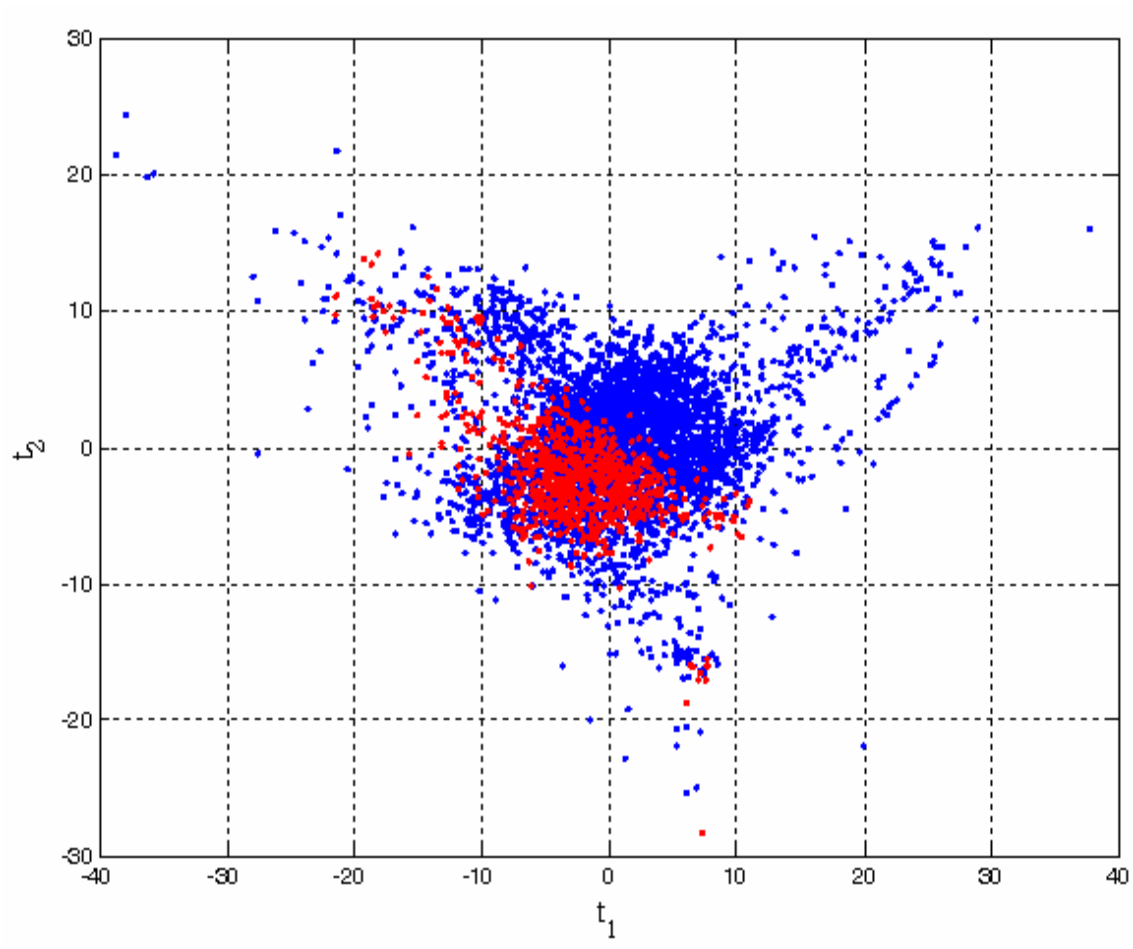


Figure 22: Feature space (t_1-t_2 score plot) of MIR model A. Red dots: flame images associated with a solids discharge temperature within the range of 0.85-0.86, blue dots: all other flame images.

Chapter 7 - Conclusions

Multivariate Image Analysis and Regression techniques were used in this work to investigate the information contained in digital RGB flame images collected from an industrial rotary kiln to help improve product quality and combustion control as well as for reducing fuel consumption. This study was motivated by the current lack of on-line sensor that simultaneously quantifies the variations in the combustion process and predicts the impact of these variations on final product quality. The proposed methodology was illustrated on an industrial rotary kiln used for ore roasting, where non-premixed combustion of multiple fuels occurs. Solids discharge temperature was used in this work as the final product quality indicator of the roasted ore. The proposed method could, however, be easily adapted to cope with quality and combustion control problems in other industries using rotary kilns.

Flame images were found to be very informative since they can be used not only to quantify variations in the combustion process through flame coloration (fast dynamics), but they also provide information on the thermal history of the solids within the kiln (slow dynamics) based on the coloration of the solids and the kiln's walls. It was shown that using a single flame image collected at a given time, one can explain as much as 80% of the variations in the solids discharge temperature measured at that time. This shows that combustion variability needs to be controlled in order to reduce discharge temperature fluctuations, and in turn, overheating and fuel consumption. The strong forecast ability of flame images was also demonstrated in this paper. A single flame image collected at time t respectively explains about 60% and 30% of the variations in discharge temperature collected at time t+20 and t+80 minutes in the future, which roughly correspond to one process dead-time and one mean residence time forecasts for the industrial rotary kiln used

in this study. Future work in this area will explore different methods to efficiently use a history of flame images to further improve these forecasts.

A discussion on how to use such a sensor to enhance manual or automatic feedback control of solids discharge temperature (quality) was also presented. The predictive ability of flame images could be used into a modified Smith Predictor scheme to help overcome the process dead-time limitations. The feasibility of a very innovative cascade control scheme using image-based combustion control was also discussed. Automatic control issues will be addressed in future work, but perspectives are already very promising.

A novel approach of this study was the usage of several types of models that have been plugged to MIA processed data in order to provide better results. This approach provides freedom of choice for researchers that would search to adapt this methodology to their particular industrial situations. By providing the freedom to use any type of modeling (linear, non-linear, neural networks, etc.) this methodology provides the researcher the possibility to use whichever model type fits best with his/her particular situation. Future work will better address the usage of Non-Linear as well as Neural Network models in conjuction with MIA. Also more work has to be done in comparing the prediction models based on process data only to the image based models. An investigation should be also made on how a better model can be constructed using both source of information, image and data.

Bibliography

Allen, M.G.; Butler, C.T.; Johnson, S.A.; Lo, E.Y.; Russo, F. An Imaging Neural Network Combustion Control System for Utility Boiler Applications. Combustion and Flames, 1993, 94, 205-214.

Alvarez, P.I.; Shene, C. Experimental determination of volumetric heat transfer coefficient in a rotary dryer. Drying Technol. 1994b, 12, 1605–1627.

Alvarez, P.I.; Shene, C. Experimental study of residence time in a direct rotary dryer. Drying Technol. 1994b, 12, 1629–1651.

Bertucco, L.; Fichera, A.; Nunnari, G.; Pagano, A. A Cellular Neural Networks Approach to Flame Image Analysis for Combustion Monitoring. Proceedings of the 6th IEEE International Workshop on Cellular Neural Networks and Their Applications, 2000, 455-459.

Bharati, M.H., "Multivariate Image Analysis and Regression for Industrial Process Monitoring and Product Quality Control", Ph.D. Thesis, McMaster University, Hamilton, Ontario, Canada, 2002.

Bharati, M.H.; MacGregor, J.F. Multivariate Image Analysis for Real-Time Process Monitoring and Control. Ind. Eng. Chem. Res., 1998, 37, 4715-4724.

Bharati, M.H.; MacGregor, J.F.; Tropper, W. Softwood Lumber Grading through On-line Multivariate Image Analysis Techniques. Ind. Eng. Chem. Res., 2003, 42, 5345-5353.

Cross, M.; Blot, P. Optimizing the Operation of Straight-Grate Iron-Ore Pellet Induration Systems Using Process Models. Metall. Mater. Trans.B, 1999, 30B, 803-813.

Didriksen, H. Model based predictive control of a rotary dryer, Chemical Engineering Journal, 86 2002 53–60;

Douglas, P.L., A.Kwade, P.L. Lee and S.K. Mallick, "Simulation of a rotary Dryer for Sugar Crystalline", Drying Technol. 11, 129-155 (1993).

Duchesne, C.; Thibault, J.; Bazin, C. Modeling of the solid transportation within an industrial rotary dryer. Ind. Eng. Chem. Res. 1996, 35, 2342–2354.

Duchesne, C., Thibault, J., Bazin, C. Dynamics and Assessment of Some Control Strategies of a Simulated Industrial Rotary Dryer, Drying Technology, 1997**,** 15(2), 477-510.

Esbensen, K.H.; Geladi, P. Strategy of Multivariate Image Analysis (MIA). Chemom. Intell. Lab. Syst., 1989, *7*, 67-86.

Eriksson, L., Johansson, E., Kettaneh-Wold, N., and Wold, S, Multi- and Megavariate Data Analysis: Principles and Applications; Umetrics Academy, Sweden, 2001.

Finnie, G.J.; Kruyt, N.P.; Ye, M.; Zeilstra, C.; Kuipers, J.A.M. Longitudinal and transverse mixing in rotary kilns: A discrete element method approach. Chemical Engineering Science 2005, 60, 4083 – 4091.

Geladi, P.; Grahn, H., Multivariate Image Analysis; Wiley: Chichester, UK, 1996.

Huang, Y.; Yan, Y.; Lu, G.; Reed, A. On-line Flicker Measurement of Gaseous Flames by Image Processing and Spectral Analysis. Meas. Sci. Technol., 1999, 10, 726-733.

