A NEW TEXTURE SEGMENTATION APPROACH USING DATA MINING ALGORITHMS

Seetha J.[1], **Varadharajan R.**[2]

[1]St. Peter's University, Chennai-54.
[2]SASTRA University, Thanjavur.
Email: Seetha.venkat80@gmail.com

ABSTRACT

Texture segmentation is one of the most important topic for Image analysis, understanding and interpretation. It is actually a topic where a lot of different approaches lead to more or less satisfying results. In general all of them try to match a particular feature or a feature vector which describes the analyzed region. Subsequently a threshold or threshold vector is applied and a texture class is assigned to the region. This paper describes how data mining algorithms can be used advantageously for texture based segmentation. Using a reference image with known texture, a model for a classifier is trained, that is applied to image regions of unknown texture. For the data mining it is necessary to calculate many different features and rate them (e.g by their information gain or correlation) accordingly. Only the best features selected this way are used to train a classifier, which is then used to segment subsequent images. Using this selected classifier it is possible to determine the location where a specific texture occurs in the image. The performance of the classifier is demonstrated for synthetic test images.

Keywords Texture segmentation, Texture, Image Processing, Image texture Analysis, Texture classification

I. INTRODUCTION

When solving problems in the area of texture based segmentation, the general way is to find a criterion which best describes the texture, calculate a figure of merit, and apply a threshold on it. Values above the selected threshold indicate the sought texture. Various authors [1]–[5] attempt to find meaningful criteria to selectively describe a particular texture. These features can get quite complex and in many cases it would also be possible to detect the texture by a combination of simple texture descriptors. However, the selection of the optimal descriptor is not always straight forward.

This work was motivated by an industrial project where scratches on coils lateral areas are to be detected. Since furthermore regions with other defect types yielding different textures should also be detected, a data mining approach seems to be reasonable since it can be used to select the optimal texture descriptors (also called features or attributes). In other words, image processing meets data mining.

This paper is structured as follows: In the next section the feature extraction of the image is discussed to provide the input data set for the data mining process. Subsequently the work3ow of the data mining process is described in Sec. III. This work3ow is tested on synthetic images and further on real images of coils with scratches where the performance of the work3ow is discussed (Sec. IV). Finally future improvements are suggested and an outlook is given.

A. Formalization of the problem

Let $\Omega\, C\, R^2$ be the domain of an image and $\{\Omega_1\}_{i=1,\dots N}$ be a partition of Ω into N (unknown) regions[1]. We assume that the pixels contained in the region Ω_1 are a Gauss-Markov process[2]. An other words, there exist (unknown) parameters $A, \in R^{n\times n}\, C_i \in R^{m_1\times n}$, covariance matrices $Q_1 \in R^{n\times n}$, $R_1 \in R^{m_1\times m_1}$, white, zero-mean Gaussian processes $\{v(t)\} \in R^n \cdot \{w_i(t)\} \in R^{m_1}$ and a process $\{x(t)\} \in R^n$ such that the pixels at each region $\Omega SUB1$ at each instant otf time are given by

$$\left\{ \begin{array}{l} x(t+1) = A_i x(t) + \sqrt{Q_i}\, v(t) \colon x(t_0) = x_i, 0 \\ y_i(t) = C_i x(t) + \sqrt{R_i}\, w_i(t) \end{array} \right\} \quad (1)$$

Given this generative model, one way to formalize the problem of segmenting a sequence of images is the following : Given a sequence of images $\{y(t) \in R^m, t = 1, \dots T)$ with two or more distinct regions $\Omega_l, l = 1, \dots N > 2$ that satisfy the model (1), estimate both the regions Ω_i and the model parameters of each region, namely the matrices A_i, C_i, the initial state $x_i 0$ and the covariance of the driving process Q_i.

B. Relation to prior work

The model for the spatio-temporal statistics of one region that we proposed in the previous section was first used in [5]. Similar statistical models were also used by others [21, 6, 24, 9]. For synthesis of spatio-temporal textures, statistical generative models can be replaced by procedural techniques such as [19, 25].

The analytical tools we use to infer the model parameters are borrowed from the literature of subspace system identification[15], and the discrepancy measure n] between different Models is inspired by[3].Since textures can be considered as a degenerate case of dynamic textures, our work relates to texture segmentation.

The proposed algorithm partition the image domain of a video sequence into regions with constant spatio-temporal statistics.

II. DYNAMIC TEXTURE LEARNING AND COMPARISON

If the regions Ω_i, $i = 1, \dots N$ were known, one would just be left with two problems: one is the learning of the model parameters, which we review in Sect. 2.1, the other is the computation of a discrepancy measure between different dynamic textures, which we discuss in Sect. 2.2.

A. Learning

It is well known [10] that a positive-definite covariance sequence with rational spectrum corresponds to an equiva-lence class of second-order stationary processes. It is then possible to choose as a representative of each class a Gauss-Markov model with the given covariance. In other words, for a given region Ω_i, we can assume that there exist a positive integer n, a process $\{x(t)\}$ with realizations in R^n (the "state") with initial condition $x(t_0)$), some matrices A_i and C_i and a symmetric positive semi-definite matrix

$\begin{bmatrix} Q_i & S_i \\ S_i^T & R_i \end{bmatrix} \geq 0$, where $S_i = E[w(t)\, v^T(t)]$, such that $\{y(t)\}$ is the output of model (1). Since we assume that the noise affecting the state $v(t)$ and the noise affecting the output $w(t)$ are independent, we have that $S_i = 0$.

