Using pattern recognition to automatically localize reflection hyperbolas in data from ground penetrating radar

Christian Maas, Jörg Schmalzl *

Institut für Geophysik, Universität Münster, Corrensstr. 24, D-48149 Münster, Germany

ARTICLE INFO

Article history:
Received 5 November 2012
Received in revised form
4 March 2013
Accepted 15 April 2013
Available online 24 April 2013

Keywords:
Reflection hyperbola
Haartraining
OpenCV
GPR
AdaBoost
Viola-Jones

ABSTRACT

Ground Penetrating Radar (GPR) is used for the localization of supply lines, land mines, pipes and many other buried objects. These objects can be recognized in the recorded data as reflection hyperbolas with a typical shape depending on depth and material of the object and the surrounding material. To obtain the parameters, the shape of the hyperbola has to be fitted. In the last years several methods were developed to automate this task during post-processing. In this paper we show another approach for the automated localization of reflection hyperbolas in GPR data by solving a pattern recognition problem in grayscale images. In contrast to other methods our detection program is also able to immediately mark potential objects in real-time. For this task we use a version of the Viola-Jones learning algorithm, which is part of the open source library “OpenCV”. This algorithm was initially developed for face recognition, but can be adapted to any other simple shape. In our program it is used to narrow down the location of reflection hyperbolas to certain areas in the GPR data. In order to extract the exact location and the velocity of the hyperbolas we apply a simple Hough Transform for hyperbolas. Because the Viola-Jones Algorithm reduces the input for the computational expensive Hough Transform dramatically the detection system can also be implemented on normal field computers, so on-site application is possible. The developed detection system shows promising results and detection rates in unprocessed radargrams. In order to improve the detection results and apply the program to noisy radar images more data of different GPR systems as input for the learning algorithm is necessary.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

The Ground Penetrating Radar (GPR) is a geophysical method used for the investigation of the surface-near underground. It is commonly utilized for the localization of objects like pipes, land mines or other buried objects. Many of these targets produce a characteristic hyperbolic reflection in the recorded data. With GPR large areas can be measured very efficiently and fast, but the data-processing and the localization of buried objects is very time-consuming. To extract the exact position and the material of the target every single radargram has to be processed, analyzed and all hyperbolas have to be located. Kaneko (1991), Al-Nuaimy et al. (2000), Shihab et al. (2002), Cui et al. (2010) and Chen and Cohn (2010) solve this problem by putting it down to a pattern recognition process in grayscale images. They use different methods of image analysis in combination with learning algorithms in order to dramatically reduce the expenditure of time for data-processing. It would be very desirable to search and fit reflection hyperbolas already in the field. Many of the above methods are unsuitable for on-site application because they are computational demanding or need processed data. This paper applies another new method, which can also be implemented on field computers and is able to locate hyperbolas in unprocessed GPR data on-site in real-time. Our approach is based on machine learning. In a first step the rough positions of the hyperbolic reflections are estimated. For this task we use an implementation of the Viola-Jones Algorithm (Viola and Jones, 2001), which is part of the “Open Source Computer Vision Library” (OpenCV). This algorithm was initially developed for face recognition, but can be easily adapted to any other simple shape. Once the algorithm has learned to recognize the shape of reflection hyperbolas they can be detected in real-time in unprocessed radargrams. In literature the Viola-Jones Algorithm is already utilized for real time detection of various objects (Viola and Jones, 2004; Trep钓ow and Zell, 2004; Barczak and Dadgostar, 2005; Dailey and Bo, in press).

After the Viola–Jones Algorithm has narrowed down the position of the hyperbolic reflections to certain areas, the position and depth of all detected hyperbolas, as well as the resulting velocities of the target are extracted by using a simple Hough Transform for hyperbolas (Duda and Hart, 1972). Because the Viola-Jones Algorithm has already
2. Viola–Jones implementation in OpenCV

The “Open Source Computer Vision Library” (OpenCV) is written in C/C++ and comes with many features for computer vision and image processing. It was first published by Intel in 1999, from 2008 to 2012 it was further developed by Willow Garage, since 2012 it is supported by the company Itseez.

The OpenCV-library contains a variant of the Viola–Jones Face-Detector (Viola and Jones, 2001) which was extended by Lienhart and Maydt (2002) and is called “Haartraining”. This algorithm belongs to the supervised learning algorithms with strictly separated learning and detection processes. In this section we want to present the basic concepts of the Viola–Jones Algorithm.