J.J. del Coz Diaz, F. Rodriguez Mazon, P.J. Garcia Nieto, F.J. Suarez Dominguez, Design and finite element analysis of a wet cycle cement rotary kiln, November 2001

Järvensivu, M.; Saari, K.; Jämsä-Jounela, S.-L. Intelligent control system of an industrial lime kiln process, Control Engineering Practice, 2001, 9, 589-606.

Kamke, F.A.; Wilson, J.B. Computer simulation of a rotary dryer. AIChE J. 1986, 32, 263–268.

Keyvan, S. Diagnostics and Control of Natural Gas Fired Furnaces via Flame Image Analysis. Sensors and Automation FY03 Annual Review Meeting, Office of Industrial Technologies, Energy Efficiency and Renewable Energy, US Department of Energy, 2003 (http://www.oit.doe.gov/sens_cont/pdfs/annual_0603/keyvan_03.pdf).

Lied, T.T., and Esbensen, K.H., "Principles of MIR, Multivariate Image Regression I: Regression Typology and Representative Application Studies"; Chemometrics and Intelligent Laboratory Systems, Vol. 58, pp. 213-226, 2001.

Lied, T.T., Geladi, P., and Esbensen, K.H., "Multivariate Image Regression (MIR): Implementation of Image PLSR – First Forays"; Journal of Chemometrics, Vol 14, pp. 585-598, 2000.

Liu, J.J.; MacGregor, J.F. Modeling and Optimization of Product Appearance: Application to Injection-Molded Plastic Panels. Ind. Eng. Chem. Res., 2005, 44, 4687-4696.

Lu, G.; Yan, Y.; Huang, Y.; Reed, A. An Intelligent Vision System for Monitoring and Control of Combustion Flames. Measurement and Control, 1999, 32, 164-168.

Lu, G.; Yan, Y.; Ward, D.D. Advanced Monitoring, Characterisation and Evaluation of Gas Fired Flames in a Utility Boiler. Journal of the Institute of Energy, 2000, 73, 43-49.

Marias, F.; Roustan, H.; Pichat, A. Modelling of a rotary kiln for the pyrolysis of aluminium waste. Chemical Engineering Science 2005, 60, 4609 – 4622.

Martens, M.A.; Oliveira, L.S.; Franca, A.S.; Modeling and simulation of petroleum coke calcinations in rotary kilns. Fuel,. 2001, 80, 1611-1622.

Myklestad, O., "Heat and Mass Transfer in Rotary Dryers", Chem. Eng. Prog. Symp. Ser. 59. 129-137 (1963)

Najim, K. 1989, "Modeling and Learning Control of a Rotary Phosphate Dryer", Int. J. Systems Sci.,20, 1627-1636.

Otomo, T., Nakagawa, T. and Akaike, H., "Statistical Approach to Computer Control of Cement Rotary Kilns", Automatica, Vol. 8, pp. 35-48. Pergamon Press, 1972.

Robinson, J.W., 1989a, "The Delta T – A New Drying Model for Pulp and Paper", Proc. TAPPI Engineering Conference, Atlanta, Aug., 183-187.

Robinson, J.W., 1989b, "The Delta T – A New Concept in Dryer Control", Proc. American Association of Textile Chemists and Colorists, Philadelphia, Oct. 276-279.

Robinson, J.W., 1992, "Improve Dryer Control", Chem. Eng. Prog. December, 28-33.

Rovaglio, M.; Manca, D.; Biardi, G. Dynamic modeling of waste incineration plants with rotary kilns: Comparisons between experimental and simulation data. Chem. Eng. Sci., 1998, 53, 2727-2742.

Rumelhart, D.E., Hinton, G. and Williams, R. (1986). Learning internal representation by error propagation, Parallel Distributed Processing, 1, MIT Press, 318-364.

Shimoda, M.; Sugano, A.; Kimura, T.; Watanabe, Y.; Ishiyama, K. Prediction Methods of Unburnt Carbon for Coal Fired Utility Boiler Using Image Processing Technique of Combustion Flame. IEEE Transactions on Energy Conversion, 1990, 5, 640-645.

Tao, W.; Burkhardt, H. Vision-Guided Flame Control Using Fuzzy Logic and Neural Networks. Particle and Particle System Characterisation, 1995, 12, 87-94.

Tarca, A.L. (2004). "Neural networks in multiphase reactors data mining: feature selection, prior knowledge, and model design", PhD Dissertation at Laval University, Quebec.

Victor, J.; Costeira, J.; Tomé, J.; Sentieiro, J. A Computer Vision System for the Characterization and Classification of Flames in Glass Furnaces. Conference Record of the IEEE Industry Applications Society Annual Meeting, 1991, 1109-1117.

Wang, F.; Wang, X.J.; Ma, Z.Y.; Yan, J.H.; Chi, Y.; Wei, C.Y.; Ni, M.J.; Cen, K.F. The Research on the Estimation for the NOx Emissive Concentration of the Pulverized Coal by the Flame Image Processing Technique. Fuel, 2002, 81, 2113-2120.

Yamaguchi, T.; Grattan, K.T.V.; Uchiyama, H.; Yamada, T. A Practical Fiber Optic Air-Ratio Sensor Operating by Flame Color Detection. Rev. Sci. Instrum., 1997, 68, 197-202.

Yan, Y.; Lu, G.; Colechin, M. Monitoring and characterisation of pulverized coal flames using digital imaging techniques. Fuel, 2002, 81, 647-656.

Yu, H.; MacGregor, J.F. Monitoring Flames in an Industrial Boiler Using Multivariate Image Analysis. AIChE J. 2004, 50, 1474-1483.

Yu, H.; MacGregor, J.F. Multivariate image analysis and regression for prediction of coating content and distribution in the production of snack foods. Chemom. Intell. Lab. Syst., 2003, 67, 125-144.

Yu, H.; MacGregor, J.F.; Haarsma, G.; Bourg, W. Digital Imaging for Online Monitoring and Control of Industrial Snack Food Processes. Ind. Eng. Chem. Res., 2003, 42, 3036-3044.

Appendix 1 - Source Code

Script1.m

```
imageDirs = dir('images');
%calculating global variance covariance matrix for each directory
for i=1:length(imageDirs(:,1))
    if imageDirs(i).isdir == 1
        if strcmp(imageDirs(i).name,'.')
            continue;
        end
        if strcmp(imageDirs(i).name,'..')
            continue;
        end
        disp('Processing directory:');
        disp(imageDirs(i).name);
        %calculating the global variance-covariance matrix and inserting
        %image data in the imageSource table. The variance covariance
        %matrix is keept from one processing of an image directory to
        %another
        globalVarCov(strcat('/',imageDirs(i).name));
    end
end
for i=1:length(imageDirs(:,1))
    if imageDirs(i).isdir == 1
        if strcmp(imageDirs(i).name,'.')
            continue;
        end
        if strcmp(imageDirs(i).name,'..')
            continue;
        end
        disp('Processing directory:');
        disp(imageDirs(i).name);
        % calculating the histogram using the global variance-covariance
        % matrix
        processImages(strcat('/',imageDirs(i).name));
    end
end
```

processImages.m

```
function t = processImages(imageDir)
% this will process one directory at a time
tic
disp('Processing images and calculating histograms with the global
variance-covariance matrix...');
imageDir = strcat('images',imageDir);
imageFiles = dir(imageDir);
globalZ = zeros(3);
db = DatabaseBridge('varcovsum');
globalZ = load('global-varcov.var','-ASCII');
% obtaining the gloabal loadings
% Kernel PCA performs SVD on the globalZ matrix. the P matrix is the
% loadings matrix.
[P,E,PT] = svd(globalZ);
imageFiles = dir(imageDir);
imageNumber = 0;
% reading the absolute score limits
v = db.readAbsoluteT;
limits = [];
sizel = v.size-1;
for i = 0:sizel
    dbl = java.lang.Double(v.get(i));
    limits(i+1) = dbl.doubleValue;
end
save('limits.var','limits','-ASCII', '-TABS');
%starting the bin hostogram calculations
for i = 1:length(imageFiles(:,1))
    if imageFiles(i).isdir == 1
        continue
    end
    i;
    %disp('Processing Image Number:');
    %disp(imageNumber);
    %reading image to be precessed
    imageNumber = imageNumber+1;
    imageName = strcat(imageDir,'/',imageFiles(i).name);
    originalImage = imread(imageName ,'bmp');
    %reading the image creation date
    datevect = datevec(imageFiles(i).date);

db.setAquisitionDate(datevect(1),datevect(2),datevect(3),datevect(4),date
vect(5),datevect(6));
    %calculating the scoreplot
    [binMatrix] = binHistogramGlobalLoadings(originalImage,P,limits);
    %getting the matrix sliced according the desired resolution
    sliceVector = histogramSlicer(binMatrix,9);
    db.writeBins(sliceVector,imageFiles(i).name);
end
disp('The total elapsed time is:');
disp(toc);
```