The choice of matrices A_i, $C_i\, Q_i$, R_i is not unique, in the sense that there are infinitely many models that give rise to exactly the same measurement covariance sequence start-ing from suitable initial conditions[3]. In other words, any given process has *not* a unique model, but an *equivalence class* of models. In order to identify a unique model of the type (1) from a sample path $y(t)$, we choose a representative of each equivalence class as suggested in [5], i.e. we will make the following assumptions :$m_i > n$ and rank $(C_i) = n$ and choose a model realization that makes the columns of C_i orthonormal, i.e $C_i^T C_i = I_n$. This guarantees that the model corresponding to a given dataset is uniquely determined. This model corresponds to a canonical realization [7].

The problem of going from data to models can be formulated as follows: given measurements of a sample path of the process: $y(1) \dots y(T)$: $T >> n$, estimate A_i, $C_i\, Q_i$, a canonical model relization of the process $\{y(t)\}$. Ideally, we would want the maximum likelihood solution from the finite sample, this is

$$A_i,\ C_i,\ Q_i,\ x_i,\ o] \tag{2}$$

$$= \underset{\substack{A_i \cdot C_1 \\ Q_1 \cdot x_1 \cdot 0}}{\operatorname{argmax}} \log p\,(y(1) \dots y(T) \mid A_i,\ C_i,\ Q_i \,..\, x_i 0)$$

Notice that, as we said in Section 1.1, we do not model the covariance of the measurement noise R_i, since that carries no information on the underlying process. The asymptotically efficient solution for the estimation problem (2), as $T \to \infty$, can be found in [23], while accurate description of its implementation, named N4SID, can be found in [15]. In practice, for computational efficiency, we settle for the suboptimal solution described in [5].

B. Discrepancy between dynamic textures

Assuming that the parameters A_i, C_i, Q_i, $x_{i,0}$ have been inferred for each region, in order to set the stage for a segmentation procedure, one has to define a discrepancy mea-sure among regions. The difficulty in doing so is that each region is described not only by the parameters above, but by an equivalence class of such parameters, obtained by changes of basis of the state-space $\{x(t)\}$ in model (1). Therefore, a suitable discrepancy measure has to compare not the parameters directly, but their equivalence classes.

One technique for doing so has been recently proposed in [3]. It consists of building infinite observability matrices, whose columns span the vector space generated by the measurements $y(t)$ of the model (1), and that represent the high-dimensional subspace of the infinite-dimensional space of all possible measurements. Then one can compute the geo-metric angles between such subspaces through their embed-ding.

More formally, let $A \in R^{m \times p}$ and $B \in R^{m \times q}$ be two matrices with full column rank, and suppose that $p \geq q$. The q principal angles $\theta_k \in 0, \frac{\pi}{2}$ between range (A) and range (B) are recursively defined for $k = 1, 2, \ldots q$ as

$$\cos \theta_1 = \max_{\substack{x \in Rp \\ y \in Rp}} \frac{|x^T A^T B_y|}{||Ax||_2 \, ||By||_2} \tag{3}$$

$$= \frac{|x_1^T A^T B_{y_1}|}{||Ax_1||_2 \, ||By_1||_2}$$

$$\cos \theta_k = \max_{\substack{x \in Rp \\ y \in 2p}} \frac{|x^T AtupT B_y|}{||A_x||_2 \, ||B_y||_2}$$

$$= \frac{|x_k^T A^T B_{yk}|}{||Ax_k||_2 \, ||By_k||_2}, \text{ for } k = 2 \ldots q$$

subject to $x_i^T A^T Ax = 0$ and $y_i^T B^T By = 0$.

for $i = 1, 2 \ldots k - 1$

subject to $x_i^T A^T Ax = 0$ and $y_i^T B^T By = 0$, for $i = 1, 2, \ldots k - 1$

Now, let M_1 and M_2 be two models of the type (1), with the same output dimensionality, which are characterized in state space terms by their system matrices A_1 and A_2 and output matrix C_1 and C_2 respectively. Their infinite observability matrices O_i, for $i = 1, 2$, are defined as

$$O_i = [C_i^T A_i^T C_i^T \ldots (A_i^T)^n C_i^T \ldots] T \in R^{x \times n} \tag{4}$$

and the principal angles between the ranges of O_1 and O_2 are referred to as *subspace angles*. Their computation can be carried out in closed form, and entails the computation of the eigen values of the solution of a discrete-time Lyapunov equation [3].

