Automated Classification of Defect Signatures in Pipelines using Ultrasonic Images

Clément FOUQUET 1, Aymeric Histace 2, Patrick DUVAUT 2

1 TRAPIL Company, 1 Rue Charles Edouard Jeanneret, Poissy, France
Phone: +33 0682833754 ; e-mail: fouclem@gmail.com
2 ETIS / ENSEA; University of Cergy-Pontoise, 6 Avenue du Ponceau, Cergy, France
Corresponding author: aymeric.histace@u-cergy.fr

Abstract
Non-destructive inspection of oil pipeline integrity is of primary interest for an early detection of defects that can alter the structure of the pipe. Usual inspection is performed using a pipeline inspection device equipped with ultrasonic sensors (up to 512) regularly dispatched all around its circumference. The tool is inserted in the pipeline and is driven by the flow of the medium. The travelling time of the ultrasound beam from the sensor to the internal surface and from the internal surface and the external surface of the pipe is converted into distance that reflects the thickness of the pipe versus the sensor angular position, and the discrete location of the inspection tool along the path. Data are visualized as large images (up to 10 Giga pixels) highly distorted by intense noise. Processing of these data is made manually and is particularly time consuming. Following previous publications (2011 NDT in Progress and 2012 ICNDT conference) in which a focus was given to automatic weld detection along the pipe and automatic interest areas segmentation, this article focuses on the identification of pre-detected areas, using a complete pattern recognition process on totally new data.

Keywords: Image Processing, Machine learning, Ultrasonic NDT, Pipeline Integrity Control

1. Introduction

The Pipeline Operators Forum has undertaken an initiative [1] to specify “Intelligent Pig Inspection” protocols. This method contributes to ‘smart pipeline monitoring’ by automatically detecting and locating interest zones which are good candidates for a following classification step. Defects in pipeline can be of a very large set of natures, from natural causes as corrosion, to human-caused ones such as dents.

This contribution follows two previous publications [2], [3] that presented a very accurate circumferential weld detection method, followed by a rough segmentation of tubes, so we’ll consider in this paper having at our disposal a large amount of sub-images containing interest zones, which may or may not contain an actual defect.

There are several difficulties inherent to the identification of such interest zones: we are in a multi-class problem (four classes of defects and false alarms which we have to reject). The defects are of natural kind: they can present large differences in terms of characteristics (size, shape, texture) and are thus difficult to model accurately. Finally, the population of each class we have at our disposal is very unequal: we have a huge amount of false alarms but quite few defects, especially in certain classes.

In order to address these challenges, we chose to follow a machine learning process. First we extract coherent characteristics from our sub-images, then we use our database to train a classifier algorithm (the Random Forest) to differentiate new data.

2. Introduction to machine learning process

2.1 Overall view

The concept of machine learning is to project usable training data into a multi-dimensional space where it is possible to draw frontiers differentiating the classes we need to identify. Each dimension of the space is formed by a numerical feature extracted from the original data.
(image, signal…). For didactic purposes, let us consider a problem where we need to classify three kinds of Iris flowers: Iris Setosas, Iris Virginicas and Iris Versicolors. If we have at our disposal a set of Irises that we do know the family, we can measure the petal length and the sepal length of each of the flowers: we thus have two features and can project our training data into this space. A representation of the result is shown in Fig. 1.

Fig. 1. Example of training set using two features on the Iris database (WEKA explorer[4])

The concept of machine learning is to use an algorithm named classifier that is able to draw coherent frontiers using the training data in order to use the resulting frontiers to decide on the class of an unknown data by projecting this data into the space and looking in which region it falls. In the example above we can see that finding a good frontier to differentiate the Setosas from the other two is quite easy, but differentiating Versicolors from Virginicas is more complex due to the fact that the two data clusters are merged (overlap). In order to have a good separation and thus finding a good frontier between these two would require additional information, i.e. features. Having more features will often lead to better recognition performances, providing that the features are useful to the considered problem. But growing the dimension of the space also leads to algorithmic difficulties for classifiers, which means a compromise on the number of features have to be found.

2.2 The Random Forest classifier

2.2.1 The binary decision tree

The binary decision tree is one of the oldest classifier designed [5]. Its principle is to realise a succession of one-dimension splits of the feature space in order to separate between the different classes. At each step of the algorithm, it solves an optimisation problem aiming to differentiate classes by choosing the best couple feature / value to split the space. The result is a succession of binary decision forming a tree, which each end (leaf) of the tree line deciding to one class. An example of the construction of a binary decision tree on a very simple problem (two classes, two dimension) is shown on Fig. 2.
The binary decision tree has the particularity to be an unstable classifier. It means that the slightest modification of the training database might result in a large modification of the resulting treeline. It also has a major problem of over-learning, meaning that it has difficulties to generalize and often leads to misclassification on complex problems where classes are overlapped. Fig. 3 illustrates this problem, showing that the increasing number of leaves in the constructed tree results in expected decrease in learning error rate, but not in real error rate, when we provide examples the classifier didn’t learned from.