The basic idea of this algorithm is to learn during the so-called training to identify an object by analyzing many positive (containing object) and negative (not containing object) sample images. When training is completed the output classifier can be used for object recognition.

During detection the algorithm divides the input images into many sub-images by moving a search window at multiple scales over it. Then each sub-window is classified as object or no-object. For this task haar-like features are used.

2.1. Haar-like features

Viola and Jones (2001) use a combination of simple haar-wavelet-like features shown in Fig. 1 to analyze each sub-image and classify it in two classes: object or no-object. Originally Viola and Jones introduced the features 1a, 1b and 2a-c, Lienhart and Maydt (2002) extended this set to 45 rotated features (1c, 1d, 2e-h, 3b).

The advantage of using features instead of raw pixel values is the possibility to compensate little variations in the appearance of the object which makes classification easier. Furthermore they can be calculated very fast by utilizing the so-called “integral image” (see Section 2.2). In addition to that they can be scaled independently in vertical or horizontal direction to build up a linear independent and overcomplete basis of usable features for the analysis of sub-windows.

The computation of these features is based on the comparison of pixel intensities. In the case of two rectangles the pixel sum of the black area is subtracted from the pixel sum of the white area (Fig. 1). If the feature consists of three rectangles the sum of the two white areas is added up and then subtracted from the sum of the inner black area.

We assume that the top left corner of a rectangle (r) is located at (x, y) and has the width b, the height h and the orientation α∈[0°, 45°), so it is described by r=(x, y, b, h, α). If the pixel sum of a rectangle is denoted by RecSum, the features f have the following form:

\[ f = \omega_1 \cdot \text{RecSum}(r_1) + \omega_2 \cdot \text{RecSum}(r_2). \]

(1)

where the weights \( \omega_1, \omega_2 \in \mathbb{R} \) consider differences in area size of the different rectangles. To compute RecSum very quickly Viola and Jones (2001) introduce the integral image.

2.2. Integral image

In order to evaluate the features of Section 2.1 many pixel sums have to be computed. This can be done very fast and in constant time for any rectangle size by using the integral image approach, which is derived from the “Summed Area Table” (Crow, 1984). The integral image \( \text{int}(x, y) \) of the original image therefore has to be calculated only once. Then every feature can be determined by four table lookups, regardless of scaling and position. \( \text{int}(x, y) \) is defined after Viola and Jones (2001) as pixel sum of the rectangle with the top left corner at \((0, 0)\) and the bottom right corner at \((x, y)\):

\[ \text{int}(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} \text{int}(i, j). \]

(2)

It can be derived with the following expression in one pass from left to right and top to bottom over all pixels of the original image (Lienhart and Maydt, 2002):

\[ \text{int}(x, y) = \text{int}(x-1, y) + \text{int}(x, y-1) - \text{int}(x-1, y-1) - \text{int}(0, 0). \]

(3)

Based on Eqs. (2) and (3) the pixel sum of a rectangle with \( r=(x, y, b, h, α=0°) \), for example rectangle D in Fig. 2, can be determined with four table lookups

\[ \text{RecSum}(r) = \text{int}(x, y) - \text{int}(x, y - b) - \text{int}(x - b, y) + \text{int}(x - b, y - b). \]

(4)

For rotated features the integral image can be computed analogously. In this case two passes over the original image are needed (see Lienhart and Maydt, 2002).

2.3. Adaboost

In general Adaboost is used to improve the performance of simple learning algorithms (Freund, 1996). The Adaboost learning algorithm is based on a simple principle: Instead of using one single hypothesis to make a decision, the opinion of several experts is taken into account. The final decision is made through majority decision of all expert opinions.

The Viola–Jones Algorithm uses a variant of Adaboost to select a small set of suitable out of the enormous pool of available features and also to train a “strong classifier” (Freund and Schapire, 1996), which decides during detection whether a sub-window contains the searched object or not. This classifier itself consists of a linear combination of “weak classifiers”. They are called weak because in themselves they have a large error. To examine the input images each “weak classifier” utilizes haar-like features (see Section 2.1) which are selected by Adaboost.
For the training process an enormous number of positive and negative sample images is necessary. In every boosting round Adaboost chooses a low number of good classification functions (weak classifiers) that optimally describe the object and separate the positive and negative sample images. To minimize the misclassified samples the weak classifiers are repeatedly applied on different input images. Misclassified samples get a higher, and correctly classified examples a lower weight for the next boosting round. This process is repeated several times. Therefore the boosting algorithm concentrates on samples which are harder to learn.