globalVarCov.m

```
function t = globalVarCov(imageDir)
% this will process one directory at a time
tic
disp('Calculating the global variance-covariance matrix...');
imageDir = strcat('images',imageDir);
imageFiles = dir(imageDir);

imageNumber = 0;
globalZ = load('global-varcov.var','-ASCII');
if globalZ == 0
    globalZ = zeros(3);
end
db = DatabaseBridge('varcovsum');
for i = 1:length(imageFiles(:,1))
    if imageFiles(i).isdir == 1
        continue
    end
    %reading image to be precessed
    %disp('Writing Image Number:');
    %disp(imageNumber);
    imageName = imageFiles(i).name;
    imageLocation = strcat(imageDir,'/',imageName);
    originalImage = imread(imageLocation ,'bmp');
    IMG_WIDTH = size(originalImage, 1);
    IMG_HEIGHT = size(originalImage, 2);

    X = double(reshape(originalImage,
IMG_WIDTH*IMG_HEIGHT,ndims(originalImage)));
    % the variance covariance matrix
    Z = X'*X;
    % summing the variance-covariance matrix to the global
    % variance-covariance matrix
    globalZ = globalZ + Z;
    % inserting image data in the database the tmax and loading are not
    % calculated anymore for the histogram but a value from the database
is
    % used
    %reading the image creation date
    datevect = datevec(imageFiles(i).date);

db.setAquisitionDate(datevect(1),datevect(2),datevect(3),datevect(4),date
vect(5),datevect(6));
    %calculating the scoreplot
    [binMatrix,T,P] = binHistogram(originalImage);
    t1min = min(T(:,1));
    t2min = min(T(:,2));
    t3min = min(T(:,3));

    t1max = max(T(:,1));
    t2max = max(T(:,2));
    t3max = max(T(:,3));
```

```
    redMean = mean2(originalImage(:,:,1));
    greenMean = mean2(originalImage(:,:,2));
    blueMean = mean2(originalImage(:,:,3));
    std2Red = std2(originalImage(:,:,1));
    std2Green = std2(originalImage(:,:,2));
    std2Blue = std2(originalImage(:,:,3));
    p11 = P(1,1);
    p12 = P(1,2);
    p13 = P(1,3);
    p21 = P(2,1);
    p22 = P(2,2);
    p23 = P(2,3);
    p31 = P(3,1);
    p32 = P(3,2);
    p33 = P(3,3);

db.write(p11,p12,p13,p21,p22,p23,p31,p32,p33,t1min,t2min,t3min,t1max,t2ma
x,t3max,redMean,greenMean,blueMean,std2Red,std2Green,std2Blue,imageNumber
,imageName);
end
% saving the global variance-covariance matrix
save('global-varcov.var','globalZ','-ASCII', '-TABS');
disp('The total time for the global variance-covariance matrix and image
data is:');
disp(toc);
```

binHistogramGlobalLoading.m

```
function [binMatrix,T] = binHistogramAbs(originalImage, P, TLIMS)
% this function returns the scores based on the global loadings
RGB = originalImage;
IMG_WIDTH = size(RGB, 1);
IMG_HEIGHT = size(RGB, 2);
% X is a matrix that contiains the images as columns
X = double(reshape(RGB, IMG_WIDTH*IMG_HEIGHT,ndims(RGB)));
% Calulating the scores with the global loadgins
T = X*P;
% Each column of T is a score vector of X
t1 = T(:,1);
t2 = T(:,2);
t3 = T(:,3);
% Will create the 2-D Histogram by arranging the (t1,t2) combination in
% bins, where each bin is a square area covering a certain range of
(t1,t2)
% values. The number of (t1,t2) combination is calculated and it is
% associated to each bin. In fact in our case a bin equals to a point in
% the 256*256 matrix, each point having associated a number that
represents
% the number of (t1,t2) combinations falling into that point.
nbins = 255; % Number of bins between min & max of t1 & t2

% Determine the Increments by which t1 & t2 should be divided into based
on
% their respective ranges x-dimension is t1

%disp('x:');
xmin = TLIMS(1);
xmax = TLIMS(2);
dx = (xmax - xmin)/nbins;

% y-dimension is t2
%disp('x:');
ymin = TLIMS(3);
ymax = TLIMS(4);
dy = (ymax - ymin)/nbins;


% Create a matrix A containing zeros. A has dimensions of 256 x 256 since
% the 2D Histogram can be represented as a 256 x 256 image. It also will
% constitute the (t1,t2) bin matrix returned by this function, each point
% being a 2-D bin.

A = zeros(256,256);



% ****** Shift t1 and t2 to Begin at dx and dy, respectively ******
if (xmin < 0),
   t1shf = t1 + abs(xmin) + dx;
```

```
end;

if (ymin < 0),
   t2shf = t2 + abs(ymin) + dy;
end;

if (xmin > 0),
   t1shf = t1 - abs(xmin) + dx;
end;

if (ymin > 0),
   t2shf = t2 - abs(ymin) + dy;
end;

% ****************************************************************

colnum = zeros(size(t1)); % Vector representing t1 index in the (t1,t2)
set
rownum = zeros(size(t2)); % Vector representing t2 index in the (t1,t2)
set

% Following FOR loop goes through each (t1,t2) combination of the shifted
t1 & t2 vectors and
% assigns a particular (row,column) combination in A. The bin count of
that (row,column) combinaiton
% is then updated
for i = 1:length(t1), % Go through the shifted t1 & t2 vectors one
element at a time

    colnum(i) = round((t1shf(i))/dx);% Assign column number in A based on
current value of shifted t1 vector
    rownum(i) = round((t2shf(i))/dy);% Assign row number in A based on
current value of shifted t2 vector

    if (rownum(i) > 0) & (colnum(i) > 0)
       if (rownum(i) < 256) & (colnum(i) < 256)
             A(rownum(i),colnum(i)) = A(rownum(i),colnum(i)) + 1;%
Increase count by 1 in the assigned bin/pixel of A
       end
    end

end

% Now A should be a 256 x 256 matrix with bin counts at assigned
% (row,column) combinations. This will be returned by this function.

binMatrix = A;
```

histogramSlicer.m

```
function [sliceVector] = histogramSlicer(binMatrix,resolution)
% this function will take the matrix and slice it in piecese that have a
% "resolution" number of points and then addup the content of each piece
an
% create a new histogram named sliceVector resolution has to be smaller
% that the size of the matrix.

[n,m] = size(binMatrix);

if ( n ~= m)
    error('Matrix has to be square.');
    return;
end

if (resolution > n || resolution > m)
    error('Resolution has to be smaller that the size of the matrix.');
    return;
end


% for the moment only square matrixes will be taken into account

count = 1;
histogram(1)=0;
j1 = 0;
j2 = 0;
k1 = 0;
k2 = 0;

%calculating the number of cells that would result after sliceing the
%matrix wihtout the remainders
remainder = rem(n,resolution);
cellNr = fix(n/resolution);

notSquare = 0;
if remainder ~= 0
    notSquare =1;
end

for j=1:cellNr
    j;
    for k=1:cellNr
        k;
        j1 = (j * resolution)-resolution +1;
        j2 = j * resolution;

        k1 = (k * resolution)-resolution +1;
        k2 = k * resolution;

        bin = binMatrix(j1:j2,k1:k2);
        % the hits on this submatrix
        histogram(count) = sum(sum(bin));
```

```
        count = count + 1;

        % calculating the vertical (column) remainder
        if (notSquare == 1 && k == cellNr && j ~= cellNr )
            k1 = k2 +1;
            k2 = m;
            bin = binMatrix(j1:j2,k1:k2);
            % the hits on this submatrix
            histogram(count) = sum(sum(bin));
            count = count +1;
        end

        % calculating the horizontal (row) remainder
        if (notSquare == 1 && j == cellNr && k ~= cellNr)
            j1 = j2+1;
            j2 = m;
            bin = binMatrix(j1:j2,k1:k2);
            % the hits on this submatrix
            histogram(count) = sum(sum(bin));
            count = count +1;
        end
        %calulating the corner
        if (notSquare == 1 && k == cellNr && j == cellNr)
            %column remainder
            tempK1 = k1;
            tempK2 = k2;
            k1 = k2 +1;
            k2 = m;
            bin = binMatrix(j1:j2,k1:k2);
            % the hits on this submatrix
            histogram(count) = sum(sum(bin));
            count = count +1;

            %row remainder
            j1 = j2+1;
            j2 = m;
            bin = binMatrix(j1:j2,tempK1:tempK2);
            % the hits on this submatrix
            histogram(count) = sum(sum(bin));
            count = count +1;

            %corner remainder
            bin = binMatrix(j1:j2,k1:k2);
            % the hits on this submatrix
            histogram(count) = sum(sum(bin));
            count = count +1;
        end
    end
end


sliceVector = histogram;
```