2.2.1 The Random Forest

The Random Forest was first presented by L. Breiman in [6]. It is an evolution of the binary decision trees, aiming at exploiting their instability to mitigate their over-learning problem. By injecting a slight randomness in the training set and construction process of the tree, the resulting treeline becomes much different. By applying this process on a large number of trees, we obtain a wide variety of trees that have been trained on near-similar problem, but having largely different constructions. By fusing all these trees decisions by majority vote, the overall decision becomes fuzzy in difficult (overlapped) frontiers, reinforcing the generalization capacity.
Random Forest uses two ways to inject randomness in the binary decision tree. The first is called Bootstrap Aggregating (Bagging). It aims to create slight variants of the original training set to train each tree of the forest. Each subset (bootstrap) is created by randomly picking samples in the original set while enabling to pick the same sample more than one time and conserving the effective number of samples. The resulting training set will thus contain duplicates, while some data will not be present. The second way to inject randomness is to alter the way the trees are constructed. Originally, the tree will find the best couple feature / value amongst all features available to realise its splits. The Random Feature Selection (RFS) concept is to find that best couple not amongst all the features but in a subset of randomly selected features at each step (this time without allowing picking the same feature twice).

The addition of these two means to inject randomness in the construction of each unique tree allows the overall forest to be an extremely potent classifier, capable of handling high dimensional feature spaces, with very good generalization capacities.

3. Feature extraction on our problem

3.1 Problematic description

As we stated before we consider a large amount of sub-images, direct result of our previous paper about rough segmentation of tube images. Our problem is composed of five classes we need to differentiate:

- **Dents:** Dents are punctual, geometrical deformation of the tube, most of the time in consequence of external aggression (hit, rock pressuring the tube…) They are usually circle shaped and can be coupled with metal loss. They are often presenting complete signal loss due to the rebound of the ultrasonic wave on a non-perpendicular plane:

![Fig. 4. Illustration of a typical “Dent” default](image)

- **Metal losses:** Metal loss are often fabrication irregularities on the pipeline, or any generic reason that led to a significant loss of metal.

![Fig. 5. Illustration of a typical “Metal loss” default](image)
- **Mid-wall defects:** Mid-wall defects are consequences of the pipeline metal containing air bubbles which can grow and lead to leaks.

![Fig. 6. Illustration of a typical “Mid-Wall-defect” default](image)

- **Corrosions:**

![Fig. 7. Illustration of a typical “Corrosion” default](image)

- **False alarms:** this class contains any noise or misdetection from the rough segmentation, which are to be rejected.

![Fig. 8. Illustration of a typical “False Alarm”](image)

The objective of the identification process is to tag the defects correctly as well as to reject false alarms.

### 3.2 Fine segmentation

The first step is to get a fine segmentation of each defect. Extracting features requires precise edge of the defect, in order to compile shape or texture descriptors. Luckily, the sub-images are close enough to the defect that no noise can deeply alter the detection so a simple Otsu binarisation [7] leads to proper results:
By using this fine segmentation and looking at the biggest connected component, we can extract our feature with precise information.

### 3.3 Choosing features

As we had no prior evidence on which feature we had to use on our problem we decided to go through a two-step approach: first, we would gather all the features that we thought could help identifying defects, then we used a Principal Component Analysis (PCA, as described in [8]) in order to narrow the size of the feature vector, eliminating useless and/or redundant ones. We first gathered a set of feature derived directly from experts operative mode, then we gathered a set of shape describers [9], and finally a last set of texture describers extracted from gray-level co-occurrence matrices (GLCM) [10], [11], [12].

The first feature vector, containing all of the previously stated, was composed of 35 features. After using the PCA we could narrow this vector to 27 features while keeping 99% of the information contained.

