Petroleum well drilling monitoring through cutting image analysis and artificial intelligence techniques

Ivan R. Guilherme, Aparecido N. Marana, João P. Papa, Giovani Chiachia, Luis C.S. Afonso, Kazuo Miura, Marcus V.D. Ferreira, Francisco Torres

1. Introduction

Offshore petroleum well drilling is an expensive, complex and time-consuming operation and it demands a high qualification level from the drilling executors.

One of the trends of the oil industry is the application of real time measurements and optimization of production operations with the purpose of guaranteeing a safe and effective/low cost drilling execution. Nowadays, there exists several data acquisition systems for petroleum well drilling monitoring, in which a large amount of data is generated at each time. One of these systems is the Mud-Logging, which is responsible for measuring a set of mechanical and geological parameters.

The data generated by Mud-Logging, together with the cutting analysis produced during the drilling operation, allows the drilled soil lithological analysis (Tavares, 2006), which are carried out in deep ranges defined by geology. The generated cutting samples available at the vibrating shake shakers are examined by some expert technician in order to evaluate whether a problem is occurring during the drilling process. Generally, these cuttings have similar shape and sizes in typical situations, and distortions beyond the known normal patterns can indicate the presence of some anomaly, such that the collapse of the well borehole walls (Tavares, 2006).

Some works have been dedicated for monitoring the petroleum well drilling process (Frantiek et al., 2000; Hayajneh, 2007; Mcleod and Minarovic, 1994; de Serapião et al., 2007; Grigorescu et al., 2008; ao Serapi and Mendes, 2009), but none of them were guided by the cutting image analysis. Frantiek et al. (2000) proposed to monitor the rock disintegration process at drilling with the application of acoustic signal. A fourier transform of the generated signal was performed for further statistical analysis. de Serapião et al. (2007) used artificial immune systems for classification of several stages in petroleum drilling. Coelho et al. (2005), Fonseca et al. (2006) and Yilmaz et al. (2002) used neural networks for monitoring drilling activities, and Serapiao et al. (2006) applied support vector machines for classification of petroleum well drilling operations.

However, its is known that an artificial neural network with multi-layer perceptrons (ANN-MLP), for example, can address linearly, piecewise linearly, and nonlinearly separable problems, but not non-separable situations (Haykin, 1994). As an unstable classifier, collections of ANN-MLP (Kuncheva, 2004) can improve its performance up to some unknown limit of classifiers (Reyzin and Schapire, 2006). Support vector machines (SVM) have been proposed to overcome the nonlinearly separable problem, by assuming linearly separable classes in a higher-dimensional space.
feature space (Boser et al., 1992). Its computational cost rapidly increases with the training set size and the number of support vectors. As a binary classifier, multiple SVMs are required to solve a multi-class problem (Duan and Keerthi, 2005). Tang and Mazzoni (2006) proposed a method to reduce the number of support vectors in the multi-class problem. Their approach suffers from slow convergence and high computational cost, because they first minimize the number of support vectors in several binary SVMs, and then share these vectors among the machines. Panda et al. (2006) presented a method to reduce the training set size before computing the SVM algorithm. Their approach aims to identify and remove samples likely related to non-support vectors. However, in all SVM approaches, the assumption of separability may also not be valid in any space of finite dimension (Collobert and Bengio, 2004).

Recently, a novel graph-based classifier that reduce the pattern recognition problem to an optimal-path forest (OPF) computation in the feature space induced by a graph was presented (Papa et al., 2009b). This kind of classifier do not interpret the classification task as a hyperplanes optimization problem, but as a combinatorial optimum-path computation from some key samples (prototypes) to the remaining nodes. Each prototype becomes a root from its optimum-path tree and each node is classified according to its strongly connected prototype, that defines a discrete optimal partition (influence region) of the feature space.

The OPF classifier has some advantages with respect to the aforementioned classifiers: (i) it is free of parameters, (ii) do not assume any shape/separability of the feature space and (iii) run training phase faster. The OPF classifier have been demonstrated to be superior than ANN-MLP and similar to SVM, but much faster (Papa et al., 2009b).

As aforementioned, the previous works use only the information provided by sensors to monitor and assess the petroleum well drilling process. Some important information, such as the cutting information is extremely important image-based system like the proposed one. Based on the cutting's size and shape, a well collapse can be identified in time to avoid further damages. The main idea here, in the future, is to use the information provided by our system into Mud-Logging, aiming its better performance.