binHistogram.m

```
function [binMatrix,T,P] = binHistogram(originalImage)
% this function will perform the SVD on a particular image and will
return
% the matrix that is a 2D bin histogram of the image, the T matrix that
has
% as columns the score vectors and the P matrix that has as columns the
% loading vectors

RGB = originalImage;

IMG_WIDTH = size(RGB, 1);
IMG_HEIGHT = size(RGB, 2);

% X is a matrix that contiains the images as columns
X = double(reshape(RGB, IMG_WIDTH*IMG_HEIGHT,ndims(RGB)));

% Calculates the variance-covariance matrix of X (the Kernel matrix) wich
is only 3 x 3 in dimension
Z = X'*X;

% Kernel PCA performs SVD on the Z matrix. Here P contains is the
eigenvectors of Z as columns, E contains
% the eigenvalues (on the diagonal) of Z, and PT contains the
eigenvectors as rows.

[P,E,PT] = svd(Z);

% Trace of E (i.e. sum of all eigenvalues) is the Total sum of squares in
X
% One can determine the amount of variation explained by each Principal
Component through dividing
% individual eigenvalues by the sum of squares.

% The eigenvectors of Z are also the loading vectors of X. So we can find
% the score vectors by multiplying P with X
T = X*P;

% Each column of T is a score vector of X
t1 = T(:,1);
t2 = T(:,2);
t3 = T(:,3);

% Will create the 2-D Histogram by arranging the (t1,t2) combination in
% bins, where each bin is a square area covering a certain range of
(t1,t2)
% values. The number of (t1,t2) combination is calculated and it is
% associated to each bin. In fact in our case a bin equals to a point in
% the 256*256 matrix, each point having associated a number that
represents
% the number of (t1,t2) combinations falling into that point.

nbins = 255; % Number of bins between min & max of t1 & t2
```

```
% Determine the Increments by which t1 & t2 should be divided into based
on
% their respective ranges x-dimension is t1

%disp('x:');
xmin = min(t1);
xmax = max(t1);
dx = (xmax - xmin)/nbins;

% y-dimension is t2
%disp('x:');
ymin = min(t2);
ymax = max(t2);
dy = (ymax - ymin)/nbins;


% Create a matrix A containing zeros. A has dimensions of 256 x 256 since
% the 2D Histogram can be represented as a 256 x 256 image. It also will
% constitute the (t1,t2) bin matrix returned by this function, each point
% being a 2-D bin.

A = zeros(256,256);



% ****** Shift t1 and t2 to Begin at dx and dy, respectively ******
if (xmin < 0),
   t1shf = t1 + abs(xmin) + dx;
end;

if (ymin < 0),
   t2shf = t2 + abs(ymin) + dy;
end;



if (xmin > 0),
   t1shf = t1 - abs(xmin) + dx;
end;

if (ymin > 0),
   t2shf = t2 - abs(ymin) + dy;
end;

% ****************************************************************


colnum = zeros(size(t1)); % Vector representing t1 index in the (t1,t2)
set
rownum = zeros(size(t2)); % Vector representing t2 index in the (t1,t2)
set



% Following FOR loop goes through each (t1,t2) combination of the shifted
t1 & t2 vectors and
```

```
% assigns a particular (row,column) combination in A. The bin count of
that (row,column) combinaiton
% is then updated
for i = 1:length(t1), % Go through the shifted t1 & t2 vectors one
element at a time

    colnum(i) = round((t1shf(i))/dx);% Assign column number in A based on
current value of shifted t1 vector
    rownum(i) = round((t2shf(i))/dy);% Assign row number in A based on
current value of shifted t2 vector

    if (rownum(i) > 0) & (colnum(i) > 0)
       if (rownum(i) < 256) & (colnum(i) < 256)
             A(rownum(i),colnum(i)) = A(rownum(i),colnum(i)) + 1;%
Increase count by 1 in the assigned bin/pixel of A
       end
    end

end

% Now A should be a 256 x 256 matrix with bin counts at assigned
% (row,column) combinations. This will be returned by this function.

binMatrix = A;
```

export.sql

```
SELECT

imageSource.imageName,
processData.DateTime AS AcquisitionTime,
processData.k4_tit5_noncorrige AS dt,
processData.dt5min AS dt5min,
processData.dt10min AS dt10min,
processData.dt15min AS dt15min,
processData.dt20min AS dt20min,
prediction.dt AS predictedDT,
prediction.dt5 AS predictedDT5,
prediction.dt10 AS predictedDT10,
prediction.dt15 AS predictedDT15,
prediction.dt20 AS predictedDT20

FROM
            dbo.processData LEFT OUTER JOIN
            dbo.imageSource ON dbo.processData.id =
dbo.imageSource.processDataID
            INNER JOIN
            dbo.prediction ON dbo.imageSource.imageName =
dbo.prediction.imageName


WHERE

processData.DateTime > '2004-11-20 0:0:0' AND
processData.DateTime < '2004-11-24 0:0:0' AND

CONVERT(char(26), processData.DateTime, 109) LIKE '%00:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%05:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%10:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%15:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%20:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%25:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%30:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%35:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%40:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%45:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%50:00:0%' OR
CONVERT(char(26), processData.DateTime, 109) LIKE '%55:00:0%'



Order BY processData.DateTime
```