While more than one distance for single-input single-output (SISO) linear dynamical systems have been defined based on subspace angles [3, 12], the extension to the multiple-input multiple-output (MIMO) case is not trivial given the lack of the concept of the inverse of a MIMO system. However, it has been shown in [18] that subspace angles between infinite observability matrices have very high discriminative power under the hypothesis of stability and observability of the compared systems. With this in mind, we measure the discrepancy between different spatio-temporal statistics associated to different models by com-paring either the set of subspace angles or their combination via Martin's distance [3] defined as

$$d_2 M(M_1 \cdot M_2) = ln \prod_{k=1}^{n} \frac{1}{\cos^2 Q_k} \tag{5}$$

III. DYNAMIC TEXTURE SEGMENTATION

In Sect. 2.1 we have seen that, if the boundaries of each region were known, one could easily estimate a simple model of the spatio-temporal statistics within each region. Unfortunately, in general one does not know the boundaries, which are instead part of the inference process. If the dynamic texture associated with each pixel were known, then one could easily determine the regions by thresholding or by other grouping or segmentation techniques. However, a dynamic texture associated with a certain pixel ξ, as defined in equation (1), depends on the whole region

Ω_i containing the pixel ξ. Therefore, we have a classic "chicken-and-egg" problem: If we knew the regions, we could easily identify the dynamical models and if we knew the dynamical models we could easily segment the regions. Unfortunately, we know neither.

Rather than seeking for an estimate of the regions and the model parameters in one shot, we can instead adopt a two-stage algorithm to circumvent the model complexity issue: We first associate a local *signature* to each pixel, by integrating visual information on a fixed spatial neighborhood of that pixel; then we group together pixels with similar signatures in a region-based segmentation approach. The signatures are computed from the subspace angles relative to a reference model, following the ideas outlined in previous sections. We describe this simple and yet effective approach in the following subsections.

A. A geometric approach

We start by considering the neighborhoods $B(\xi)\, C\Omega$ around each pixel $\xi \in \Omega$. We then associate to each pixel location ξ the dynamics of the spatio-temporal region by computing (ξ) from $A(\xi), C(\xi) = N4SID\, y(\xi, t)\, \xi\, B(\xi), t = 1, \ldots T.$ For each pixel ξ we generate a local spatio-temporal *signature* given by the cosines of the sub-space angles $\{\theta_j(\xi)\}$ between $O(\xi)$ and a reference model,

$O(\xi_n)$:

$$i(\xi) = (\cos\theta_1(\xi) \ldots \cos\theta_n(\xi)) \tag{6}$$

We call this approach "geometric" since the signatures are constructed using subspace angles, rather than responses of banks of filter as is more common in static texture segmentation.

With the above representation, the problem of dynamic texture segmentation can be formulated as one of grouping regions of similar spatio-temporal signatures. We propose to perform this grouping by reverting to the Mumford-Shah functional [13]. A segmentation of the image plane Ω into a set of pairwise disjoint regions Ω_i of constant signature $s_i \in R''$ is obtained by minimizing the cost functional

$$E(\Gamma, \{s_i\}) = \sum_i \int_{\Omega_i} (s(\xi) - s_i)^2\, d\, si + v[\Gamma], \tag{7}$$

simultaneously with respect to the region descriptors $\{s_i\}$ modeling the average signature of each region and with respect to the boundary Ã separating these regions (an appropriate representation of which will be introduced in the next section). The first term in the functional (7) aims at maximizing the homogeneity with respect to the signatures in each region Ω_i, whereas the second term aims at minimizing the length $|\Gamma|$ of the separating boundary.

B. A level set formulation

In the following, we will restrict the class of possible solutions of the proposed variational problem to two-phase solutions, i.e. solutions in which each pixel is associated with one of two dynamic texture models. All results do, however, extend to the case of multiple phases. For the implementation of the boundary Ã in the functional (7) we revert to the implicit level set based representation proposed by Chan, Sandberg and Vese [2, 22].

Compared to explicit contour representations, the level set based representations [14] have several favorable prop-erties. Firstly, they do not restrict the topology of the evolv-ing boundary, thereby facilitating splitting and merging dur-ing the evolution. And secondly, one does not need to take care of a regridding of control or marker points.

Let the boundary Γ in (7) be given by the zero level set of a function $\varphi : \Omega \to R$:

$$\Gamma = \{\xi \in \Omega \mid \phi\ (\xi) = \} \tag{8}$$

With the Heaviside function

$$H(\phi) = \begin{Bmatrix} 1 & \text{if } \phi \geq 0 \\ 0 & \text{if } \phi < 0 \end{Bmatrix} \tag{9}$$

the functional (7) can be replaced by a functional on the level set function ϕ:

$$E(\phi, \{s_i\}) = \int_{\Omega} (s(\xi) - s_1)^2 H(\phi)\, d\xi$$

$$+ \int_{\Omega} (s(\xi) - s_2)^2 (1 - H(\phi)\, d\xi$$

$$+ p[\Gamma] \tag{10}$$

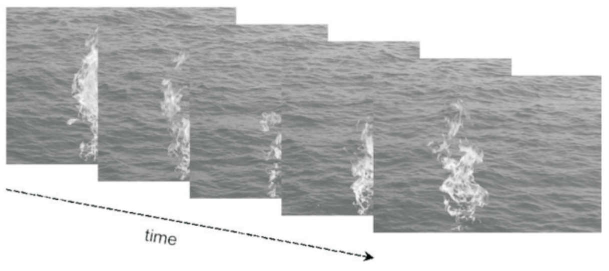


Fig. 1. Example of a composition of dynamic textures: fire on ocean waves. The time evolution of the sequence is rendered by overlapping a few snap-shots corresponding to different image frames.