3. Artificial intelligence background

In this section we will briefly discuss some of the artificial intelligence techniques used in our work. Sections 3.1 and 3.2 present, respectively, support vector machines and support vector regression. Sections 3.3 and 3.4 address the optimum-path forest classifiers and fitting functions.

3.1. Support vector machines

One of the fundamental problems of learning theory is stated as: given two classes of known objects, assign one of them to a new unknown object. Thus, the objective in a two-class pattern recognition is to infer a function

\[ f : X \rightarrow \{-1, 1\} \] (1)

regarding the input–output of the training data (Schlkopf and Smola, 2002).

Based on the principle of structural risk minimization (Vapnik, 1999), the SVM optimization process is aimed at establishing a separating function while accomplishing with the trade-off that exists between generalization and overfitting.

In his former time, Vapnik (1999) considered the class of hyperplanes in some dot product space \( \mathcal{H} \).

\[ \langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \] (2)
where \( \mathbf{w}, \mathbf{x} \in \mathcal{H}, b \in \mathbb{R} \), corresponding to decision functions
\[
 f(x) = \text{sgn} \left( \langle \mathbf{w}, \mathbf{x} \rangle + b \right)
\]
and, based on two arguments, he proposed the generalized portrait learning algorithm for problems which are separable by hyperplanes:

1. Among all hyperplanes separating the data, there exists a unique optimal hyperplane distinguished by the maximum margin of separation between any training point and the hyperplane.

2. The overfitting of the separating hyperplanes decreases with increasing margin.

So, to construct the optimal hyperplane, it is necessary to solve
\[
 \begin{align*}
 \text{minimize } & \frac{1}{2} \| \mathbf{w} \|^2 \\
 \text{subject to } & \quad y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \text{for all } i = 1, \ldots, m \\
 \end{align*}
\]
with the constraint (5) ensuring that \( f(\mathbf{x}) \) will be +1 for \( y = +1 \) and -1 for \( y = -1 \), and also fixing the scale of \( \mathbf{w} \). A wide exposition of these arguments is provided by Schölkopf and Smola (2002).

The function \( \tau \) in (4) is called the objective function, while (5) are the inequality constraints. Together, they form a so-called constrained optimization problem. The separating function is then a weighted combination of elements of the training set. These elements are called support vectors and characterize the boundary between the two classes.

The replacement referred to as the kernel trick (Schölkopf and Smola, 2002) is used to extend the concept of hyperplane classifiers to nonlinear support vector machines. However, even with the advantage of “kernelizing” the problem, the separating hyperplane may still not exist. To allow that some examples may violate Eq. (5), the slack variables \( \xi \geq 0 \) are introduced (Schölkopf and Smola, 2002), which leads to the constraints
\[
 y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \quad \text{for all } i = 1, \ldots, m
\]
A classifier that generalizes well is then found by controlling both the margin (through \( \| \mathbf{w} \| \) ) and the sum of the slacks variables \( \sum_i \xi_i \). In this context, a possible accomplishment of such a soft margin classifier is obtained by minimizing the objective function
\[
 \tau(\mathbf{w}, \xi) = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{m} \xi_i
\]
subject to the constraint (6), where the constant \( C > 0 \) determines the balance between overfitting and generalization. Due to the tuning variable \( C \), this kind of SVM is normally referred to as C-SVC and represents SVM classification on its original form (Cortes and Vapnik, 1995).

3.2. Support vector regression

The SVM framework was firstly proposed for classification purposes. Nevertheless, its intuition can be slightly modified to accomplish with more general problems called regression estimation. Rather than dealing with outputs \( y \in \{ +1 \} \), the regression case is concerned with estimating real-valued functions.

To cope with regression, a different soft margin is considered during the optimization. Such a margin is constructed by using Vapnik’s \( \varepsilon \)-insensitive loss function (Schölkopf and Smola, 2002), which quantifies the loss incurred by predicting \( f(\mathbf{x}) \) instead of \( y \) as
\[
 c(x,y,f(x)) = |y - f(x)|_\varepsilon = \max(0,|y - f(x)| - \varepsilon)
\]
Thus, to estimate the linear regression \( f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \), the following equation has to be minimized:
\[
 \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{m} |y_i - f(\mathbf{x}_i)|_\varepsilon
\]
As in the classification case, it is possible to transform this into a constrained optimization problem by introducing slack variables. Due to the nature of the constraints, here is necessary two types of slack variables for the cases \( f(\mathbf{x}) - y_i > \varepsilon \) and \( y_i - f(\mathbf{x}) > \varepsilon \), which are, respectively, denoted as \( \varepsilon_i \) and \( \varepsilon_i^* \) and collectively represented as \( \xi_i \).