DatabaseBridge.java

```
import java.io.*;
import java.sql.*;
import java.util.*;

// version june 19 2005. updated the timeshift method to shift up to 80
min.
// version sept 19 2005, updated to shift back 80 min.

public class DatabaseBridge {

    Vector pool = null;

    private int year,month,day,hour,minute,second;

    PrintWriter writer = null;

    String time = null;
    String processDataInsert = null;
    String processDataIDQuery = null;
    String insertHistSQL = null;
    int processDataID = 0;

    static GregorianCalendar startTimeshiftDate = null;
    static GregorianCalendar endTimeshiftDate = null;

    public DatabaseBridge(){
        this("varcovsum");
        System.out.println("The default database is varcovsum! ");
    }

    public DatabaseBridge(String dbName) {

        pool = new Vector();

        try {
            writer = new PrintWriter(new
FileWriter("error.txt",true),true);
            writer.println((new java.util.Date()).toString());
            //DriverManager.setLogWriter( new PrintWriter( System.out )
);
        }
        catch (IOException io) {io.printStackTrace(System.out);}

        try {


//Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
            Class.forName("com.inet.tds.TdsDriver");
            //Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
            //String url = "jdbc:microsoft:sqlserver://localhost:1433";
            String url = "jdbc:inetdae7:localhost:1433?database="+dbName;
            //String url = "jdbc:odbc:shift";
```

```
            for (int i=0;i<10;i++) {
                Connection conn = DriverManager.getConnection(url, "sa",
"sa");
                pool.add(conn);
            }
        }
        catch(Exception e) {
            e.printStackTrace(System.out);
        }


    }

    public static void main(String[] args) {


        DatabaseBridge m = new DatabaseBridge("varcovsum");
        //m.generateHistogramTable(256);
        //m.setAquisitionDate(2004, 04, 05, 15, 01, 56);

        m.setStartTimeshiftDate(2004,04,04,0,0,0);
        m.setEndTimeshiftDate(2005,04,04,0,0,0);

//        System.out.println("StartDate: " +
m.getDate(startTimeshiftDate));
//        System.out.println("EndDate: " + m.getDate(endTimeshiftDate));

        m.timeShift80Back();

       //m.writePredictions("test",170,12121,12121,121245,2342);

    }

    public void setAquisitionDate(int y, int m, int d, int h, int mn, int
s) {

        year = y;
        month = m;
      day = d;
        hour = h;
        minute = mn;
        second = s;

        //computing the right date for the process data.
        int sec = second;


        //the DayLightSavingTime bug
        //TODO have to implement this with GregorianCaledar

        if ( (month == 6 && year ==2004) || (month == 7 && year ==2004)
|| (month == 8 && year ==2004)) {

            hour = hour+1;
            if (hour > 24) {
                day = day+1;
```

```
                hour = hour-24;
            }
            //TODO correction if the end of month
        }


        //the normal case aquisition
        if (sec <= 4)
            second=0;
        else if (5 <= sec && sec <=14)
            second=10;
        else if (15 <= sec && sec <=24)
            second=20;
        else if (25 <= sec && sec <=34)
            second=30;
        else if (35 <= sec && sec <=44)
            second=40;
        else if (45 <= sec && sec <=54)
            second=50;
        else if (55 <= sec) {
            second = 0;
            minute +=1;
            if (minute == 60) {
                minute = 0;
                hour +=1;
                if (hour == 24){
                    hour=0;
                    day +=1;
                    //in case we will have data that will come in right
at the end of the month we have to be very carefully.
                }
            }
        }
        time = year+"-"+month+"-"+day+" "+hour+":"+minute+":"+second;

    }

    //before calling this method the setAquisitionDate should be called
    public void write(double p11, double p12, double p13, double p21,
double p22, double p23, double p31, double p32, double p33, double t1min,
double t2min, double t3min, double t1max, double t2max, double t3max,
double redMean, double greenMean, double blueMean,  double std2Red,
double std2Green, double std2Blue, int imageNumber,String imageName){

        try {

            Connection con = getConnection();
            //reading the processdata ID and inserting the process Data
from the import table into the processData table

            processDataIDQuery = "select id from processData where
DateTime = '" + time + "'";

            PreparedStatement pst1 =
con.prepareStatement(processDataIDQuery);
            ResultSet rs1 =  pst1.executeQuery();
            rs1.next();
```

```
            processDataID = rs1.getInt(1);

            //inserting the data in the source image table
            PreparedStatement pstmt = con.prepareStatement("insert into
imageSource
(p11,p12,p13,p21,p22,p23,p31,p32,p33,t1min,t2min,t3min,t1max,t2max,t3max,
redMean,greenMean,blueMean,std2Red,std2Green,std2Blue,processDataID,image
Name) values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");


            pstmt.setDouble(1,p11);
            pstmt.setDouble(2,p12);
            pstmt.setDouble(3,p13);
            pstmt.setDouble(4,p21);
            pstmt.setDouble(5,p22);
            pstmt.setDouble(6,p23);
            pstmt.setDouble(7,p31);
            pstmt.setDouble(8,p32);
            pstmt.setDouble(9,p33);


            pstmt.setDouble(10,t1min);
            pstmt.setDouble(11,t2min);
            pstmt.setDouble(12,t3min);
            pstmt.setDouble(13,t1max);
            pstmt.setDouble(14,t2max);
            pstmt.setDouble(15,t3max);
            pstmt.setDouble(16,redMean);
            pstmt.setDouble(17,greenMean);
            pstmt.setDouble(18,blueMean);
            pstmt.setDouble(19,std2Red);
            pstmt.setDouble(20,std2Green);
            pstmt.setDouble(21,std2Blue);
            pstmt.setDouble(22,processDataID);
            pstmt.setString(23,imageName);


            pstmt.execute();
            pstmt.close();
            pst1.close();

        }
        catch (SQLException e) {
            e.printStackTrace(writer);
            writer.println("Time: " + time);
            writer.println("DataInsert: " + processDataInsert);
            writer.println("DataIDQuery: " + processDataIDQuery);
            writer.println("DataID: " + processDataID);
            writer.flush();
        }

    }

    private Connection getConnection (){

        Connection conn = null;
```

```
        for (int i=0;i<pool.size();i++){
            conn = (Connection)pool.get(i);
            try {
                conn.getMetaData();
                break;
            }
            catch(Exception e) {
                writer.println("The connection number is:"+i);
                pool.removeElementAt(i);
                e.printStackTrace(writer);
            }
        }
        return conn;

    }

    public void generateHistogramTable(int histogramAccuracy){

        try {

            Connection conn = getConnection();
            Statement stmt = conn.createStatement();

            stmt.execute("CREATE TABLE histogram ( ID INT
IDENTITY(1,1))");
            stmt.execute("ALTER TABLE histogram ADD hist0 INT NULL");

            for (int i=0;i<histogramAccuracy;i++){
                stmt.execute("ALTER TABLE histogram ADD hist"+i+" INT
NULL");
            }
            stmt.close();
            conn.close();
        }
        catch (SQLException e) {e.printStackTrace(System.out);}
    }

    public Vector readAbsoluteT(){

        Vector v = new Vector();

        try {

            Connection conn = getConnection();
            Statement stmt = conn.createStatement();

            ResultSet rs = stmt.executeQuery("Select MIN(t1min),
MAX(t1max) ,  MIN(t2min) , MAX(t2max) ,  MIN(t3min) , MAX(t3max) from
imageSource");
            rs.next();

            v.add(new Double(rs.getDouble(1)));
            v.add(new Double(rs.getDouble(2)));
            v.add(new Double(rs.getDouble(3)));
            v.add(new Double(rs.getDouble(4)));
            v.add(new Double(rs.getDouble(5)));
            v.add(new Double(rs.getDouble(6)));
```

```
            stmt.close();

        }
        catch (SQLException e) {e.printStackTrace(writer);}
        return v;
    }

    public void writeBins(int[] histogram,String imgName){

        imgName = imgName.trim();
        int imageSourceID = 0;
        try {

            Connection con = getConnection();

            Statement st = con.createStatement();
            ResultSet rslt = st.executeQuery("select [id] from
imageSource where imageName like '%"+ imgName +"%'");
            rslt.next();
            imageSourceID = rslt.getInt(1);

            Statement st1 = null;

            insertHistSQL = "insert into histogram ( imageSourceID, ";
            String values = " values( " + imageSourceID + " , ";
            for (int i=0; i<histogram.length-1;i++){
                insertHistSQL += ("hist" + i + ", ");
                values += (histogram[i] + ", ");
            }
            insertHistSQL += ("hist" + (histogram.length-1) + " ) ");
            values += (histogram[histogram.length-1] + " ) ");

            insertHistSQL += values;

            st1 = con.createStatement();
            st1.executeUpdate(insertHistSQL);

            st.close();
            st1.close();

        }
        catch (SQLException e) {
            e.printStackTrace(writer);
            writer.println("ImageName: " + imgName);
            writer.println("Histogram Insert: " + insertHistSQL);
            writer.println("ImageSourceID: " + imageSourceID);
            writer.flush();
        }

    }
    public void setStartTimeshiftDate(int y, int m, int d, int h, int
min, int s){

        startTimeshiftDate = new GregorianCalendar();
        startTimeshiftDate.set(Calendar.YEAR,y);
        startTimeshiftDate.set(Calendar.MONTH,m);
```

```
        startTimeshiftDate.set(Calendar.DAY_OF_MONTH,d);
        startTimeshiftDate.set(Calendar.HOUR,h);
        startTimeshiftDate.set(Calendar.MINUTE,min);
        startTimeshiftDate.set(Calendar.SECOND,s);


    }
    public void setEndTimeshiftDate(int yn, int mn, int dn, int hn, int
minn, int sn){

        endTimeshiftDate = new GregorianCalendar();
        endTimeshiftDate.set(Calendar.YEAR,yn);
        endTimeshiftDate.set(Calendar.MONTH,mn);
        endTimeshiftDate.set(Calendar.DAY_OF_MONTH,dn);
        endTimeshiftDate.set(Calendar.HOUR,hn);
        endTimeshiftDate.set(Calendar.MINUTE,minn);
        endTimeshiftDate.set(Calendar.SECOND,sn);

    }

    public void timeShift() {

        System.out.println("Starting the timeshift...");

        String step = "";
        String query = "";
        String currentTime = "";

        long elapsedTime = System.currentTimeMillis();

        try{


            currentTime = "'" +startTimeshiftDate.get(Calendar.YEAR) + "-
" + (startTimeshiftDate.get(Calendar.MONTH)+1) + "-" +
startTimeshiftDate.get(Calendar.DAY_OF_MONTH) + " 0:0:0'";
            GregorianCalendar currentTimeshiftDate = new
GregorianCalendar();

currentTimeshiftDate.setTimeInMillis(startTimeshiftDate.getTimeInMillis()
);
            GregorianCalendar currentTimeshiftDate20 = new
GregorianCalendar();
            GregorianCalendar timeshift5min = new GregorianCalendar();
            GregorianCalendar timeshift10min = new GregorianCalendar();
            GregorianCalendar timeshift15min =new GregorianCalendar();
            GregorianCalendar timeshift20min =new GregorianCalendar();

            Connection con = getConnection();

            Statement st = con.createStatement();
            query = "select top 1 id from processData where DateTime >" +
currentTime + " order by DateTime";
            step = "start";
            ResultSet rs = st.executeQuery(query);
            //System.out.println(currentTime);
```

```
            rs.next();

            int processDataID = rs.getInt(1);
            //System.out.println(processDataID);

            //System.out.println("currentTimeShiftDate: " +
getDate(currentTimeshiftDate));

            Statement st1 = con.createStatement();
            query = "select DateTime from processData where id =" +
processDataID;
            step = "select first record";
            ResultSet rs1 = st1.executeQuery(query);
            rs1.next();

            Timestamp ts = rs1.getTimestamp(1);
            //System.out.println(ts.toString());
            currentTimeshiftDate.setTimeInMillis(ts.getTime());
            //System.out.println("The current date after initialization:
" + getDate(currentTimeshiftDate));


currentTimeshiftDate20.setTimeInMillis(currentTimeshiftDate.getTimeInMill
is());
            currentTimeshiftDate20.add(Calendar.MINUTE,20);

            //System.out.println("The current currentTimeShiftDate15
after initialization: " + getDate(currentTimeshiftDate15));
            //System.out.println("The endTimeshiftDate before while: " +
getDate(endTimeshiftDate));

            while (before(currentTimeshiftDate20,endTimeshiftDate)) {

                int dt5min = 0;
                int dt10min = 0;
                int dt15min = 0;
                int dt20min = 0;

                java.sql.Timestamp shiftDate = null;


timeshift5min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis());
                timeshift5min.add(Calendar.MINUTE, 5);


timeshift10min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis());
                timeshift10min.add(Calendar.MINUTE, 10);


timeshift15min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis());
                timeshift15min.add(Calendar.MINUTE, 15);


timeshift20min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis());
                timeshift20min.add(Calendar.MINUTE, 20);
```

```
//                System.out.println("Shift 5 min:
"+getDate(timeshift5min));
//                System.out.println("Shift 10 min:
"+getDate(timeshift10min));
//                System.out.println("Shift 15 min:
"+getDate(timeshift15min));

                try {

                    //discharge temp for t+5 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t+5";
                    PreparedStatement pst1 = con.prepareStatement(query);
                    shiftDate = new
java.sql.Timestamp(timeshift5min.getTimeInMillis());
                    //System.out.println("Shift5 "+shiftDate.toString());
                    //System.out.println(query);
                    pst1.setTimestamp(1,shiftDate);
                currentTime = shiftDate.toString();
                    ResultSet rs2 = pst1.executeQuery();

                    rs2.next();

                    dt5min = rs2.getInt(1);
                    //System.out.println("The 5min dt: "+dt5min);

                    //discharge temp for t+10 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t+10";
                    PreparedStatement pst2 = con.prepareStatement(query);

                    shiftDate = new
java.sql.Timestamp(timeshift10min.getTimeInMillis());
                    //System.out.println("Shift10
"+shiftDate.toString());
                    currentTime = shiftDate.toString();
                    pst2.setTimestamp(1,shiftDate);
                    ResultSet rs3 = pst2.executeQuery();

                    rs3.next();

                    dt10min = rs3.getInt(1);
                    //System.out.println("The 10min dt: "+dt10min);


                    //discharge temp for t+15 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t+15";
                    PreparedStatement pst3 = con.prepareStatement(query);

                    shiftDate = new
java.sql.Timestamp(timeshift15min.getTimeInMillis());
                    //System.out.println("Shift15
"+shiftDate.toString());
```

```
                    currentTime = shiftDate.toString();
                    pst3.setTimestamp(1,shiftDate);
                    ResultSet rs4 = pst3.executeQuery();

                    rs4.next();

                    dt15min = rs4.getInt(1);
                    //System.out.println("The 15min dt: "+dt15min);

                    //discharge temp for t+20 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t+20";
                    PreparedStatement pst4 = con.prepareStatement(query);

                    shiftDate = new
java.sql.Timestamp(timeshift20min.getTimeInMillis());
                    //System.out.println("Shift20
"+shiftDate.toString());
                    currentTime = shiftDate.toString();
                    pst4.setTimestamp(1,shiftDate);
                    ResultSet rs5 = pst4.executeQuery();

                    rs5.next();

                    dt20min = rs5.getInt(1);
                    //System.out.println("The 20min dt: "+dt20min);

                }


                catch (SQLException e) {

                    //writer.println("Elapsed time per period: " +
((System.currentTimeMillis()-elapsedTime)/1000));
                    //System.out.println("Elapsed time per period: " +
((System.currentTimeMillis()-elapsedTime)/1000));
                    //writer.println("Timeshift out of range!; Time=
"+currentTime);
                    dt5min=0;
                    dt10min=0;
                    dt15min=0;
                    dt20min=0;

                    //e.printStackTrace(writer);
                    //the timeshift trailings - end of range
                    writer.println("**********************");
                    writer.println(query);
                    writer.println(currentTime);
                    writer.println(step);
                    writer.println("**********************");
                    writer.flush();

                }
```

```