Finally Adaboost forms a final strong classifier out of the trained weak classifiers. In the following the Adaboost algorithm after (Viola and Jones, 2001) is described.

- Given example images \( (x_1, y_1), \ldots, (x_N, y_N) \) with \( y_i \in \{0, +1\}, y_i = 1 \) for positive sample images, \( y_i = 0 \) for negative sample images.
- Initialize weights: \( \omega(t) = \frac{1}{2m}, \frac{1}{2l} \) for \( y_i = \{0, 1\} \), \( m \) and \( l \) are the number of negatives and positives respectively.

\[
\text{for } t = 1, \ldots, T \text{ do}
\]

1. Normalize the weights: \( \omega_{t+1} = \omega_t / \sum_{t=1}^{T} \omega_t \)
2. For each feature \( j \), train a weak classifier \( h_j(x) \in \{0, 1\} \). The error \( \epsilon_j \) is calculated with respect to \( \omega_t \) by
\[
\epsilon_j = \sum_{i=1}^{N} \omega_t | h_j(x_i) - y_i |.
\]
3. Choose the classifier \( h_j \) with the lowest error \( \epsilon_j \).
4. Update the weights: \( \omega_{t+1} = \omega_t \beta_j^{-\epsilon_j} \), where \( \epsilon_j = 0 \) if example \( x_i \) is classified correctly, otherwise \( \epsilon_j = 1 \), and
\[
\beta_j = \epsilon_j / (1 - \epsilon_j).
\]

\[
\text{end for}
\]

- The final strong classifier:
\[
h(x) = \begin{cases} 
1 & \sum_{t=1}^{T} \alpha_t h_j(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\
0 & \text{otherwise}
\end{cases}
\]

with \( \alpha_t = \log 1 / \beta_t \)

2.4. Cascade of classifiers

To increase the detection performance and reduce the computation time Viola and Jones use a cascade of classifiers, which is comparable with a degenerated decision tree (Viola and Jones, 2001). Each stage of a cascade consists of a strong classifier which was trained with Adaboost and consists of a linear combination of weak classifiers. During detection-process the strong classifier of the first stage makes a decision whether the sub-window contains the object. If this is true, it is passed to the second stage of the cascade. When the result is negative the sub-window is rejected immediately and is not analyzed by the following stages (see Fig. 3).

The aim of this approach is to eliminate as much sub-windows as possible in the lower stages of the cascade. Therefore simpler classifiers are used, whereas in higher stages because of the increasing difficulty of the problem the classifiers are getting complexer with each stage.

During training new stages are added to the cascade until given detection and false-positive rates are achieved. These values can
be determined for a cascade with $K$ stages as follows:

$$F = \prod_{i=1}^{K} f_{p_i},$$

(6)

with $f_{p_i}$ false-positive rate of the $i$-th stage. The overall detection rate $D$ of the cascade can be determined analogously

$$D = \prod_{i=1}^{K} d_i,$$

(7)

### 3. Hough Transform

The second part of the program consists of a simple Hough Transform for hyperbolas, which is based on the classical Hough Transform for finding lines in an image that was patented by Hough (1962). Duda and Hart (1972) extended it to the detection of simple geometrical shapes in binary images. The advantage of this method is the reliability in noisy and disturbed binary images. It takes advantage of the fact that simple geometrical objects can be described completely by a few parameters. For the classical Hough Transform all points of the searched object are transformed according to the appropriate mathematical equation from the binary image into the parameter or Hough space. Now the best fitting parameters for the geometrical figure can be extracted by locating the position of the local maximum in the parameter space.

More details and applications of the Hough Transform can be found for example in Illingworth and Kittler (1988).

For the implementation of the Hough Transform we utilize the concept of the accumulator array. This array has the dimension $d$ that corresponds to the number of parameters of the object, the cells of the array match to discrete values of their corresponding parameters. Therefore they have to be divided into discrete values, the range has to be chosen before calculation. In the following the general algorithm for the Hough Transform for different simple geometrical shapes is illustrated (after Huang et al., 1985):

- Determine the minimum and maximum values for each parameter. Through this the dimension of the accumulator array is defined.
- Initialize all accumulator cells with zero.
- Transform each non-zero binary element from the image plane to the parameter plane according to the mathematical equation and increment the corresponding accumulator cell by one.
- Extract the matching parameter for the geometrical figure by searching for a local maximum in the parameter space.