The optimization problem is then given by
\[
 \begin{align*}
 \text{minimize } & \tau(\mathbf{w}, \xi) = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*) \\
 \text{subject to } & \quad f(\mathbf{x}_i) - y_i \leq \varepsilon + \xi_i \\
 & \quad y_i - f(\mathbf{x}_i) \leq \varepsilon + \xi_i^* \\
 & \quad \xi_i, \xi_i^* \geq 0 \quad \text{for all } i = 1, \ldots, m
\end{align*}
\]
As it can be observed in (11) and (12), any error smaller than \( \varepsilon \) does not infer penalties in the objective function (10). This way, a tube with radius \( \varepsilon \) is fitted to the data and the trade-off between generalization and overfitting is determined by minimizing (10) (Schölkopf and Smola, 2002). This sort of regression SVMs is commonly referred to as \( \varepsilon \)-SVR.

3.3. Optimum-path forest classifier

Let \( Z_1 \) and \( Z_2 \) be the training and test sets with \( |Z_1| \) and \( |Z_2| \) samples such as points or image elements (e.g., pixels, voxels, shapes and texture information). Let \( \lambda(s) \) be the function that assigns the correct label \( i, i = 1,2, \ldots, c \), from class \( i \) to any sample \( s \in Z_1 \cup Z_2 \). \( Z_1 \) is a labeled set used to the design of the classifier and \( Z_2 \) is used to assess the performance of classifier and it is kept unseen during the project.

Let \( S \subseteq Z_1 \) be a set of prototypes of all classes (i.e., key samples that best represent the classes). Let \( v \) be an algorithm which extracts \( n \) attributes (color, shape or texture properties) from any sample \( s \in Z_1 \cup Z_2 \) and returns a vector \( \mathbf{v}(s) \in \mathbb{R}^n \). The distance \( d(s,t) \) between two samples, \( s \) and \( t \), is the one between their feature vectors \( \mathbf{v}(s) \) and \( \mathbf{v}(t) \). One can use any valid metric (e.g., Euclidean) or a more elaborated distance algorithm.

Our problem consists of using \( S, \langle v, d \rangle \) and \( Z_1 \) to project an optimal classifier which can predict the correct label \( \lambda(s) \) of any sample \( s \in Z_2 \). The OPF classifier creates a discrete optimal partition of the feature space such that any sample \( s \in Z_2 \) can be classified according to this partition. This partition is an optimum path forest (OPF) computed in \( \mathbb{R}^n \) by the image foresting transform (IFT) algorithm (Falcão et al., 2004).

3.3.1. Training

Let \( \{Z_1, A\} \) be a complete graph whose nodes are the training samples and any pair of samples defines an arc in \( A = Z_1 \times Z_1 \). The arcs do not need to be stored and so the graph does not need to be explicitly represented. A path is a sequence of distinct samples \( \pi_i = \langle s_1, s_2, \ldots, t \rangle \) with terminus at a sample \( t \). A path is said trivial if \( \pi_i = \langle t \rangle \). We assign to each path \( \pi_i \) a cost \( f(\pi_i) \) given by a connectivity function \( f \). A path \( \pi_i \) is said optimum if \( f(\pi_i) \leq f(\pi_\tau) \) for any other path \( \pi_\tau \). We also denote by \( \pi_\tau = \langle s, t \rangle \) the concatenation of a path \( \pi_i \) and an arc \( \{s,t\} \).
We will address the connectivity function \( f_{\text{max}} \):

\[
f_{\text{max}}(s) = \begin{cases} 
0 & \text{if } s \in S \\
+\infty & \text{otherwise}
\end{cases}
\]

such that \( f_{\text{max}}(\pi_i \cdot \langle s,t \rangle) \) computes the maximum distance between adjacent samples along the path \( \pi_i \cdot \langle s,t \rangle \). The minimization of \( f_{\text{max}} \) assigns to every sample \( t \in Z_i \) an optimum path \( P(t) \) from the set \( S \subseteq Z_i \) of prototypes, whose minimum cost \( C(t) \) is

\[
C(t) = \min_{\forall s \in (d,A)} f_{\text{max}}(\pi_i)
\]