                //inserting the shifted discharge temperatures
                query = "update processData set dt5min=?, dt10min=?,
dt15min=?, dt20min=? where DateTime = ?";
                step = "updating the processData info";
                PreparedStatement pst5 = con.prepareStatement(query);

                shiftDate = new
java.sql.Timestamp(currentTimeshiftDate.getTimeInMillis());
//                System.out.println("Update: "+query);
//                System.out.println("dt5min: "+dt5min);
//                System.out.println("dt10min: "+dt10min);
//                System.out.println("dt15min: "+dt15min);
//                System.out.println("dt20min: "+dt20min);
//                System.out.println("DateTime "+shiftDate.toString());
                currentTime = shiftDate.toString();

                pst5.setInt(1,dt5min);
                pst5.setInt(2,dt10min);
                pst5.setInt(3,dt15min);
                pst5.setInt(4,dt20min);
                pst5.setTimestamp(5,shiftDate);
                int out = pst5.executeUpdate();
                //System.out.println("Lines afected: " +out);

                //getting the next datetime available
                query = "select TOP 1 DateTime from processData where
DateTime > ? order by DateTime";
                PreparedStatement pst6 = con.prepareStatement(query);
                step = "End of cycle";
                shiftDate = new
java.sql.Timestamp(currentTimeshiftDate.getTimeInMillis());
                currentTime = shiftDate.toString();
                pst6.setTimestamp(1,new
java.sql.Timestamp(currentTimeshiftDate.getTimeInMillis()));
                ResultSet rs6 = pst6.executeQuery();
                rs6.next();

currentTimeshiftDate.setTimeInMillis((rs6.getTimestamp(1)).getTime());

currentTimeshiftDate20.setTimeInMillis(currentTimeshiftDate.getTimeInMill
is());
                currentTimeshiftDate20.add(Calendar.SECOND,1200);

                //System.out.println("Condition at the end: " +
before(currentTimeshiftDate15,endTimeshiftDate));

            }

            System.out.println("Finished the timeshift!");
            System.out.println("Total elapsed time: " +
((System.currentTimeMillis()-elapsedTime)/1000));

        }
        catch (SQLException e) {
```

```
            e.printStackTrace(writer);
            writer.println(query);
            writer.println(currentTime);
            writer.println(step);
            writer.flush();
        }

    }

    private String getDate(GregorianCalendar c) {
        String str = c.get(Calendar.YEAR) + "-" + c.get(Calendar.MONTH) +
"-" + c.get(Calendar.DAY_OF_MONTH) + " " + c.get(Calendar.HOUR) + ":" +
c.get(Calendar.MINUTE) + ":" + c.get(Calendar.SECOND);
        return str;
    }

    private boolean before(Calendar before, Calendar after) {
        if (before.get(Calendar.YEAR) < after.get(Calendar.YEAR))
            return true;
        if (before.get(Calendar.YEAR) > after.get(Calendar.YEAR))
            return false;
        if (before.get(Calendar.MONTH) < after.get(Calendar.MONTH))
            return true;
        if (before.get(Calendar.MONTH) > after.get(Calendar.MONTH))
            return false;
        if (before.get(Calendar.DAY_OF_MONTH) <
after.get(Calendar.DAY_OF_MONTH))
            return true;
        if (before.get(Calendar.DAY_OF_MONTH) >
after.get(Calendar.DAY_OF_MONTH))
            return false;
        if (before.get(Calendar.HOUR) < after.get(Calendar.HOUR))
            return true;
        if (before.get(Calendar.HOUR) > after.get(Calendar.HOUR))
            return false;
        if (before.get(Calendar.MINUTE) < after.get(Calendar.MINUTE))
            return true;
        if (before.get(Calendar.MINUTE) > after.get(Calendar.MINUTE))
            return false;
        if (before.get(Calendar.SECOND) < after.get(Calendar.SECOND))
            return true;
        return false;

    }

    // this method will write the predictions into the database
prediction table
    public void writePredictions(String imageName, double dt, double dt5,
double dt10, double dt15, double dt20, double dt40, double dt60, double
dt80){

        try {

            Connection con = getConnection();

            //inserting the data in the source image table
```

```
            processDataInsert = "insert into prediction
(imageName,dt,dt5,dt10,dt15,dt20,dt40,dt60,dt80)
values(?,?,?,?,?,?,?,?,?)";
            PreparedStatement pstmt =
con.prepareStatement(processDataInsert);


            pstmt.setString(1,imageName);
            pstmt.setDouble(2,dt);
            pstmt.setDouble(3,dt5);
            pstmt.setDouble(4,dt10);
            pstmt.setDouble(5,dt15);
            pstmt.setDouble(6,dt20);
            pstmt.setDouble(7,dt40);
            pstmt.setDouble(8,dt60);
            pstmt.setDouble(9,dt80);

            pstmt.execute();
            pstmt.close();

        }
        catch (SQLException e) {
            e.printStackTrace(writer);
            writer.println("DataInsert: " + processDataInsert);
            writer.flush();
        }

    }
    //this will timeshift with 40,60, respectively 80 min into the
processData table from the database
    public void timeShift80() {

        System.out.println("Starting the timeshift 80...");

        String step = "";
        String query = "";
        String currentTime = "";

        long elapsedTime = System.currentTimeMillis();

        try{


            currentTime = "'" +startTimeshiftDate.get(Calendar.YEAR) + "-
" + (startTimeshiftDate.get(Calendar.MONTH)+1) + "-" +
startTimeshiftDate.get(Calendar.DAY_OF_MONTH) + " 0:0:0'";
            GregorianCalendar currentTimeshiftDate = new
GregorianCalendar();

currentTimeshiftDate.setTimeInMillis(startTimeshiftDate.getTimeInMillis()
);
            GregorianCalendar currentTimeshiftDate80 = new
GregorianCalendar();
            GregorianCalendar timeshift40min = new GregorianCalendar();
            GregorianCalendar timeshift60min = new GregorianCalendar();
            GregorianCalendar timeshift80min =new GregorianCalendar();
```

```
            Connection con = getConnection();

            Statement st = con.createStatement();
            query = "select top 1 id from processData where DateTime >" +
currentTime + " order by DateTime";
            step = "start";
            ResultSet rs = st.executeQuery(query);
            //System.out.println(currentTime);

            rs.next();

            int processDataID = rs.getInt(1);
            //System.out.println(processDataID);

            //System.out.println("currentTimeShiftDate: " +
getDate(currentTimeshiftDate));

            Statement st1 = con.createStatement();
            query = "select DateTime from processData where id =" +
processDataID;
            step = "select first record";
            ResultSet rs1 = st1.executeQuery(query);
            rs1.next();

            Timestamp ts = rs1.getTimestamp(1);
            //System.out.println(ts.toString());
            currentTimeshiftDate.setTimeInMillis(ts.getTime());
            //System.out.println("The current date after initialization:
" + getDate(currentTimeshiftDate));


currentTimeshiftDate80.setTimeInMillis(currentTimeshiftDate.getTimeInMill
is());
            currentTimeshiftDate80.add(Calendar.MINUTE,80);

            //System.out.println("The current currentTimeShiftDate15
after initialization: " + getDate(currentTimeshiftDate15));
            //System.out.println("The endTimeshiftDate before while: " +
getDate(endTimeshiftDate));

            while (before(currentTimeshiftDate80,endTimeshiftDate)) {

                int dt40min = 0;
                int dt60min = 0;
                int dt80min = 0;

                java.sql.Timestamp shiftDate = null;


timeshift40min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis());
                timeshift40min.add(Calendar.MINUTE, 40);


timeshift60min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis());
                timeshift60min.add(Calendar.MINUTE, 60);
```

```
timeshift80min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis());
                timeshift80min.add(Calendar.MINUTE, 80);


//                System.out.println("Shift 40 min:
"+getDate(timeshift5min));
//                System.out.println("Shift 60 min:
"+getDate(timeshift10min));
//                System.out.println("Shift 80 min:
"+getDate(timeshift15min));

                try {

                    //discharge temp for t+40 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t+40";
                    PreparedStatement pst1 = con.prepareStatement(query);
                    shiftDate = new
java.sql.Timestamp(timeshift40min.getTimeInMillis());
                    //System.out.println("Shift40
"+shiftDate.toString());
                    //System.out.println(query);
                    pst1.setTimestamp(1,shiftDate);
                currentTime = shiftDate.toString();
                    ResultSet rs2 = pst1.executeQuery();

                    rs2.next();

                    dt40min = rs2.getInt(1);
                    //System.out.println("The 40 min dt: "+dt40min);

                    //discharge temp for t+60 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t+60";
                    PreparedStatement pst2 = con.prepareStatement(query);

                    shiftDate = new
java.sql.Timestamp(timeshift60min.getTimeInMillis());
                    //System.out.println("Shift10
"+shiftDate.toString());
                    currentTime = shiftDate.toString();
                    pst2.setTimestamp(1,shiftDate);
                    ResultSet rs3 = pst2.executeQuery();

                    rs3.next();

                    dt60min = rs3.getInt(1);
                    //System.out.println("The 60min dt: "+dt60min);


                    //discharge temp for t+80 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t+80";
```

```
                    PreparedStatement pst3 = con.prepareStatement(query);

                    shiftDate = new
java.sql.Timestamp(timeshift80min.getTimeInMillis());
                    //System.out.println("Shift80
"+shiftDate.toString());
                    currentTime = shiftDate.toString();
                    pst3.setTimestamp(1,shiftDate);
                    ResultSet rs4 = pst3.executeQuery();

                    rs4.next();

                    dt80min = rs4.getInt(1);
                    //System.out.println("The 15min dt: "+dt15min);

                }


                catch (SQLException e) {

                    //writer.println("Elapsed time per period: " +
((System.currentTimeMillis()-elapsedTime)/1000));
                    //System.out.println("Elapsed time per period: " +
((System.currentTimeMillis()-elapsedTime)/1000));
                    //writer.println("Timeshift out of range!; Time=
"+currentTime);
                    dt40min=0;
                    dt60min=0;
                    dt80min=0;

                    //e.printStackTrace(writer);
                    //the timeshift trailings - end of range
                    writer.println("**********************");
                    writer.println(query);
                    writer.println(currentTime);
                    writer.println(step);
                    writer.println("**********************");
                    writer.flush();

                }


                //inserting the shifted discharge temperatures
                query = "update processData set dt40min=?, dt60min=?,
dt80min=? where DateTime = ?";
                step = "updating the processData info";
                PreparedStatement pst5 = con.prepareStatement(query);

                shiftDate = new
java.sql.Timestamp(currentTimeshiftDate.getTimeInMillis());
//                System.out.println("Update: "+query);
//                System.out.println("dt40min: "+dt40min);
//                System.out.println("dt60min: "+dt60min);
//                System.out.println("dt80min: "+dt80min);
//                System.out.println("DateTime "+shiftDate.toString());
                currentTime = shiftDate.toString();
```

```
                pst5.setInt(1,dt40min);
                pst5.setInt(2,dt60min);
                pst5.setInt(3,dt80min);
                pst5.setTimestamp(4,shiftDate);
                int out = pst5.executeUpdate();
                //System.out.println("Lines afected: " +out);

                //getting the next datetime available
                query = "select TOP 1 DateTime from processData where
DateTime > ? order by DateTime";
                PreparedStatement pst6 = con.prepareStatement(query);
                step = "End of cycle";
                shiftDate = new
java.sql.Timestamp(currentTimeshiftDate.getTimeInMillis());
                currentTime = shiftDate.toString();
                pst6.setTimestamp(1,new
java.sql.Timestamp(currentTimeshiftDate.getTimeInMillis()));
                ResultSet rs6 = pst6.executeQuery();
                rs6.next();

currentTimeshiftDate.setTimeInMillis((rs6.getTimestamp(1)).getTime());

currentTimeshiftDate80.setTimeInMillis(currentTimeshiftDate.getTimeInMill
is());
                currentTimeshiftDate80.add(Calendar.SECOND,4800);

                //System.out.println("Condition at the end: " +
before(currentTimeshiftDate15,endTimeshiftDate));

            }

            System.out.println("Finished the timeshift!");
            System.out.println("Total elapsed time: " +
((System.currentTimeMillis()-elapsedTime)/1000));

        }
        catch (SQLException e) {
            e.printStackTrace(writer);
            writer.println(query);
            writer.println(currentTime);
            writer.println(step);
            writer.flush();
        }

    }


        public void timeShift80Back() {

        System.out.println("Starting the timeshift...");

        String step = "";
        String query = "";
        String currentTime = "";

        long elapsedTime = System.currentTimeMillis();
```

```
        try{


            currentTime = "'" +startTimeshiftDate.get(Calendar.YEAR) + "-
" + (startTimeshiftDate.get(Calendar.MONTH)+1) + "-" +
startTimeshiftDate.get(Calendar.DAY_OF_MONTH) + " 0:0:0'";
            GregorianCalendar currentTimeshiftDate = new
GregorianCalendar();

currentTimeshiftDate.setTimeInMillis(startTimeshiftDate.getTimeInMillis()
);
            GregorianCalendar timeshiftBack20min = new
GregorianCalendar();
            GregorianCalendar timeshiftBack40min = new
GregorianCalendar();
            GregorianCalendar timeshiftBack60min =new
GregorianCalendar();
            GregorianCalendar timeshiftBack80min =new
GregorianCalendar();

            Connection con = getConnection();

            Statement st = con.createStatement();
            query = "select top 1 id from processData where DateTime >" +
currentTime + " order by DateTime";
            step = "start";
            ResultSet rs = st.executeQuery(query);
            //System.out.println(currentTime);

            rs.next();

            int processDataID = rs.getInt(1);
            //System.out.println(processDataID);

            //System.out.println("currentTimeShiftDate: " +
getDate(currentTimeshiftDate));

            Statement st1 = con.createStatement();
            query = "select DateTime from processData where id =" +
processDataID;
            step = "select first record";
            ResultSet rs1 = st1.executeQuery(query);
            rs1.next();

            Timestamp ts = rs1.getTimestamp(1);
            //System.out.println(ts.toString());
            currentTimeshiftDate.setTimeInMillis(ts.getTime());
            //System.out.println("The current date after initialization:
" + getDate(currentTimeshiftDate));

            while (before(currentTimeshiftDate,endTimeshiftDate)) {

                int dt20minBack = 0;
                int dt40minBack = 0;
                int dt60minBack = 0;
                int dt80minBack = 0;
```

```
                java.sql.Timestamp shiftDate = null;


timeshiftBack20min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis()
);
                timeshiftBack20min.add(Calendar.MINUTE, -20);


timeshiftBack40min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis()
);
                timeshiftBack40min.add(Calendar.MINUTE, -40);


timeshiftBack60min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis()
);
                timeshiftBack60min.add(Calendar.MINUTE, -60);


timeshiftBack80min.setTimeInMillis(currentTimeshiftDate.getTimeInMillis()
);
                timeshiftBack80min.add(Calendar.MINUTE, -80);

                try {

                    //discharge temp for t+5 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t-20";
                    PreparedStatement pst1 = con.prepareStatement(query);
                    shiftDate = new
java.sql.Timestamp(timeshiftBack20min.getTimeInMillis());
                    //System.out.println("Shift5 "+shiftDate.toString());
                    //System.out.println(query);
                    pst1.setTimestamp(1,shiftDate);
                currentTime = shiftDate.toString();
                    ResultSet rs2 = pst1.executeQuery();

                    rs2.next();

                    dt20minBack = rs2.getInt(1);
                    //System.out.println("The 20min back dt:
"+dt40minBack);

                    //discharge temp for t+10 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t-40";
                    PreparedStatement pst2 = con.prepareStatement(query);

                    shiftDate = new
java.sql.Timestamp(timeshiftBack40min.getTimeInMillis());
                    //System.out.println("Shift10
"+shiftDate.toString());
                    currentTime = shiftDate.toString();
                    pst2.setTimestamp(1,shiftDate);
                    ResultSet rs3 = pst2.executeQuery();
```