These 27 features are divided as we stated before in three categories. First category are the ad-hoc ones, inspired by the experts:

- Length of the image
- Width of the image
- Number of connected components in the binary image
- Area of the biggest connected component
- Relative metal loss in the biggest connected component
- Standard deviation of thickness values in the biggest connected component
- Average gradient on the edges of the biggest connected component
The second category is the shape descriptors:

- Eccentricity: The ratio of the length of the longest chord of the shape to the longest chord perpendicular to it
- Length of the long axis of the best matching ellipse
- Length of the short axis of the best matching ellipse
- Angle of said ellipse
- Perimeter of the biggest connected component

The last category is the texture descriptors, obtained from computing the GLCM on the image. Co-occurrence matrices are defined for a couple \((a, b)\) of grey levels and a translation \(t\). The translation \(t\) is defined by the couple \((d, \theta)\) where \(d\) is the distance and \(\theta\) is the angle formed between \(t\) and the horizontal. Hence, the matrix \(MC(d, \theta)(a, b)\) is the number of couples of sites \((s, s+t)\) of the considered region, separated by the translation vector \(t\) with \(s\) has the grey level \(a\) and \(s+t\) has the grey level \(b\). In the end, for an image coded with \(N\) grey levels, the co-occurrence matrix will be of size \(NxN\). The example shown in Fig. 4, is realised for \(t = (0, -1)\).

\[
\begin{align*}
\text{Source} & = \begin{pmatrix}
1 & 2 & 1 & 3 & 4 \\
2 & 3 & 1 & 2 & 4 \\
3 & 3 & 2 & 1 & 1 \\
\end{pmatrix} \\
\text{GLCM} & = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\end{align*}
\]

Fig. 4. Example of GLCM construction

The texture descriptors we use in our feature vector are directly computed from the GLCM of the biggest connected component:

- Autocorrelation: 
  \[ f_1 = \sum_i \sum_j (i, j) p(i, j) \]

- Correlation: 
  \[ f_2 = \frac{\sum_i \sum_j (i, j) p(i, j) - \mu_x \mu_y}{\sigma_x \sigma_y} \]

- Cluster prominence: 
  \[ f_3 = \sum_i \sum_j (i + j - \mu_x - \mu_y)^4 p(i, j) \]

- Cluster shade: 
  \[ f_4 = \sum_i \sum_j (i + j - \mu_x - \mu_y)^3 p(i, j) \]

- Dissimilarity: 
  \[ f_5 = \sum_i \sum_j |i - j| p(i, j) \]
- Entropy: \( f_6 = -\sum_i\sum_j p(i, j) \log(p(i, j)) \)

- Variance (self explanatory)

- Average of the sums: \( f_7 = \sum_{i=2}^{2N} i p_{x+y}(i) \)

- Variance of the sums: \( f_8 = \sum_{i=2}^{2N} (i - f_7)^2 p_{x+y}(i) \)

- Entropy of the sums: \( f_9 = -\sum_{i=2}^{2N} p_{x+y}(i) \log(p_{x+y}(i)) \)

- Variance of the differences: \( f_{10} = \text{var}(p_{x-y}) \)

- Entropy of the differences: \( f_{11} = -\sum_{i=0}^{N-1} p_{x-y}(i) \log(p_{x-y}(i)) \)

- Information of the measures (1): \( f_{12} = \frac{f_6 - \left\{ -\sum_i\sum_j p(i, j) \log(p_x(i)p_y(j)) \right\}}{\max\{\text{ent}(p_x), \text{ent}(p_y)\}} \)

Information of the correlation (2):

- \( f_{13} = (1 - \exp\left[-2\left\{ -\sum_i\sum_j p_x(i)p_y(j) \log(p_x(i)p_y(j)) \right\}\right] - f_6 \)

- Normalized inverse difference: \( f_{14} = \sum_{i,j=1}^{N} \frac{C_{ij}}{1 + \left|i-j\right|^2 / N^2} \)

With these features we can extract the information we need to train the classifier on the dataset we have at our disposal.

4. Classification and results on real cases

4.1 Dataset and classifier

Our training and our test dataset is coming directly from the result of the previous rough segmentation as we stated before. The test line we used was 12km long, with a diameter change (10 to 12 inch) and a reference thickness change. The dataset contains 350 defects with more than 50,000 false alarms.
To compare our forest, we decided to benchmark them with a neural network [13]. We
decided to reject SVM [14] mainly because of the difficulty in the kernel choice and
parameters.
We used a 10-fold cross validation process to compute our results.

4.2 Balance issue

With such a difference between populations of defects and false alarms, the classifier cannot
be trained directly. Classifiers are designed to get the best overall performance on the training
set, meaning that if we train it directly on such an unbalanced set, it will always decide false
alarm in order to get 99.9% of overall recognition rate, which obviously is irrelevant. We so
have to rebalance our set. Two ways are available: either we diminish the number of false
alarms (simply randomly pick 1% of our false alarm population), or we can increase
artificially our defect population by duplicating the ones we have, adding a slight Gaussian
noise ($\sigma=10\%$) on each feature. We trained our forest with both ways to rebalance the dataset
and results are shown below.