The minimization of \( f_{\text{max}} \) is computed by Algorithm 1, called OPF algorithm (Papa et al., 2009b), which is an extension of the general image foresting transform (IFT) algorithm (Falcão et al., 2004) from the image domain to the feature space, here specialized for \( f_{\text{max}} \). This process assigns one optimum path from \( S \) to each training sample \( t \) in a non-decreasing order of minimum cost, such that the graph is partitioned into an optimum-path forest \( P(t) \) (a function with no cycles which assigns to each \( t \in Z_i \) its predecessor \( P(t) \) in \( P(t) \) or a marker nil when \( t \in S \). The root \( R(t) \in S \) of \( P(t) \) can be obtained from \( P(t) \) by following the predecessors backwards along the path, but its label is propagated during the algorithm by setting \( L(t) \leftarrow l(R(t)) \). The OPF algorithm is described below, as well its explanation.

**Algorithm 1. OPF Algorithm**

**Input:** A training set \( Z_i \), labeled prototypes \( S \subseteq Z_i \), the pair \( (d,A) \) for feature vector and distance computations.

**Output:** Optimum-path forest \( P \), cost map \( C \) and label map \( L \).

**Auxiliary:** Priority queue \( Q \) and cost variable \( cst \).

1. For each \( s \in Z_i \), set \( C(s) \leftarrow +\infty \).
2. For each \( s \in S \), do
3. \( C(s) \leftarrow 0 \), \( P(s) \leftarrow \text{nil} \), \( L(s) \leftarrow \lambda(s) \), and insert \( s \) in \( Q \).
4. While \( Q \) is not empty, do
5. Remove from \( Q \) a sample \( s \) such that \( C(s) \) is minimum.
6. For each \( t \in Z_i \), such that \( t \neq s \) and \( C(t) > C(s) \), do
7. Compute \( cst \leftarrow \max(C(s),d(s,t)) \).
8. If \( cst < C(t) \), then
9. If \( C(t) \neq +\infty \), then remove \( t \) from \( Q \).
10. \( P(t)\leftarrow s \), \( L(t)\leftarrow L(s) \) and \( C(t)\leftarrow \text{cst} \).
11. Insert \( t \) in \( Q \).

Lines 1–3 initialize maps and insert prototypes in \( Q \). The main loop computes an optimum path from \( S \) to every sample \( s \) in a non-decreasing order of minimum cost (Lines 4–11). At each iteration, a path of minimum cost \( C(s) \) is obtained in \( P \) when we remove its last node \( s \) from \( Q \) (Line 5). Ties are broken in \( Q \) using first-in-first-out policy. That is, when two optimum paths reach an ambiguous sample \( s \) with the same minimum cost, \( s \) is assigned to the first path that reached it. Note that \( C(t) > C(s) \) in Line 6 is false when \( t \) has been removed from \( Q \) and, therefore, \( C(t) \neq +\infty \) in Line 9 is true only when \( t \in Q \). Lines 8–11 evaluate if the path that reaches an adjacent node \( t \) through \( s \) is cheaper than the current path with terminus \( t \) and update the position of \( t \) in \( Q \), \( C(t) \), \( L(t) \) and \( P(t) \) accordingly.

We say that \( S^* \) is an optimum set of prototypes when Algorithm 1 minimizes the classification errors in \( Z_i \). \( S^* \) can be found by exploiting the theoretical relation between minimum-spanning tree (MST) (Cormen et al., 1990) and optimum-path tree for \( f_{\text{max}} \) (Allene et al., 2007; Rocha et al., 2008). By computing a MST in the complete graph (\( Z_i, A \)), we obtain a connected acyclic graph whose nodes are all samples of \( Z_i \) and the arcs are unweighted and directed by the distances \( d \) between adjacent samples. The spanning tree is optimum in the sense that the sum of its arc weights is minimum as compared to any other spanning tree in the complete graph. In the MST, every pair of samples is connected by a single path which is optimum according to \( f_{\text{max}} \). That is, the minimum-spanning tree contains one optimum-path tree for any selected root node.