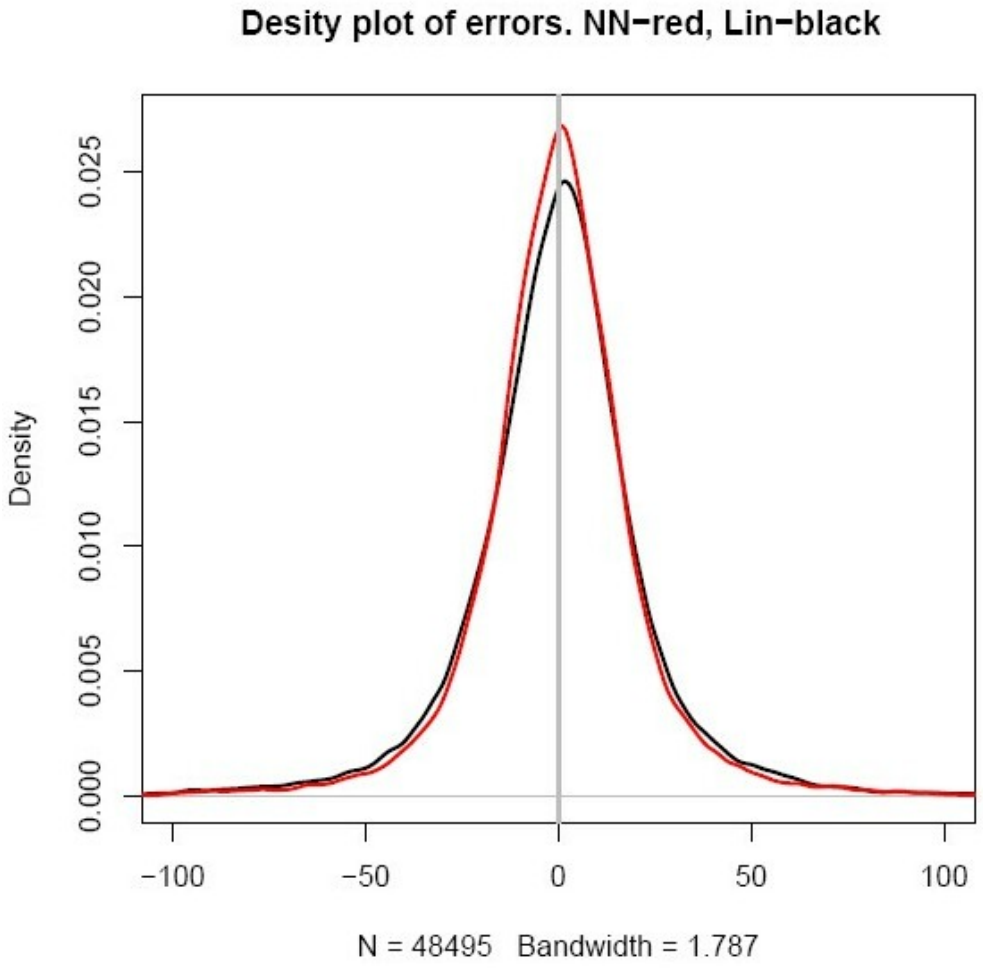
```
                    rs3.next();

                    dt40minBack = rs3.getInt(1);
                    //System.out.println("The 40 min Back dt:
"+dt60minBack);


                    //discharge temp for t+15 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t-60";
                    PreparedStatement pst3 = con.prepareStatement(query);

                    shiftDate = new
java.sql.Timestamp(timeshiftBack60min.getTimeInMillis());
                    //System.out.println("Shift15
"+shiftDate.toString());
                    currentTime = shiftDate.toString();
                    pst3.setTimestamp(1,shiftDate);
                    ResultSet rs4 = pst3.executeQuery();

                    rs4.next();

                    dt60minBack = rs4.getInt(1);
                    //System.out.println("The 60 min Back dt:
"+dt60minBack);

                    //discharge temp for t+20 min
                    query = "select k4_tit5_noncorrige from processData
where DateTime = ?";
                    step = "select the t-80";
                    PreparedStatement pst4 = con.prepareStatement(query);

                    shiftDate = new
java.sql.Timestamp(timeshiftBack80min.getTimeInMillis());
                    //System.out.println("Shift20
"+shiftDate.toString());
                    currentTime = shiftDate.toString();
                    pst4.setTimestamp(1,shiftDate);
                    ResultSet rs5 = pst4.executeQuery();

                    rs5.next();

                    dt80minBack = rs5.getInt(1);
                    //System.out.println("The 80 min Back dt:
"+dt80minBack);

                }


                catch (SQLException e) {

                    //writer.println("Elapsed time per period: " +
((System.currentTimeMillis()-elapsedTime)/1000));
                    //System.out.println("Elapsed time per period: " +
((System.currentTimeMillis()-elapsedTime)/1000));
```

```
                    //writer.println("Timeshift out of range!; Time=
"+currentTime);
                    dt20minBack=0;
                    dt40minBack=0;
                    dt60minBack=0;
                    dt80minBack=0;

                    //e.printStackTrace(writer);
                    //the timeshift trailings - end of range
                    writer.println("**********************");
                    writer.println(query);
                    writer.println(currentTime);
                    writer.println(step);
                    writer.println("**********************");
                    writer.flush();

                }





                //inserting the shifted discharge temperatures
                query = "update processData set dtBack20min=?,
dtBack40min=?, dtBack60min=?, dtBack80min=? where DateTime = ?";
                step = "updating the processData info";
                PreparedStatement pst5 = con.prepareStatement(query);

                shiftDate = new
java.sql.Timestamp(currentTimeshiftDate.getTimeInMillis());
//                System.out.println("Update: "+query);
//                System.out.println("dt5min: "+dt5min);
//                System.out.println("dt10min: "+dt10min);
//                System.out.println("dt15min: "+dt15min);
//                System.out.println("dt20min: "+dt20min);
//                System.out.println("DateTime "+shiftDate.toString());
                currentTime = shiftDate.toString();

                pst5.setInt(1,dt20minBack);
                pst5.setInt(2,dt40minBack);
                pst5.setInt(3,dt60minBack);
                pst5.setInt(4,dt80minBack);
                pst5.setTimestamp(5,shiftDate);
                int out = pst5.executeUpdate();
                //System.out.println("Lines afected: " +out);

                //getting the next datetime available
                query = "select TOP 1 DateTime from processData where
DateTime > ? order by DateTime";
                PreparedStatement pst6 = con.prepareStatement(query);
                step = "End of cycle";
                shiftDate = new
java.sql.Timestamp(currentTimeshiftDate.getTimeInMillis());
                currentTime = shiftDate.toString();
                pst6.setTimestamp(1,new
java.sql.Timestamp(currentTimeshiftDate.getTimeInMillis()));
```