In our case the object that has to be located is a hyperbolic reflection. After Ristic et al. (2009) the travel time of the reflected wave can be expressed with the approximation of a point reflector through

$$t^2 = t_0^2 + \frac{4}{v^2} (x-x_0)^2.$$  

(8)

The form of the resulting reflection hyperbola depends on the velocity $v$ and the position $(x_0, t_0)$ of the vertex, $x$ is the horizontal displacement and $t$ the corresponding travel time. For the Hough Transform we rearrange Eq. (8)

$$t_0 = \sqrt{t^2 - \frac{4}{v^2} (x-x_0)^2},$$

(9)

where $(x, t)$ represent the coordinates of the reflection hyperbola and $x_0, t_0$ and $v$ are the parameters that are Hough-transformed.

Normally the whole image has to be transformed several times for different parameter combinations to find the best-fitting combination. Therefore the largest disadvantage of this method is the high computational requirement: The more parameters are needed for the description, the more time is necessary for the calculations. Our approach is to first limit the image area that has to be transformed by the Viola–Jones Algorithm to decrease the computational demand dramatically and make an on-site use possible.

### 4. The detection scheme

#### 4.1. Data

For training and evaluation of the final cascades four different sets of GPR data are used. One was recorded using Butterfly-antennas with a mean frequency of 50 MHz and a “GSSI² Terra-Sirch System SIR®-3000”. The second data set was measured with a 200 MHz GSSI antenna, type “5106” and Wu-King antennas with a mean frequency of 30 MHz. The other two sets each contain data of a “GSSI 5103” antenna with 400 MHz and a antenna type “3101D” with 900 MHz. All utilized GPR data is unprocessed, only an automatic gain function was applied which can also be calculated on-site.

For testing and evaluating the developed detection program all available data is divided into parts. Only the first three sets are utilized for the learn process, out of them the input images for training and also two test sets with 133 images each (”test set 1” and ”test set 2”) are created. The test sets are not used for the learning process and are chosen randomly, they are necessary to evaluate and test the final detection program. All images of the fourth set form a third test set (”test set 3”) and are only used for evaluating the final cascades. This data set consists of 76 images.

#### 4.2. Haar training

For the training of the final cascade of classifiers a huge number of positive and negative sample images is necessary. The exact position of each reflection hyperbola has to be marked by hand in the positive sample images. Therefore the GPR data is converted into grayscale images and flipped vertically to double the number of available positive sample images. This seems to be reasonable because we assume the background noise in the data has no vertical symmetry. Then every hyperbola is marked with a rectangle, the exact position is saved and the resulting positive sample images (see Fig. 4) are scaled to a fixed resolution for training, for example to 20 $\times$ 20 pixel. The ratio of height to width of the marked hyperbolae is very important. It has to correspond to the aspect ratio of the basic resolution which is fixed for the whole training process. In addition to that the hyperbola has to be centred in the rectangle which should not contain too much, but also not too little background. Otherwise the background would be recognized as part of the hyperbolae and they would not be detected without this unique background. To obtain negative sample images (Fig. 5) the grayscale images are cut, so that they do not contain any hyperbola. The resolution of the negatives can be random, however they should not be smaller than the basic resolution, because the training algorithm automatically creates negative sample images appropriate to the used resolution.

After preparations for the training process a total of 19 final cascades with different basic resolutions, sample images and learning parameters are created for this work in order to find the best parameter combination. All in all for training following settings are used:

2 Geophysical Survey Systems, Inc.
resolutions. In addition to that we test resolutions of 20
best results (Lienhart et al., 2003). For that reason we use both
of cases re
26
sample images with a resolution of 20
signi
10% while the complexity of computation does not increase
feature orientations the false-positive rate can be decreased by
For this work we use Gentle Adaboost, which requires more
boost
"Discrete Adaboost", “Real Adaboost” and “Gentle Ada-
they differ in the way the sample weights are calculated.
For this work we use Gentle Adaboost, which requires more
features than other Adaboost algorithms, but shows the best
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
the other hand all potential hyperbolas with less numbers of
detections are labelled with a red rectangle in the input image.
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
lar is Hough-transformed, the number of
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
The features: For training upright and 45° rotated features are used
(see Section 2.1). After Lienhart et al. (2003) by utilizing both
feature orientations the false-positive rate can be decreased by
10% while the complexity of computation does not increase
Significantly.
Sample image resolution: For frontal face detection positive
sample images with a resolution of 20 × 20 or 24 × 24 pixel show
best results (Lienhart et al., 2003). For that reason we use both
resolutions. In addition to that we test resolutions of 20 × 26 and
and 26 × 34 pixel with an aspect ratio of 0.765, because in the majority
of cases reflection hyperbolas do not have a quadratic shape.
AdaBoost: For the Viola–Jones Algorithm in OpenCV several
variants of the Adaboost learning algorithm are implemented, for
example “Discrete Adaboost”, “Real Adaboost” and “Gentle Ada-
boost”. They differ in the way the sample weights are calculated.
For this work we use Gentle Adaboost, which requires more
features than other Adaboost algorithms, but shows the best
detection results (Lienhart et al., 2003). More details concerning
the various variants of Adaboost can be found in Peters (2006).
Furthermore the number of features each weak classifier consists
of can be chosen for training. Originally the Viola-Jones Algorithm
uses only one feature, but the utilization of two features can
increase the performance (Lienhart et al., 2003). For this reason we
try out both settings.
Detection and false-positive rate: For each stage of the final
cascade a detection rate of 0.999 and a false-positive rate of 0.5 is
demanded. New weak classifiers are constructed and added to the
current stage until both rates are reached. The maximum number
of cascade stages is limited to 30. With these settings, the assumption
of a representative set of sample images and Eq. (7) we
we can achieve a maximum detection rate of 0.99930=0.9704 for
the whole cascade. With Eq. (6) a maximum false-positive rate of
0.530=9.3132 × 10−10 is possible.

The training of the final cascade is a very time-consuming task.
Depending on the number of sample images, the resolution and
the settings, the training can last up to two weeks on modern
computers until the final cascade of classifiers is trained. Once the
the training is completed it can be used for detection.

4.3. The detection program

The developed program for the detection of reflection hyper-
bolas consists of two components. Using the Viola-Jones Algo-
rithm the coarse position of hyperbolas is narrowed down to
certain areas. These regions are further analyzed by the Hough
Transform to calculate the coordinates (x0, t0) of the vertex as well
as the velocity v. Using this approach the computational demand
for the Hough Transform is significantly reduced (see Section 3).

For the Viola–Jones detector the scaling factor of the search
window is important for the detection performance and results. As
described in Section 2 a search window is moved with different
scalings over the input image. Then each sub-window is analyzed
by the detector. With the search-scale-factor the expansion of the
search window after one analysis of the input image can be
influenced. For example if this factor is set to 1.1 the search
window size is increased by 10% after each search round. The
nearer this factor is to 1.0, the finer the search, but the detection
process will take longer and more false detections will appear.

If the position and scale of the search window varies a little, it
is possible that one hyperbola is detected several times. To deal
with these multiple detections and also to reduce false detections
post-processing is necessary. If more than a given minimum
number of neighbouring detections are made in a certain region,
the post-processing step unites them into one single detection. On
the other hand all potential hyperbolas with less numbers of
detections are rejected (see Fig. 6). Finally all recognized hyperbo-
las are labelled with a red rectangle in the input image.

In the following step only the previously marked regions are
analyzed by the Hough Transform. Therefore they are first
smoothed with a Gaussian filter to reduce noise and artefacts.
Then these regions are converted by a canny edge detector (see
Canny, 1986) into a binary image. In a next step every detection of
the Viola–Jones Algorithm is Hough-transformed, the number of
accumulator-cells for x0 and t0 is defined by the number of edge
pixels (x, t). The third parameter v is varied in a given interval,
which can be set by the user. These three parameters define the
size of the accumulator-array. Then every edge pixel (x, t) is
transformed with Eq. (9) into the Hough plane for every discrete
value of v. Only if the calculated value for t0 is real and greater
than zero the corresponding cell in the accumulator-array is
incremented by one. When this procedure is finished for every
possible parameter combination, the accumulator-cell with the
maximum count is searched and the coordinates (x0, t0) of the

A selection of positive sample images for training, the resolution is 24 × 24 pixel.