The optimum prototypes are the closest elements of the MST with different labels in \( Z_i \). By removing the arcs between different classes, their adjacent samples become prototypes in \( S^* \) and Algorithm 1 can compute an optimum-path forest in \( Z_i \). Note that, a given class may be represented by multiple prototypes (i.e., optimum-path trees) and there must exist at least one prototype per class. It is not difficult to see that the optimum paths between classes tend to pass through the same removed arcs of the minimum-spanning tree. The choice of prototypes as described above aims to block these passages, reducing the chances of samples in any given class be reached by optimum paths from prototypes of other classes.

### 3.3.2 Classification

For any sample \( t \in Z_i \), we consider all arcs connecting \( t \) with samples \( s \in Z_i \), as though \( t \) were part of the training graph. Considering all possible paths from \( S^* \) to \( t \), we find the optimum path \( P^*(t) \) from \( S^* \) and label \( t \) with the class \( \lambda(t) \). That is, \( \lambda(t) \) is the most strongly connected prototype \( R(t) \in S^* \). This path can be identified incrementally, by evaluating the optimum cost \( C(t) \) as

\[
C(t) = \min_{\forall s \in Z_i} \left( \max(C(s),d(s,t)) \right)
\]

Let the node \( s^* \in Z_i \) be the one that satisfies Eq. (16) (i.e., the predecessor \( P(t) \) in the optimum path \( P^*(t) \)). Given that \( L(s^*) = \lambda(t) \), the classification simply assigns \( L(s^*) \) as the class of \( t \). An error occurs when \( L(s^*) \neq \lambda(t) \).

### 3.4 Fitting Functions

As we know, the problem of pattern recognition rely onto labeling an unknown pattern in a predetermined class. However, there exists some applications in which we cannot deal with predefined classes, and the main objective is to estimate, for each pattern, a measure in a valued interval.

Given a set of patterns in which we know the desired measures (training set), it is possible to use this set to estimate the coefficients of a fitting function. In such a way, this function can be further used to estimate the unknown pattern measures.

Polynomial functions can be used as fitting functions. Although high order polynomials can obtain better approximation results, they require bigger training sets, and also generate oscillations in the fitted function. For instance, the polynomial of degree three for bidimensional patterns can be defined as

\[
p(x) = p_0 + p_1 x + p_2 x^2 + p_3 x^3
\]

As we can see, the polynomials of degree three of two variables have 16 coefficients, i.e. \( c_i, i = 1, 2, \ldots, 16 \), which need to be estimated from the training set.

#### 3.4.1 Cubic polynomial coefficient estimation

Let the polynomial given by Eq. (17). Suppose that we know the values of this polynomial for a set of \( N \) bidimensional patterns.
According to three predefined volume classes: E (none concentration), L (low concentration of cuttings) and finally, the class H, which means high concentration of cuttings at the vibrating shale shaker. Fig. 2 show an example from each class of cutting volume. We performed a texture analysis to create the feature vector of each image, which is composed by four Haralick (1979) texture features: energy, entropy, homogeneity and contrast. These features are simply and fast computed based on the co-occurrence matrix of each image and have been demonstrated to be very discriminative in our problem.

As aforementioned, we used for classification purposes a collection of supervised classifiers: OPF, ANN-MLP, SVM and BC, which require a labeled set of samples for training that will be used to calibrate the system. The training images were obtained by manually selecting some frames from a video (20 frames/samples from each class) and a technician was requested to classify each frame as belonging to class E, L or H. This same technician was requested again to label the remaining images (20 samples from each class) to compose the test set, in order to allow a quantitative evaluation of the accuracy of each classifier. Essentially, we have a dataset with 120 samples equally distributed among the classes: a training set Z1 with 50% of the samples and a test set Z2 with 50% of the samples. These samples were randomly selected and each experiment was repeated 10 times with different sets Z1 and Z2 to compute the mean accuracy values. The average computational time of each classifier in minutes for training and classification is also reported.

For SVM implementation, we used the LibSVM package (Chang and Lin, 2001) with radial basis function (RBF) kernel, parameter optimization and the one-versus-one strategy for the multi-class problem. We use the Fast Artificial Neural Network Library (FANN) (Nissen, 2003) to implement the ANN-MLP. The network configuration is \( x:y:z \), where \( x = n \) (number of features), \( y = |Z_1| - 1 \) and \( z = c \) (number of classes) are the number of neurons in the input, hidden and output layers, respectively. For OPF we used the LibOPF package (Papa et al., 2009a) and for BC we used our own implementation.

Table 1 shows the mean accuracy and execution time in seconds for each classifier for recognizing the cutting density at the vibrating shale shaker for petroleum well drilling monitoring purposes.

Table 1 shows that OPF classifier outperformed all remaining classifiers, despite the fact of all of them achieved very good and similar results, which emphasize the robustness of the texture features in our problem. The OPF classifier was about 665 times faster than SVM in this case. Note that SVM algorithm had a slow performance due to the fact of the optimization procedure implemented in the LibSVM (Chang and Lin, 2001). However, by removing the optimization procedures, this processing time could be decreased. On the other hand, this could produce lower classification rates.

**Table 1**

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Mean accuracy</th>
<th>Mean execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPF</td>
<td>99.89 ± 0.001</td>
<td>0.0023</td>
</tr>
<tr>
<td>ANN-MLP</td>
<td>99.00 ± 1.200</td>
<td>0.0330</td>
</tr>
<tr>
<td>SVM</td>
<td>99.25 ± 0.612</td>
<td>1.5314</td>
</tr>
<tr>
<td>BC</td>
<td>92.33 ± 0.570</td>
<td>0.0020</td>
</tr>
</tbody>
</table>

Fig. 2. Zoomed regions of the vibrating shale shaker: (a) none concentration, (b) low concentration and (c) high concentration of cuttings at the vibrating shale shaker.
4.2. Cutting volume estimation

In this section, we validate SVR and FF with a cubic polynomial function for estimating cutting’s volume generated during the well drilling process. We used a labeled set of frames extracted from a video sequence for training SVR and FF, and also to evaluate them. Fig. 3 shows some of these frames, as well their annotated values. Again, we used the same set of Haralick features extracted from images as aforementioned in Section 4.1 for training SVR and FF.

After training SVR and FF with 50% of the labeled set, we tested them on the remaining set. Fig. 5 displays the results.

Fig. 4 displays the volume estimation by SVR incorporated at our developed system (at the bottom-right side). Given a video of the cutting at the vibrating shale shaker, our software can estimate in real time the cutting volume by applying support vector regression. Notice that we choose SVR instead of FF due to the best results obtained by the former approach (Fig. 5).

We can see that both techniques, i.e., SVR and FF with cubic polynomial function, obtained good results if we compared them with the ground truth (annotated curve). Note that SVR achieved the best results, with 0.007609 of mean squared error. The FF methodology achieved 0.008356 of mean squared error.

5. Conclusions

Well drilling monitoring is an important and crucial task to detect and prevent problems in the drilling process. Several studies have been developed for drilling inspection, but none of them take care about analyzing the generated cutting at the vibrating shale shakers during the drilling process. The cuttings shape and size, and even so its volume (density) are important features that allow us to identify possible problems during the drilling operation. Unexpected changes in the cuttings volume, for instance, can indicate a possible collapse of the well borehole walls.

Based on this assumption, we propose here an image-based well drilling monitoring intelligent system composed by two modules: data acquisition system and data analysis. The cutting images at the vibrating shale shaker were acquired and sent to the
data analysis module, which was previously trained with calibrated images (labeled by a technician). The system can detect in real time anomalies with respect to changes in the cutting volume in real time using support vector regression, which can indicate a possible problem during the drilling process, and further sending warning alerts. As far we know we are the first to develop an image-based system for monitoring well drilling operations based on the cuttings analysis.

In order to assess the robustness of our systems, we conducted two series of experiments. In the former, we applied a collection of four supervised classifiers for cutting volume supervised classification: OPF, ANN-MLP, SVM and BC. The first one outperformed all the remaining classifiers, both in terms of accuracy and efficiency. The images were mapped into the feature space domain by representing each one of them with a 4-D feature vector, composed by four texture Haralick features: energy, homogeneity, contrast and entropy. These features have been demonstrated to be very robust in to discriminate each frame with respect to one of the aforementioned class: E (empty shale), L (low volume at the shale) or H (high volume at the shale). Another contribution of our work is that we are the first to apply the OPF classifier in this field of knowledge.

In the latter experiment, we applied support vector regression and fitting function with cubic polynomial function for estimating the cutting volume at real time, in which SVR achieved the best results. Again, as far we know, we are the first into applying SVR and FF for cutting volume estimation using images obtained at the vibrating shale shakers. As future work, we intend into join our system with other commonly used well drilling monitoring systems. Mud-Logging. We are nowadays working to establish a quantitative measure for cuttings volume.

References