IV. FEATURE EXTRACTION

In image processing, usually presegmented objects are classified by a previously trained classifier. E.g. damaged goods are recognized by a classifier on the production line, where the features are calculated for the whole presegmented object. Since in this paper texture based segmentation will be investigated, the number of elements to classify is equal to the number of the pixels N_p in the image, which is much larger than the number of the presegmented objects N_o.

A. Subsampling the input data

To detect pixels representing textures algorithmically the memory requirement is $N_p N_f S_f$, where N_f is the number of the features and S_f is the memory needed for each feature. To save memory, only a subset of pixels is selected out of the image.

Since the texture doesn't change very abruptly in the spatial domain and usually is distributed over a bigger region of the image, selecting only a few percent of all pixels N_p is sufficient enough to train the classifier properly. Further, when the neighborhood size s_{SH} is large enough, two adjacent pixels will have similar features.

$$N_{\text{all}} = \frac{slx\, sly}{NHx\, sNHy}\, o \qquad (1)$$

with

slx image size in *x* - direction

sNHx neighborhood size in *x* - direction

sly image size *y* - direction

sNHy neighborhood size in *y* - direction

o spatial overlap of distinct neighborhoods

Depending on the parameter s_{SH} used for the feature calculation a reasonable number of pixels analyzed is:10000 is choosen. Depending on the problem andthe requirements on memory and time complexity, these parameters can be adapted.

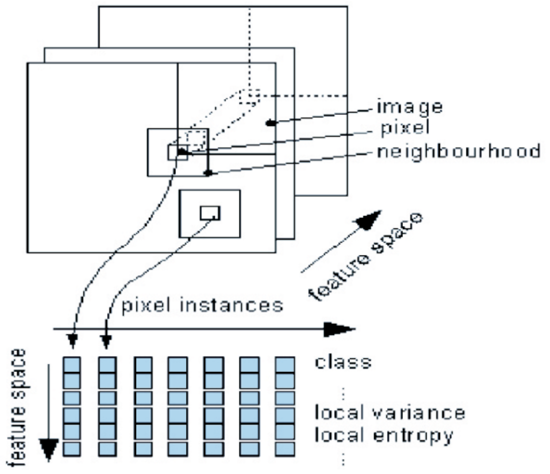


Fig. 2. only a subset of all pixels are selected from the image for the data mining process

In fig. 2 the image is sub sampled and a feature vector is calculated for each selected pixel and its neighbourhood.The selected pixels are called instance, because their spatial position is not used for data mining process.

B. Feature calculation

For each instance its feature vector is calculated. This feature vector consists of various features

For a 1000 $\times$ 1000 pixel image with a typical $s_{NH} =$ (15; 15) an overlap of 2.25 is achieved, when $N_{\text{all}} =$

Which potentially might be able to separate the Specific texture? In general these features require a local neighborhood to describe the texture and can be divided into

(a) Scalar features (e.g. variance, skewness, uniformity, entropy, local image contrast, . . .) in the form of $I(x; y) = f(I(x; y); S_{NHx}; S_{NHy};::)$;

(b) and matrix features (e. g. FFT, co–occurrence matrix, structure tensors, . . .).

$M_f(x, y, u, v) = f(I(x, y), s_{NHx}, s_{NHy} \ldots)$

which must be mapped into a scalar feature

$I_f(x, y) = f(M_f(x, y, u, v) \ldots)$ by calculating energy, constrast, etc. from the matrix feature. (u, v) are additional indices of the matrix features, depending on the neighborhood size.

For example when calculating the energy with in the low spatial frequency bands, the local matrix feature is given by

$$M_{f1}(x, y, u, v) = FFT\{I(x, y), s_{NHx}, s_{NHy}\} \tag{2}$$

and the associated scalar feature the energy with the low frequencies becomes:

$$I_f(x, y) = \Sigma_{(u, v) \in \Omega} M_{f1}(x, y, u, v)^2 \tag{3}$$

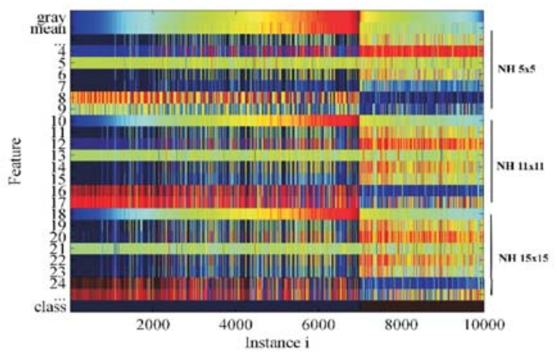


Fig. 3 Features are calculated for selected pixels (instances) of the trainings data set (see Fig. 5) and sorted by the class value and the first feature.

In Fig. 3 the feature values of 10000 instances, sorted by the class, are shown. The index of the instance is plotted on the abscissa. The ordinate encodes the different features calculated for each instance in color. The class feature (at the bottom of the image) has to be generated interactively by the user for real images of the coil (see Fig. 8(b)). On the right side there are 3000 instances indicating the texture of interest (class 1). The first 7000 instances do not belong to the sought texture. Since the instances are additionally sorted along the abscissa by a second feature, in this case the grayscale value, it is obvious that some of the features are very similar.

All features are normalized to the range 0 (blue) to 1 (red) to make them comparable. Since a varying size of the neighborhood, might accent specific textures unequal, all features are calculated for varying neighborhood sizes 5, 11, 15 (upper, middle and lower third blocks in the image).

When analyzing Fig. 3 to select features for the classifier, a human might discard e.g. feature number 7 because it has only less dependency on the class. Feature 4 and 8 might be good candidates to separate the classes. Regardless of that, the feature selection for the classifier should be done in the data mining process automatically, shown in the next section.

V. MACHINE LEARNING AND DATA MINING

Machine learning and data mining [10] are topics that have a wide application range. Classifying image and audio data is well understood and widely used (e. g. [11]–[13], Musical Audio–Mining[1]).

Basically this process can be split into two main parts — the **training** and the **test** of the classifier. Therefore a manual classification of the image texture must be available. It is used to determine the quality and error rate of the classifier. If the error rate of the generated classifier is satisfyingly small, the classifier can be applied to images with unknown texture. For the implementation of this work the open source java data mining software Weka [10][2] is used together with MATLAB7.

1. Training of the classifier: Using the previous extracted features (Sec. II) the selection of the best features is done using Wekas attribute ranking/selector algorithms (In Weka features are denoted as attributes). Some popular attribute selector algorithms are presented in the following list:

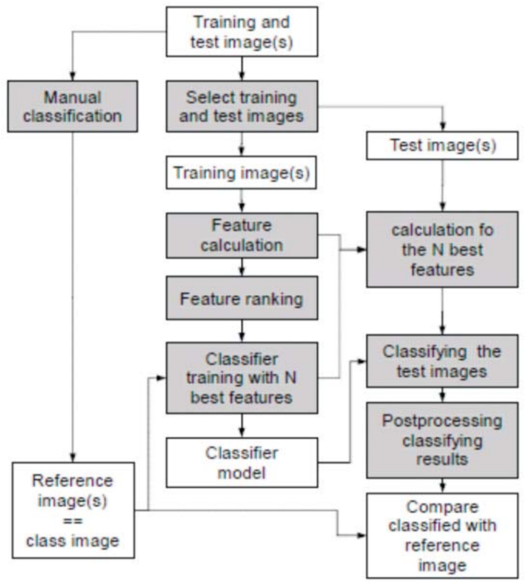
CfsSubsetEval: Evaluates the worth of a subset of at-tributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.

InfoGainAttributeEval: Evaluates the worth of an at-tribute by measuring the information gain with respect to the class.

GainRatioAttributeEval: Evaluates the worth of an at-tribute by measuring the gain ratio with respect to the class.

After selecting the attributes for the data mining process, a classifier must be trained. Weka offers a lot of different categories of classifiers:

- **bayers:** Classifiers based on the bayesian decision theory.
- **functions:** Classifiers based on regression.



Training and test image(s)
Manual classification
Select training and test images
Test image(s)
Training image(s)
Feature calculation
calculation fo the N best features
Feature ranking
Classifier training with N best features
Classifying the test images
Postprocessing classifying results
Classifier model
Reference image(s) == class image
Compare classified with reference image

Fig. 4. Texture segmentation using data mining algorithms

- **lazy:** Classifiers with less computation effort but generally high memory requirements, like e.g. nearest neighbors classifiers, which don't try to infer a density function in closed form.
- **meta:** Meta-classifiers combine several classifiers in different ways like voting, stacking or boosting, . . .
- **trees:** Classifiers building up a decision tree using the selected attributes.
- **rules:** Classifiers defining rules (similar to the trees; most rules can be transformed into trees and vice versa).

Detailed information about the classifiers and their implementation can be found at Wekas Website[4] and in the Weka documentation.

2. Testing of the classifier: When the classifier is success-fully established, it is tested with a set of test images. For each pixel or a set of pixels of the image the selected features are calculated. Using the classifier a prediction of the class can be evaluated. The performance of the classifier can be determined by some indices.

The prediction accuracy PA

$$PA = \frac{TP + TN}{P + N} \tag{4}$$

gives the ratio of the correct predictions and the prediction error PE

$$PE = \frac{FP + FN}{P + N} = 1 - PA \tag{5}$$

gives the ratio of the false predictions. The true positive ratio $TPR = \frac{TP}{P}$ gives the prediction accuracy for the positive.

class and the true negative ratio $TNR = TN_{/N}$

prediction accuracy for the negative class.

$P = TP + FN$ . . . total *nr.* of instances with **positive** class

$N = NP + TN$ . . . total nr. of instances with

negative class

TP . . . number of **true** predicted **positive**

class

FP . . . number of **false** predicted **positive**

class

TN . . . number of **true** predicted **negative**

class

FN . . . number of **false** predicted **negative**

class

In many cases it is possible to increase of the prediction accuracy by smoothing the raw prediction data, hence false predictions of a well trained classifier are less than true predictions.

VI. EXAMPLES

In this section the previously described workflow will be discussed based on examples of synthetic images and real images of a coil metal sheet layer.

After selecting the attributes for the data mining process, a classifier must be trained. Weka offers a lot of different categories of classifiers:

- **bayers:** Classifiers based on the bayesian decision theory.
- **functions:** Classifiers based on regression.

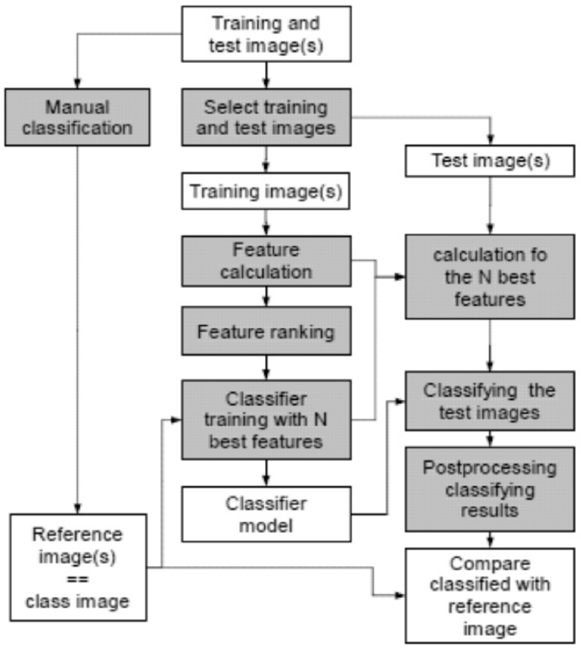

Training and test image(s)
Manual classification
Select training and test images
Test image(s)
Training image(s)
Feature calculation
calculation fo the N best features
Feature ranking
Classifying the test images
Classifier training with N best features
Postprocessing classifying results
Classifier model
Reference image(s) == class image
Compare classified with reference image

Fig. 4. Texture segmentation using data mining algorithms

- **lazy:** Classifiers with less computation effort but generally high memory requirements, like e.g. nearest neighbors classifiers, which don't try to infer a density function in closed form.
- **meta:** Meta-classifiers combine several classifiers in different ways like voting, stacking or boosting, . . .
- **trees:** Classifiers building up a decision tree using the selected attributes.
- **rules:** Classifiers defining rules (similar to the trees; most rules can be transformed into trees and vice versa).

Detailed information about the classifiers and their implementation can be found at Wekas Website[4] and in the Weka documentation.

2. Testing of the classifier: When the classifier is success-fully established, it is tested with a set of test images. For each pixel or a set of pixels of the image the selected features are calculated. Using the classifier a prediction of the class can be evaluated. The performance of the classifier can be determined by some indices.

The prediction accuracy PA

$$PA = \frac{TP + TN}{P + N} \tag{4}$$

gives the ratio of the correct predictions and the prediction error PE

$$PE = \frac{FP + FN}{P + N} = 1 - PA \tag{5}$$

gives the ratio of the false predictions. The true positive ratio $TPR = \frac{TP}{P}$ gives the prediction accuracy for the positive.

class and the true negative ratio $TNR = TN_{/N}$

prediction accuracy for the negative class.

$P = TP + FN$ . . . total *nr.* of instances with **positive** class

$N = NP + TN$ . . . total nr. of instances with

negative class

TP . . . number of **true** predicted **positive**

class

FP . . . number of **false** predicted **positive**

class

TN . . . number of **true** predicted **negative**

class

FN . . . number of **false** predicted **negative**

class

In many cases it is possible to increase of the prediction accuracy by smoothing the raw prediction data, hence false predictions of a well trained classifier are less than true predictions.

A. Synthetic Images

Using synthetic images provides more options to vary shape and texture of the test image than real images. Border conditions are well defined and textures to detect can be placed properly. Fig. 5 (a) shows an image with areas of varying textures. The background changes only slowly with position to simulate an illumination gradient always seen in real coil images. Various areas (circles and rhombi), filled with various textures, are multiplicatively linked to the background. Figure 5 (b) shows the specific class of texture, that is to be predicted.

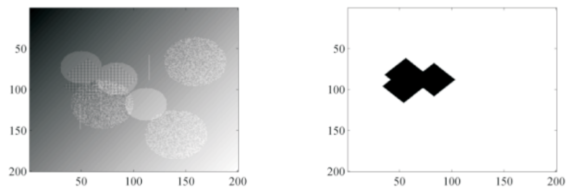


Fig. (a) Synthetic image (b) Synthetic class image
Fig. 5 Synthetic image and class to train the classifier.

The selection of the right features is done by different ranking algorithms(Classi-fierSubsetEval, InfoGainAttributeEval, PrincipalComponents, etc.), provided by Weka. Figure 6 shows the ranking result using the Info Gain Attribute Eval algorithm. To select the right features, attribute selectors can be applied to the training data set. The feature ranking algorithm is one possible method, but definitely not the best, since e.g. the fft2–features of various sizes in Fig. 6 have a high cross correlation. Discarding strongly correlated features helps to reduce features to be calculated and renders the classifier more robust. A good feature selector that selects only a few but significant features, avoids an over trained classifier

Which predicts very good results on the training data set and performs worse on the test data set?

Using the features with the largest ranking score (e.g. using a threshold of 0.45 for the InfoGain), a classifier (in this example a tree classifier) is trained. Combining various attribute selectors may help to select the most significant features.

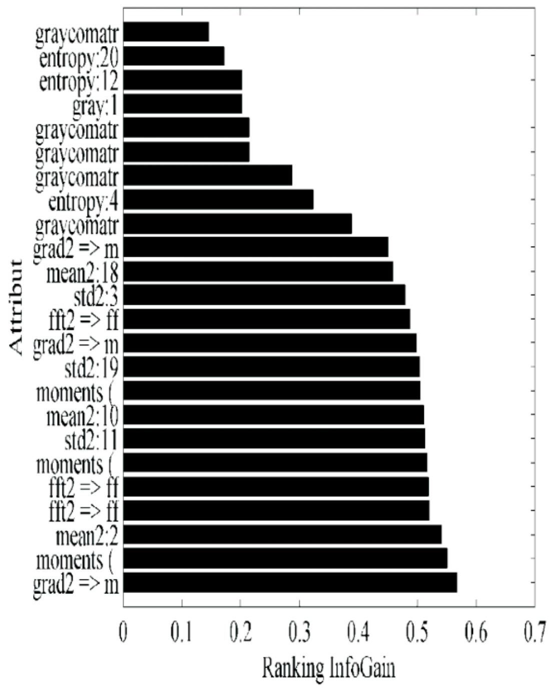
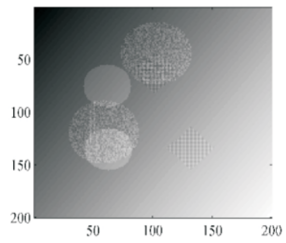
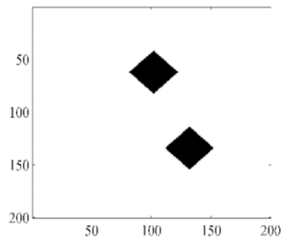
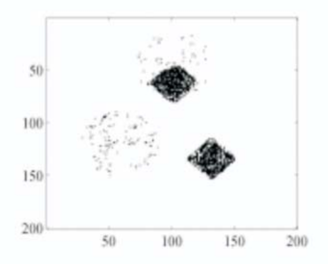
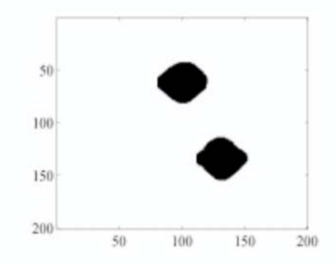


Fig. 6. Features with low ranking are discarded.

(a) Synthetic image (b) Synthetic class image

(c) Synthetic prediction image (d) Synthetic smoothed image
Fig. 7. Test data (image nr.5) set with class and prediction

Figure 7 shows a test data set a tree classifier was applied to. The searched texture is difficult to detect for a human observation, especially where the noise is very high. However, a human takes care of the a priori knowledge of the texture shape searched for, which is known to be a rhombus for the test data set.

The statistics of the classifier, tested on some other test data sets is shown in Tab I. Although PA is very large, some test sets have a very low TPR, even zero percent for test image 7. This indicates that the classifier covers all test set possibilities. However the classifier was trained with only two test images (the first training image is shown in Fig. 5), so the results are already very good. The subsequent smoothing of the raw predictions increases the accuracy slightly, but also some true predictions are removed (e.g. test image 3 and 4 where PA increases and TPR decreases). Since the sought texture should represent a scratch, that is rather small, PA **and** TPR should be large.

Table 1. Eerror Statics for the Predictions of the Synthetic Data Set

raw predictions

Image nr.	*TP*	*TN*	*FP*	*FN*	*PA*	*PE*	*TPR*	*TNR*
3	1843	6646	354	1157	84.89	15.11	61.43	94.94
4	2282	6725	275	718	90.07	9.93	76.07	96.07
5	2608	6775	225	392	93.83	6.17	86.93	96.79
6	2859	6806	194	141	96.65	30.24	0.00	99.66
8	2904	6908	92	96	98.12	1.88	96.80	98.69

smoothed predictions

Image nr.	*TP*	*TN*	*FP*	*FN*	*PA*	*PE*	*TPR*	*TNR*
3	1653	35776	91	2480	93.57	6.43	40.00	99.75
4	2720	36040	251	989	96.90	3.10	73.54	99.31
5	1675	37749	569	7	98.56	1.44	99.58	98.52
6	2506	36765	712	17	98.18	1.82	99.33	98.10
7	0	39159	0	841	97.90	2.10	0.00	100.00
8	841	38716	443	0	98.89	1.11	100.00	98.87

Table 2. Error Statics for the Predictions of the Coil Data Set

raw predictions

Image nr.	*TP*	*TN*	*FP*	*FN*	*PA*	*PE*	*TPR*	*TNR*
3	27	5144	2	1856	73.57	26.43	93.10	73.49
4	25	6582	4	418	94.00	6.00	86.21	94.03
5	28	469	1	6531	7.07	92.93	6.70	
6	23	5076	6	1924	72.54	27.46	79.31	72.51

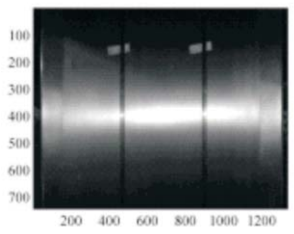
smoothed predictions

Image nr.	*TP*	*TN*	*FP*	*FN*	*PA*	*PE*	*TPR*	*TNR*
3	3903	1385351	6262	1336	99.45	0.55	38.32	99.90
4	155	1146694	1	146	99.99	0.01	99.36	99.99
5	1797	991986	1	1688	99.83	0.17	99.94	99.83
6	1404	581200	1007	1380	99.59	0.41	58.23	99.76

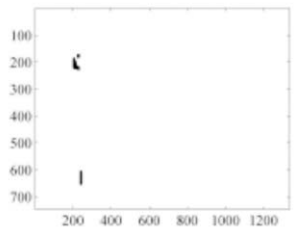
B. Real Images

In this section the work3ow is tested on eight images of the real coils lateral area. The first two images are used to select the features and train the classifier. Images 3 to 6 (see Tab. II) are used to test the classifier and their performance is shown in Tab II. A typical image set is shown in Fig. 8. The classification of the images (Fig. 8(b)) was done manually, so especially in boundary regions where the texture is not strongly distinct, the classification might be erroneous. Errors at the manual classification reduce the performance of the trained classifier.

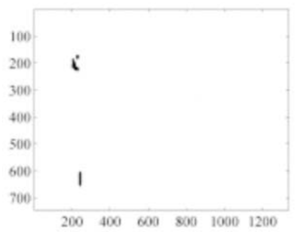
Since it is more important to detect scratches on the metal sheet surface (positive class), the true positive rate *TPR* is at least as important than the prediction accuracy *PA*.



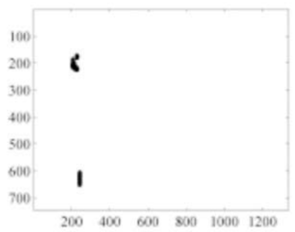
(a) Coil image



(b) Coil class image



(c) Coil prediction image



(d) Coil prediction image (smoothed)

Fig. 8 Classifying scratches on the coils lateral area (image nr. 5)

The statistics (Tab. II) show that for real images it is more difficult to built an accurate classifier. For the problem of scratch detection, the area with a scratch texture is very small. To detect the scratches (class 1 or positive) accurately the *TPR* and *PA* must be large. Because if there are only less scratches on the coil (e.g. 1%) even the simplest classifier which always predicts no scratches (class 0) will still have a large *PA* and *TNR*, but *TPR* will be zero. Consequently the classifier must be designed to detect the scratches more accurate than the background. This can be done be introducing a cost function or using more instances of the positive class if the sub sampling factor was large.

Another reason for the moderate performance is, that the image resolution is very low, so the sought scratch textures cover only small regions (see Fig. 8(b)). Further only few images were available for the training and testing of the classifier.

To increase the *TPR* more representative training and test images are necessary. Indeed the training and the test data set is not the only potential improvement, in some cases simple features will not be sufficient, so the user still has to think about features which describe the sought texture best.

VII. CONCLUSION

We propose the usage of data mining algorithms for texture detection leading to satisfying results. The only drawback is the increased numerical complexity, because a large set of features must be calculated first. Although if the feature space is reduced by the feature ranking and selection process there still is an appreciable amount of calculations to be done for the final feature classification.

The main advantage is that only little a priory knowledge is necessary about the features that describe the texture. The feature and threshold selection is done automatically when the appropriate classifier is build.

Although a lot of steps must be covered where different parameters can be optimized. Selecting the right training and test data set, features and classifier should not be underestimated. Especially when selecting the features a pre selection based on physical properties will be reasonable. But often features describing a texture by a physical model are not available, so at this point the loop closes at the frequent dealt problem: Find an feature that describes a specific texture best.

REFERENCES

[1] Duvernoy J., 1985 “Optical–digital processing of directional terrain textures invariant under translation, rotation, and change of scale,” in Applied Optics, vol. 23, no. 6, pp. 286–837.

[2] Gonzales R.C. and Woods R.E., 2002“Digital Image Processing”, 2nd Edition. Prentice Hall.

[3] Sezer O.G., Ertuzun A. and Ercil A., 2004 “Independent component analysis for texture defect detection,” Pattern Recognition and Image Analysis, vol. 14, no. 2, pp. 303–307.

[4] Sobral J., 2005 “Optimised filters for texture defect detection,” in International Conference on Image Processing (ICIP), pp. 3165–3168.

[5] Sebe N., Cohen I., Grag A. and Huang T., 2005 “Machine Learning in Computer Vision”. Springer.

[6] Neumann B., 2005 Bildverarbeitung fur Einsteiger. Springer–Verlag Heidel-berg.

[7] Davies E. R., 2005 Machine Vision — Theory, Algorithms, Practicalities, 3rd ed. Elsevier.

[8] Jain A.K., 1989 Fundamentals of Digital Image Processing. Prentice Hall.

[9] Witten I.H. and Frank E., 2005, Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed. Morgan Kaufmann, [Online]. Available:http://www.cs.waikato.ac.nz/ml/ weka/book.html

[10] Ahammer H., Krop J.M., Hackl C. and Sedivy R., 2008 “Image statistics and data mining of anal intraepithelial neoplasia,” Pattern Recognition Letters, vol. 29, no. 16, pp. 2189–2196, [Online]. Available: http://www.sciencedirect.com/ science/article/ B6V15-4T8JX8H-cf3a4040937c289e77c5109479

[11] Mazurkiewicz A. and Krawczyk H., 2002 “A parallel environment for image data mining,” Parallel Computing in Electrical Engineering, International Conference on Electrical Engineering, pp. 429–434, [Online]. Available: http://www2.computer.org/ portal/web/csdl/ doi/10.1109/PCEE.2002.1115330.