A selection of negative sample images for training (scaled).
vertex and the velocity $v$ are extracted. These parameters are used to calculate the travel time $t$ with Eq. (8). At the end the detected shape of the reflection hyperbola is drawn in the original input image (see Fig. 7).

5. Results

In this section the cascade of classifiers with the best results is introduced. For evaluating the detection results and comparing them to other cascades we use the “performance-program” (see Intel and Seo, 2006) that is delivered with the Haartraining package of OpenCV. Every final cascade of classifiers was applied on the three test data sets using this program (see Table A1). All cascades show the best results on test data set 3. The reason for that is, that this set contains the clearest hyperbolas and that the noise level of the data is low. The other two sets contain noisy data and reflections that are much more difficult to recognize.

In order to utilize the performance-program sample images with labelled hyperbolas are needed as input. For this we use the images we created for testing purposes during training preparation (Section 4.2). The performance-program utilizes the trained cascade of classifiers, applies it on all input images and compares the detected and manually marked positions of hyperbolas. If a detection matches the manual labelling it is rated as correct detection. On the other hand, if the position of the detection differs from the marking, or the detection is not manually marked as hyperbola, it is regarded as false-positive detection. In addition

Fig. 6. Example for the output of the detection program, the image belongs test data set 3 (900 MHz antenna). On the left side no post-processing was done, that is why a lot of multiple detections can be seen. On the right side post-processing was applied and all multiple detections disappear. In addition to that two correct detections are removed (arrows), because they were only detected once.

Fig. 7. One result of the Hough Transform for hyperbolas. At first the sub-window (at the top of the left) is smoothed and transferred into a binary image by a canny edge detector (at the top in the middle). Afterwards a Hough Transform is applied on the sub-window, the calculation took 384 ms for a sub-window resolution of $47 \times 62$ pixel. The calculated hyperbola equation is drawn in red (at the top of the right). At the bottom the accumulator array is illustrated, the cell with the maximum count can be seen.
to that the overall number and the number of missed hyperbolas in the input images is counted. With these information the detection and false-positive rate of each cascade can be calculated.

In Fig. 8 an output image of the performance-program is shown. Correct detections are labelled with a green, false detections with a red rectangle and the manually marked hyperbolas for training are labelled in blue. Both GPR data sets belong to test set 1. For the detection process we use a basic resolution of \(24 \times 24\) pixel, a cascade with 30 stages, 1680 weak classifiers and 2 features per weak classifier, which is the cascade that shows the best overall results with the few available data we had. The training of this cascade was done with 3020 positive and 4985 negative sample images. In the upper radargram all hyperbolas were identified correctly, furthermore one unmarked hyperbola was found (red) whereas the hyperbola at the top right corner was missed. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

All in all the results of the performance-program show that a basic resolution of \(24 \times 24\) pixel provides the best detection results, \(20 \times 20\) pixel the worst. The first five features of the best performing cascade are illustrated in Fig. 9. It becomes clear that the first four features concentrate on the intensity distribution in the area of the vertex. The fifth feature seems to be sensitive to the existence of the hyperbola branches. To obtain exact information of the cascade performance we apply our detection program with the best performing cascade on all three test sets. After that we manually count all correct and false detections as well as the overall number of contained hyperbolas. The resulting detection and false-positive rates can be seen in Table A1. As described before the manually calculated rates, especially the false-positive rate, differ significantly from the rates that are calculated with the performance-program. In Fig. 10 two output-images of the detection program with the best performing cascade can be seen. After the Viola–Jones detector has narrowed down the regions containing hyperbolas (red rectangles) these areas are Hough transformed and the hyperbola equations are calculated (blue). In the left radargram all hyperbolas were found, on the right side at least two were missed (arrows). More results can be seen in Figs. 11 and 12. The whole detection process took in most cases approximately 1–2 s at maximum.

6. Discussion

The creation of positive and negative sample images for the training process was partly difficult. Especially in the unprocessed low-frequency data many structures can be recognized. Often it is very hard to decide if these structures are hyperbolas. Besides, the used data contains many radargrams without hyperbolas, which differed a
lot from data containing them. This complicated the creation of negative sample images, because the difference between positive and negative sample images should not be too large. If just one negative sample image contains a hyperbola, the detection results are of poor quality, because the originally positive sample image is classified as false detection in every boosting round and therefore is utilized during the whole training process. In addition to that, a compromise between the size and the amount of background for the manual marking of hyperbolas for training had to be found. Too little background can result in poor detection results. If the markings contain too much background this also leads to poor detection results, because the algorithm searches for hyperbolas with similar background.

On test set four the final cascades of classifiers showed the best detection rates, although the data was not used for training. The reason for that is that this data set contains the clearest hyperbolas and has the lowest noise level. Set 1 and 2 contain much more noise which makes the recognition of hyperbolas in them more difficult and complicates the detection process for these data sets.

As can be seen in Section 5, Fig. 8 some hyperbolas were missed during the manual marking of reflection hyperbolas. Nevertheless the detection program recognized these missed hyperbolas and because of the missing label, the performance-program rated them as false detections. This leads to a higher false-positive and a lower detection rate in Table A1 than can be achieved in reality.

Unfortunately no independent data of other GPR systems was available to test the detection program. Therefore we get no insight concerning the detection results on completely independent GPR data. Although the images of the three test sets were not used for training, the data was recorded with the same GPR-systems, which were also utilized to record the sample images. It would be interesting to see how the detection program copes with data of other independent GPR-systems that was not used for training.

In this paper we show that the presented detection system works in principle and that acceptable detection results can be achieved by the use of only a few sample images for training. With more available data of varied GPR systems the detection results could be increased significantly. Another option would be to use processed sample images as input for the training or only train a classifier for one special GPR system with one or two antennas maximum. This would also improve the results on unprocessed data in the field.
A comparison with other available methods is difficult, because the most studies use processed data. For example Simi et al. (2008) achieve slightly higher detection and lower false-positive rates but these results are not comparable to our since they process and optimize the data for hyperbola extraction. All in all our results promise a detection system with much better results than could be achieved with the few available data for this work. As can be seen in several other studies the achieved results strongly depend on the data quality available for training.

7. Conclusions and outlook

Altogether the results of the developed detection program is capable of the automatic localization and fitting of reflection hyperbolas in unprocessed GPR data in real-time. By using this method the expenditure of time when searching for objects that produce a reflection hyperbola can be decreased significantly. Even with a low number of input sample images for training the detection program achieves acceptable results. It becomes clear that the quality of detection results depends strongly on the quality of the available data for training. When the detection program shall only be utilized for one special GPR system, then only data of this special system is necessary for training. With many different radargrams of various GPR systems it should be possible to create a general hyperbola-detection with good detection results, which is also able to localize reflections in real-time in the field.

Appendix

See Table A1.

References


Dailey, M., Bo, N., Towards real-time hand tracking in crowded scenes DOI: 1011.159.6410, in press.


Table A1

<table>
<thead>
<tr>
<th></th>
<th>Test set 1</th>
<th>Test set 2</th>
<th>Test set 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Calculated with performance program</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detection rate</td>
<td>0.6005</td>
<td>0.5902</td>
<td>0.7615</td>
</tr>
<tr>
<td>False-positive rate</td>
<td>0.2037</td>
<td>0.3396</td>
<td>0.2301</td>
</tr>
<tr>
<td><strong>Manually counted</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall hyperbolas</td>
<td>538</td>
<td>688</td>
<td>443</td>
</tr>
<tr>
<td>Correct detections</td>
<td>352</td>
<td>466</td>
<td>333</td>
</tr>
<tr>
<td>False detections</td>
<td>32</td>
<td>46</td>
<td>29</td>
</tr>
<tr>
<td>Missed hyperbolas</td>
<td>186</td>
<td>222</td>
<td>110</td>
</tr>
<tr>
<td>Detection rate</td>
<td>0.6544</td>
<td>0.6774</td>
<td>0.7517</td>
</tr>
<tr>
<td>False-positive rate</td>
<td>0.0595</td>
<td>0.0669</td>
<td>0.0655</td>
</tr>
</tbody>
</table>

Fig. 12. Three more results of the final detection program. All radargrams were measured with a 200 MHz antenna and belong to test data set 2.
Huang, K., Fu, K., Sheen, T., Cheng, S., 1985. Image processing of seismograms: (a) hough transformation for the detection of seismic patterns; (b) thinning processing in the seismogram. Pattern Recognition 18, 429–440.