4.2 Results

4.2.1 Results with reduced amount of false alarms

<table>
<thead>
<tr>
<th>False alarm</th>
<th>Corrosion</th>
<th>Metal loss</th>
<th>Dent</th>
<th>Mid wall</th>
</tr>
</thead>
<tbody>
<tr>
<td>False alarm</td>
<td>493</td>
<td>11</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>Corrosion</td>
<td>24</td>
<td>8</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>Metal loss</td>
<td>26</td>
<td>5</td>
<td>45</td>
<td>7</td>
</tr>
<tr>
<td>Dent</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>75</td>
</tr>
<tr>
<td>Mid wall</td>
<td>38</td>
<td>0</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>

Results for Neural Network with reduced amount of false alarms

<table>
<thead>
<tr>
<th>False alarm</th>
<th>Corrosion</th>
<th>Metal loss</th>
<th>Dent</th>
<th>Mid wall</th>
</tr>
</thead>
<tbody>
<tr>
<td>False alarm</td>
<td>523</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Corrosion</td>
<td>35</td>
<td>7</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Metal loss</td>
<td>24</td>
<td>0</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>Dent</td>
<td>8</td>
<td>1</td>
<td>6</td>
<td>71</td>
</tr>
<tr>
<td>Mid wall</td>
<td>26</td>
<td>0</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>

Results for Random Forests with reduced amount of false alarms

4.2.1 Results with increased amount of defects

<table>
<thead>
<tr>
<th>False alarm</th>
<th>Corrosion</th>
<th>Metal loss</th>
<th>Dent</th>
<th>Mid wall</th>
</tr>
</thead>
<tbody>
<tr>
<td>False alarm</td>
<td>51619</td>
<td>15</td>
<td>99</td>
<td>84</td>
</tr>
<tr>
<td>Corrosion</td>
<td>546</td>
<td>3508</td>
<td>267</td>
<td>389</td>
</tr>
<tr>
<td>Metal loss</td>
<td>509</td>
<td>125</td>
<td>7586</td>
<td>239</td>
</tr>
<tr>
<td>Dent</td>
<td>246</td>
<td>145</td>
<td>115</td>
<td>8490</td>
</tr>
<tr>
<td>Mid wall</td>
<td>875</td>
<td>97</td>
<td>132</td>
<td>312</td>
</tr>
</tbody>
</table>

Results for Neural Networks with increased amount of defects
### Results for Random Forests with increased amount of defects

<table>
<thead>
<tr>
<th></th>
<th>False alarm</th>
<th>Corrosion</th>
<th>Metal loss</th>
<th>Dent</th>
<th>Mid wall</th>
</tr>
</thead>
<tbody>
<tr>
<td>False alarm</td>
<td>51888</td>
<td>0</td>
<td>13</td>
<td>26</td>
<td>68</td>
</tr>
<tr>
<td>Corrosion</td>
<td>41</td>
<td>4908</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Metal loss</td>
<td>6</td>
<td>0</td>
<td>8579</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dent</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9292</td>
<td>0</td>
</tr>
<tr>
<td>Mid wall</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17358</td>
</tr>
</tbody>
</table>

#### 4.3 Conclusion

The results for the reduced amount of false alarms are quite equivalent between the Random Forest and the Neural Network. Both have good rejection rate on false alarms and good recognition rate on dents and mid wall. But they both struggle to identify corrosions and metal loss. The reason is simply because of the population in the training set of these two classes, coupled with the fact that those are the hardest to identify for the experts. But as we can see in the increased amount of defects case that, provided we have a large quantity of defects at our disposal to train properly the classifier, results are much better and more equal between classes, with a good advantage in performance for Random Forests. In any case, those results prove our features to be coherent and sufficient to allow a good separation of our class clusters in the feature space.

#### 5. Conclusion

We proposed in this paper a new application for machine learning methods, showing a complete and efficient identification process on pipeline defects. Efficient feature were studied and used to train a Random Forest. In the worst case scenario, with a very limited training dataset, we attain performance of 82% of overall recognition rate. In the best case scenario, with a large quantity of training data, it is possible to attain more than 99% of recognition rate.

### References

1. **Pipeline Operator Forum Draft.** Specifications and requirements for intelligent pig inspection of pipeline, 2009
2. C. Fouquet, P. Duvaut, D. Benboudjema. Automatic and circumferential oriented detection (ACOD) for pipeline intelligent inspection, NDT in progress, 2011
4. WEKA, university of Waikato. www.cs.waikato.ac.nz/ml/weka
9. R.C. Gonzalez, S.L. Eddins Digital image processing using MATLAB. 2002