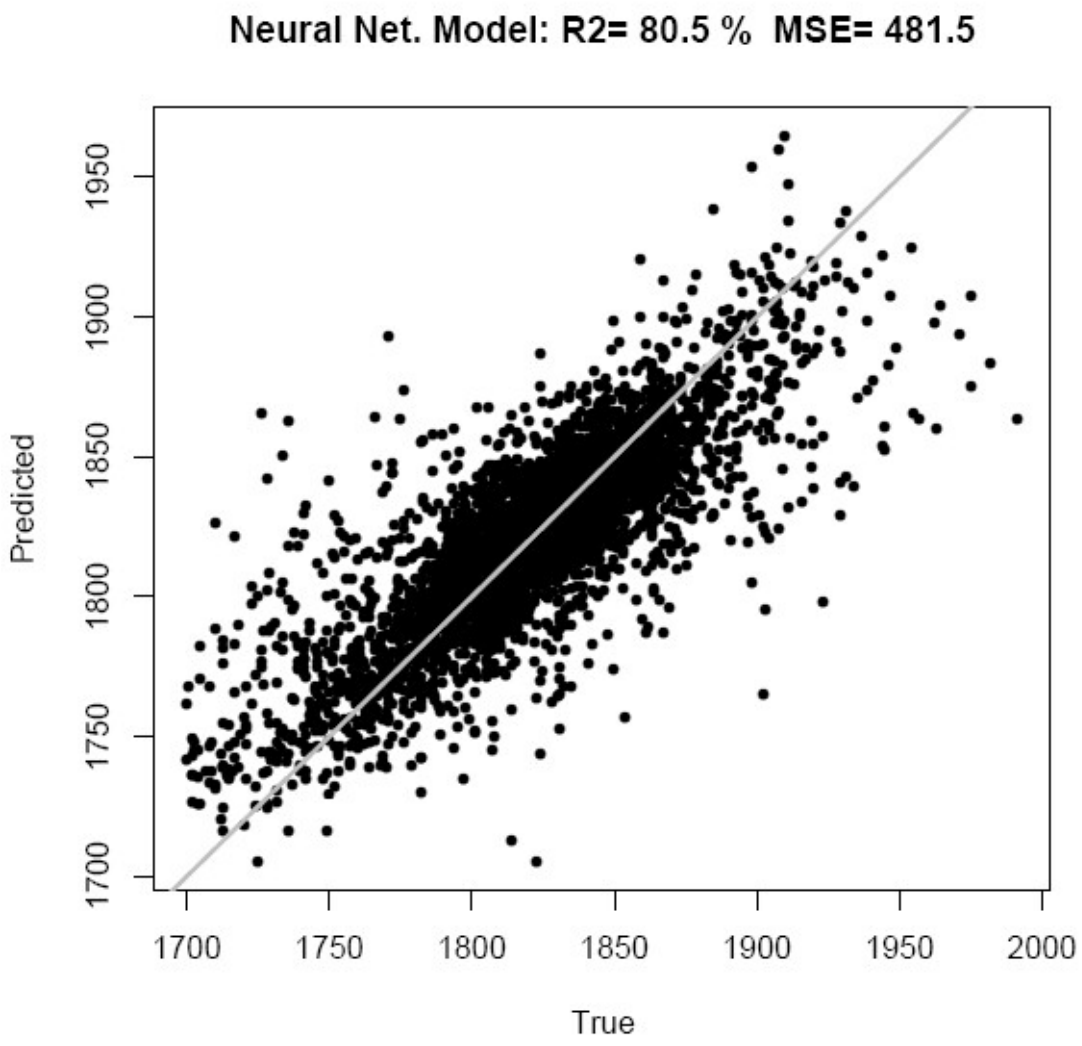
```
                ResultSet rs6 = pst6.executeQuery();
                rs6.next();

currentTimeshiftDate.setTimeInMillis((rs6.getTimestamp(1)).getTime());

                //System.out.println("Condition at the end: " +
before(currentTimeshiftDate15,endTimeshiftDate));

            }

            System.out.println("Finished the timeshift!");
            System.out.println("Total elapsed time: " +
((System.currentTimeMillis()-elapsedTime)/1000));

        }
        catch (SQLException e) {
            e.printStackTrace(writer);
            writer.println(query);
            writer.println(currentTime);
            writer.println(step);
            writer.flush();
        }

    }


}
```

Appendix 2 - Neural Network Results

Density Plot of Errors:



Predicted versus Real (True), media graph:



Distribution of Error versus the Disharge Temperature for the Neural Network Model as well as for the Linear Model: