Recognizing Separate Structural Elements of Churches Using Neural Networks

MAKSIM BRODOVSKII[1], DMITRII KOROVIN[1], MARIA CHIZHOVA[2], ANSGAR BRUNN[2] & UWE STILLA[3]

Abstract: In this paper we develop a new approach to recognize structural elements of orthodox churches. We will work with 3D point clouds, received as a result of 3D point cloud acquisitions of churches, e.g. from laser scanning. Because of the large amount of points in such clouds, we have to use a projection (elevation) to decrease the calculation effort. To get meaningful images from the projection of the point cloud we do some prior segmentation of the 3D cloud. Images binary, with a predefined resolution that depends on the resolution of the 3D point cloud. To recognize elements we decide to use neural networks (Perceptron and Counter propagation neural networks) as they allow the automation of the process and have a broad range of methods to recognize images. For the subsequent 3D modeling we use analytic expressions, that describe each of the structural church elements. A further step is the deduction of those expressions that describe each of the recognized sectional views.

1 Introduction

1.1. Motivation

In a previous article (KOROVIN et al. 2016) a new approach for the reconstruction of Russian Orthodox churches have been developed on the basis of a Bayesian network and cellular automaton. The aim of that work was to present an algorithm, which develops the optimal process sequence of the automatic search, detection of the set of geometric objects and reconstruction with high probability of buildings and building components from a point cloud independent of its destruction. The first step of it was to recognize the single components of the church. In our work we would like to present the method, with which the geometry of such components could be recognized.

1.2. Previous Work

The recognition of object components with the aid of point clouds is one of the biggest tasks of reverse engineering used in many branches like industry, architecture, 3D modeling and robotics. The extraction of geometric information from the point cloud and its interpretation play a key role in this process. There are some common algorithms and mathematical techniques using 3D data e.g. from laserscanning as well as 2D data: RANSAC, Hough Transform, Least square fitting etc. The RANSAC-algorithm finds a mathematically described model from a set of data e.g.

[1] Ivanovo State Power Engineering University (Russia)
[2] Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt, Labor für Photogrammetrie & Fernerkundung, Röntgenring 8, D-97070 Würzburg, E-Mail: [maria.chizhova, ansgar.brunn]@fhws.de
3 Technische Universität München, Photogrammetrie und Fernerkundung, Arcisstraße 21, 80333 München, E-Mail: stilla@tum.de

laser scanning point data and estimates the model parameters by the interpretation of data as inliers (belong to model) and outliers. The algorithm has its advantages and disadvantages.
Advantages are:

- robust model estimation
- feasibility to find a geometric object in a noisy point cloud

Disadvantages are:

- the algorithm can estimate only one model and is not suitable for complex objects,
- the algorithm correctness depends from the number of iterations and mistake threshold,
- time involved.

SCHNABEL et al. (2007) uses an efficient RANSAC for the fitting of geometrical forms. The algorithm finds a best-fit geometrical form from candidates for the surface.
Hough Transform is a common method for detecting simple shapes like straight lines and circles in digital images, a 3D-shape evaluation (e.g. Kernel-based Hough Transform) can be used for simple geometric shapes like plane, cylinder and sphere. Geometries detection from point cloud with Hough Transform is presented in VOSSELMAN et al. (2004) and RABBANI et al. (2005). It is possible to find curves, too. However, the algorithm is generally suitable for simple forms and has some limitations like:

- the object size must be relative big, because the number of votes can fall to neighboring objects;
- the large number of model parameters can lead to mistakes;
- the quality of data plays an important role for algorithm efficiency (data denoising is preferred)

The method of least square fitting have been considered in AHN (2004). MARSHALL et al. (2001) presents an algorithm for the least squares fitting of spheres, cylinders, cones, and tori to point data. Although the least square method became a standard method for model estimation, it is not robust and can lead to the finding of incorrect model by noisy dataset. Huber (1964) developed in his work the method of robust parameter estimation. The assumption of this method is an explicit parameter modeling, which we do not need using neural networks.
The fitting of architectural models is considered in CANCIANI (2013). The method bases on the modelling paths definition, i.e. the profile of single component was extracted by the point cloud, analytical described. Then a building component is generated from knowledge-based model of profile line using different paths like circular, square, octagon etc. In ALBY & GRUSSENMEYER (2012) the architectural elements were modeled using the comparison of a knowledge-based model with the point cloud: a geometric primitive was closely adjusted to the point cloud and the surface distance for more as 90% of, by geometric primitives simplified, model within 5 cm of the point cloud becomes a resulting model.
An interesting algorithm of a surface fitting from unorganized, sparse, noisy point cloud using neural networks based on regression algorithm is presented in YUMER & KARA (2012). A free form surface was generated from a 3D point sets through a parametric embedding and tessellation in 2D-coordinates space. Then a neural network will be trained to learn a mapping by corresponding 3D to 2D coordinates, resulting in the synthesis of 2D surface in the model space. The activation function effects at the same time on the surface on the resulting surface.

ALEXANDRE (2014) presents different ways to train convolutional neural networks (CNN) with RGB-D data (color and depth). The approaches of independently for each channel trained CNN and by transfer learning trained CNN have been compared and analyzed. A wide investigation of neural networks is considered in CICHY et al (2016). The human brain capacity to recognize visual object has been presented through Deep Neural Networks based on anatomical analyses.
The aim of our work is to recognize complex structural components, which cannot solve as geometric primitives.

2 Preprocessing and data preparation

2.1 Prepared data

The patterns for the training of neural network are symmetrical black-white 2D images of church structure elements that illustrate vertical and horizontal projections of its elements.
The input data are point clouds of church individual structural elements with the correct orientation, i.e. they are straight with no tilting. In our case, these elements are the domes, roofs and crosses. For pattern recognition, we will use 2D images, because it drastically reduces the computation due to a significant reduction in the number of points considered. These images will be black and white. Black pixels will correspond to the points of the point cloud. All other pixels are white.
In our work we have simulated in raster graphics editor Adobe Photoshop the point cloud cuts of structural elements (Fig. 1d) with taking in account the average resolution (with distance between closest points), which is usually suitable by laser scanning of such elements, and possible noise. Moreover, we try our algorithm with real laser scanning data namely point cloud cuts from Russian Orthodox Church in Wiesbaden (Germany) using 3D scanning software Trimble Realworks.

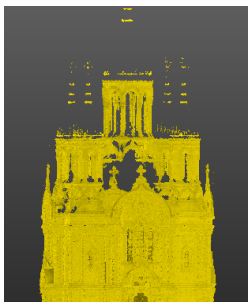

Fig.1a: Point cloud of the Russian Church in Wiesbaden

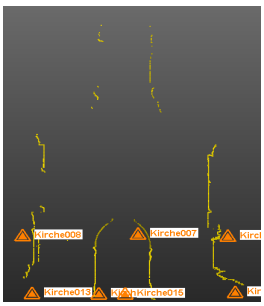

Fig.1b: Vertical projection (cuts on different distances) of the whole Church

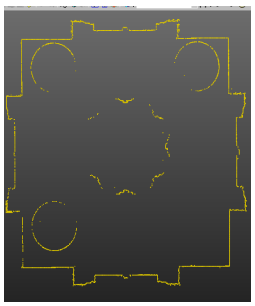

Fig.1c: Horizontal projection (cuts on different heights) of the whole Church

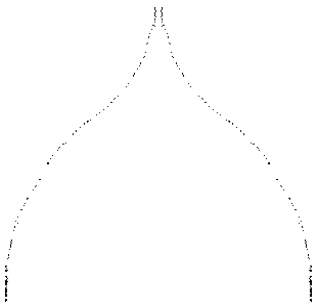

Fig.1d: Simulated vertical projection of the cupola

2.2 Processing of 3D point cloud (obtaining 2D black and white images)

Our first task is to get proper 2D image. To do this, we will use the vertical and horizontal projections. The horizontal projections will be secondary, as they will be used to obtain information

about analyzed object. In our case, there could be two types of the horizontal projections: a circle and a polygon. If the horizontal projection is a circle, we are dealing with body of rotation, and to obtain the desired vertical projection we should find a section, that would pass through the center of the circle. If the horizontal projection is a polygon, in general, the section for vertical projection would be parallel to one of the polygon sides. In polygon is a rectangle, the section for vertical projection would be parallel to the longest (or shortest) polygon side. If the polygon is not a rectangle, the projection would pass through the polygon center and intersection of two sides.

Thus, upon obtaining a horizontal projection, we should detect what is it, a circle or a polygon. One of the main differences between a polygon and a circle is presence of intervals that lie along lines. Thus, with the help of Hough transformation we could calculate the number of points that lie on such intervals. Using a predetermined threshold value for quantity of points on a segment (for example, 5 points) we can determine the number of such segments. Hough transformation also could help us to determine the analytical formulas for those lines. Using these formulas we could calculate the angles of the polygon and determine, is it a rectangle or not. The longest side of a polygon can be determined by the highest number of points that belong to the corresponding line that defines the segment. As to the center of a circle or polygon, it can be determined as mass center of black points on horizontal projection (1). Thus, using the horizontal projection, we can get the desired vertical projection, which we could use as the image for recognition. However, to increase the probability of correct recognition we could "improve the quality" of the resulting image, i.e. increase the density of black pixels. If horizontal projection is a circle, we could rotate the section plane λ (for vertical projection) that pass through the center of circle, with a certain step α (for example, 1 degree) and project this section on the plane δ so that the position of the line corresponding to the center of the circle remain constant (Fig. 2a). When the horizontal projection is a rectangle, we could move the section plane λ (for vertical projection) along the side, that is perpendicular to this section, with a certain step h (for example, 1 pixel). The vertical projection plane δ remain constant during moving of the section plane λ (Fig. 2b).

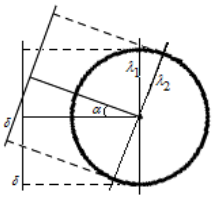

Fig. 2a: Increasing density for circle projection

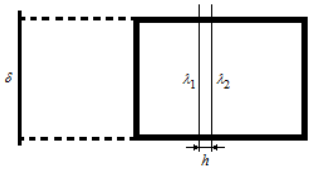

Fig. 2b: Increasing density for rectangle projection

3 Image recognition using counter propagation neural network

3.1 Outline of the method

Counter propagation neural network operates on the principle "winner takes all". It enables to apply clustering of input images and on the output it supplies the number of cluster for an input image. Counter propagation neural network consists of two layers: Kohonen layer and Grossberg layer (Fig. 3).

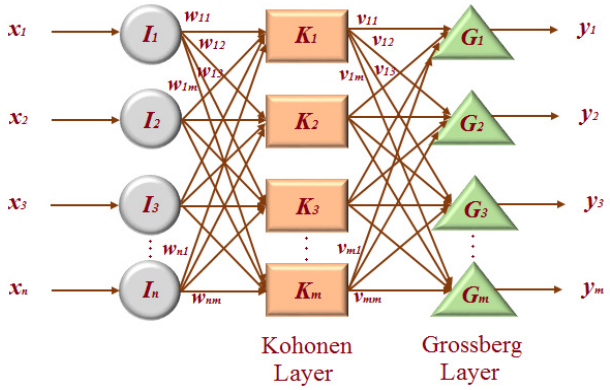

Fig. 3: Structure of Back propagation neural network

Values of Kohonen layer neurons are calculated as follows:

$$K_i = \sum_{j=1}^{n} x_j w_{ji}, \quad i = 1,2,...,m. \qquad \text{(Eq. 1)}$$

A neuron with maximum value considered as "winner" neuron and on the output it supplies 1, other neurons supply 0 to the output. Let $K_1, K_2, ..., K_m$ be the output values Kohonen layer. Grossberg layer allows us to match the "winner" neuron with the recognized cluster of objects. Values of Grossberg layer neurons are calculated using following formula:

$$G_i = \sum_{j=1}^{m} k_j v_{ji}, \quad i = 1,2,...,m. \qquad \text{(Eq. 2)}$$

In our application number of neurons in Kohonen and Grossberg layers are equal. First we train the Kohonen layer until each object cluster will activate only one unique neuron. Training of Kohonen layer is unsupervised training. For training we use formula:

$$w^*_{ij} = w_{ij} + \eta(x_i - w_{ij}), \quad i = 1,2,...,n. \qquad \text{(Eq. 3)}$$

Where w^*_{ij} — new weight value, w_{ij} — current weight value, η — speed of training ($\eta < 1$). Then we train the Grossberg layer. We supply to the input of the network image from a specific cluster of objects, it activates corresponding neuron on Kohonen layer, then we set the weights for that neuron in the Grossberg layer so that on the output our network they activates neuron with the number of the corresponding cluster. In our case all weights for "winner" neuron will be equal to 0 except for the weight v_{ij} that is equal to 1, where i is the number of neuron winner, and j is the number of corresponding cluster.

To distinguish images supplied to the input of the counter propagation neural network, it is necessary to find characteristics that will allow differentiating one image from another. For this purpose we decide to use "characteristic" points, which we determine with the help of two following ways:

1. Calculation the mass center and searching for two of the most nearest and two of the most distant points from the mass center.
2. Searching for critical points, i.e. points that have slope of the curve close to 0 or 90 degrees.

Let us review each of the methods in detail.

3.2 Calculation the mass center and searching for two of the most nearest and two of the most distant points from the mass center

The choice of this method was determined by the fact that the distribution of points on a projection of the resulting three-dimensional point cloud and the original image must be approximately the same. In theory, the projection should be different from the original image only by density of points, but the pattern of point allocation should be the same.

For calculation of coordinates for mass center of points on the image we used the following formula:

$$x_c = \frac{\sum_{k=1}^{n} x_k}{n}, \quad y_c = \frac{\sum_{k=1}^{n} y_k}{n} \qquad \text{(Eq. 4)}$$

Where (x_k, y_k) is the coordinates of black pixels on the image, and n is the number of black pixels in the image. Here and after by coordinates x and y we mean number of column and number of row on the intersection of which the pixel of interested is located. Numbering of columns and rows begin in the left low corner of image. Pixel in left low corner has coordinates (1;1).

Upon calculation of mass center (A, Fig. 4b) for black pixels on the image we search for 2 black pixels that are closest to the mass center (D and C, Fig. 4b) and 2 black pixels that are farthest form the mass center (D and E, Fig 4b), we also calculate the distances to those pixels from mass center. Moreover, when searching for the second closest or farthest pixels we do not take in account pixels from the circular neighborhood of the first corresponding point (Fig. 4b). The neighborhood radius is predefined (for example, 5% of the diagonal length of the image supplied to the input). For all received coordinates and distances we perform normalization, i.e. we divide each value by the length of the image diagonal. Obtained coordinates and lengths we supply to the input of the neural network in specific order.

The advantages of this method:

- It does not require the processing of original image except cropping on marginal black pixels;
- Density of points on the recognizable image may be less than the density of points on the reference image;
- Selected characteristic points allow differentiating a wide range of images.

Fig. 4a: Example of recognized image

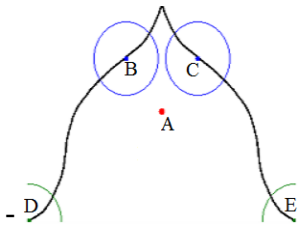

Fig. 4b: The image with mass center and related points

3.3 Searching for critical points

With "critical points" we understand points on the pattern and input data, which have slope of the curve close to 0 or 90 degrees. This method is also suitable to us, since regardless of the distribution density of pixels on the reference image and the projection, the location of the critical points and its number pro cut (object) should be approximately the same.

For each black pixel of the image we search for two closest black pixels and consider two pairs of pixels $B(x_0, y_0)$, $A(x_1, y_1)$ and $B(x_0, y_0)$, $C(x_2, y_2)$ where $B(x_0, y_0)$ —point of interest, $A(x_1, y_1)$ and $C(x_2, y_2)$ —two closest points (Fig. 4a). First, we calculate the slope coefficients of lines that pass through the selected pair of pixels:

$$k_1 = \frac{y_1 - y_0}{x_1 - x_0}, \quad k_2 = \frac{y_2 - y_0}{x_2 - x_0}. \qquad \text{(Eq. 5)}$$

Then, with the help of the coefficients, we calculate the angle of slope (Fig. 5b and 5c):

$$\alpha_1 = arctg(k_1), \quad \alpha_2 = arctg(k_2). \qquad \text{(Eq. 6)}$$

Then calculate the angle for the point of interest as the average of those two angles of slope (Fig. 4d):

$$\alpha_0 = \frac{\alpha_1 + \alpha_2}{2}. \qquad \text{(Eq. 7)}$$

For each image we search for pixels, slope angle values of which are close to 0 or 90 degrees. We consider those points as critical and use them as characteristic points to compare images. Coordinates of such points we supply to the input of neural network in specific order. The number of such points is predefined.

Advantages of this method:

- It does not require the processing of original image except cropping on marginal black pixels;
- Density of points on the recognizable image may be less than the density of points on the reference image;
- Selected characteristic points allow differentiating a wide range of images.

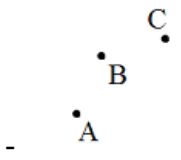

Fig. 5a: Three nearest points

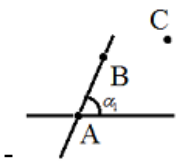

Fig. 5b: Obtaining α_1

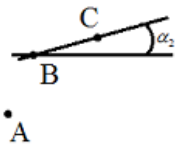

Fig. 5c: Obtaining α_2

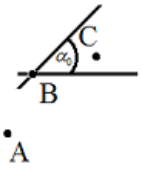

Fig. 5d: Obtaining α_0

4 Pattern recognition using perceptron

4.1 Outline of the method

Perceptron is one of the simplest neural networks, which allows variety of applications.

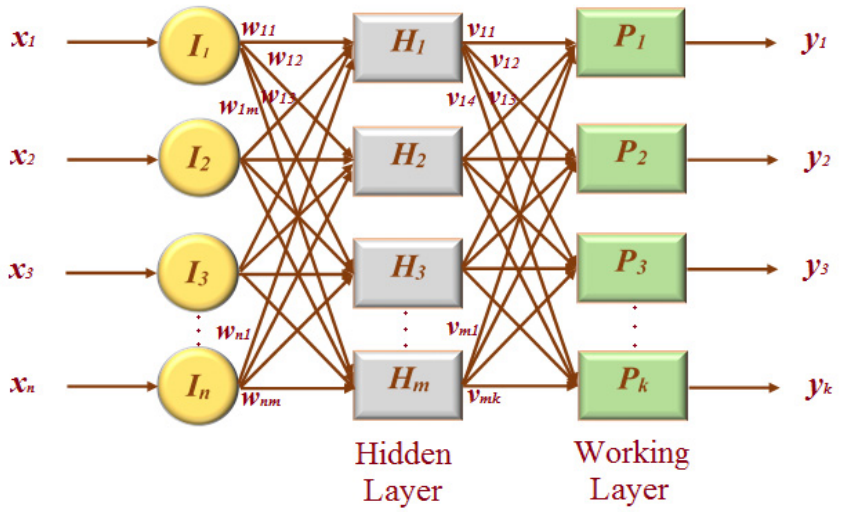

Fig. 6: Structure of perceptron with the hidden layer

Here we decide to use the hidden layer with the predefined number of neurons in it to increase an accuracy of image recognition. Values for neurons in the hidden layer are calculated as follows:

$$H_i = \sum_{j=1}^{n} x_j w_{ji}, \quad i = 1,2,\ldots, m. \qquad \text{(Eq. 8)}$$

Then, to determine output values of hidden layer we apply function f to each neuron:

$$h_i = f(H_i), \quad i = 1,2,\ldots, m, \qquad \text{(Eq. 9)}$$

$$f(x) = \frac{1}{1+e^{-x}}. \qquad \text{(Eq. 10)}$$

Calculating of output values for working layer neurons is similar:

$$P_i = \sum_{j=1}^{n} h_j v_{ji}, \quad i = 1,2,\ldots,k, \qquad \text{(Eq. 11)}$$

$$y_i = f(P_i), \quad i = 1,2,\ldots,k. \qquad \text{(Eq. 12)}$$

Perceptron training is supervised. We supply data from reference images to the input of perceptron and make an adjustments of weights based on deviation from intended output. Weights adjustment is made using back propagation training rule:

$$\delta_i^{(1)} = z_i - y_i,\ i = 1,2,...,k, \quad \text{(Eq. 13)}$$

$$\delta_i^{(2)} = \sum_{j=1}^{k} v_{ij}\delta_j,\ i = 1,2,...,m, \quad \text{(Eq. 14)}$$

$$w_{ij}^* = w_{ij} + \eta\delta_j^{(2)}\left(h_j\left(1-h_j\right)\right)x_i,\ i = 1,2,...,n,\ j = 1,2,...,m, \quad \text{(Eq. 15)}$$

$$v_{ij}^* = v_{ij} + \eta\delta_j^{(1)}\left(y_j\left(1-y_j\right)\right)h_i,\ i = 1,2,...,m,\ j = 1,2,...,k. \quad \text{(Eq. 16)}$$

Where z_i — intended output, $\delta_i^{(1)}$ — errors of working layer, $\delta_i^{(2)}$ — errors of hidden layer, w_{ij}^* and v_{ij}^* — new weight values, η — speed of training ($\eta < 1$).
In this case to the input of perceptron we supply a matrix with elements that equal 0 or 1. We will use adjacency matrix as an input, obtaining of which will be described below. The number of neurons in our working layer of perceptron coincide with the number of reference images. At the output we get 0 values from all neurons except the one, number of which, corresponds to cluster of recognized image.

4.2. Dividing the image into rectangles, adjacency of which is determined based on connected pixels

The point of this method is to divide an image into predetermined number of rectangles (for example, 400 (20x20) rectangles). All rectangles have the same size. The width and height of the rectangles are calculated as follows:

$$w_s = \frac{w_i}{n},\ h_s = \frac{h_i}{n}. \quad \text{(Eq. 17)}$$

Where w_i и h_i are width and height of the image, respectively, and $n \times n$ is predetermined number of rectangles. To get rectangles width and height as integers we add missing rows and columns of white pixels to the image edges.
To determine connection between pixels we start to move along the bottom border of the lower left corner of the image, we search for first pixel. As we find it we connect it to the nearest pixel. Next for this nearest pixel we also search for the nearest pixel, excluding from our search the first pixel, and so on. Thus, we consistently connect all black pixels on the image in one curve. Two neighboring rectangles (each rectangle have only 8 neighboring rectangles) are considered adjacent if there are two connected points, one of which belongs to the one rectangle, and the other — to the other rectangle. So we define the adjacency of neighboring rectangles (Fig. 7c). Then we create adjacency matrix of these rectangles as follows. We enumerate all rectangles from 1 to k where $k = n \times n$. Then we take a zero matrix with the size $k \times k$, and write in it 1 at the intersection of i-th row and the j-th column, if rectangles with numbers i and j are adjacent. Obviously, such a matrix would be symmetric. This matrix we supply to the input of the perceptron.

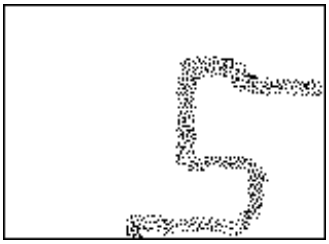

Fig. 7a: Example of image prior processing

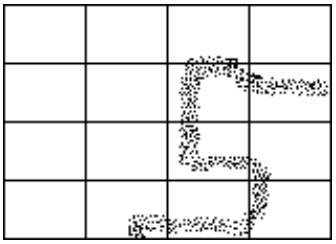

Fig. 7b: Image divided on equal rectangles

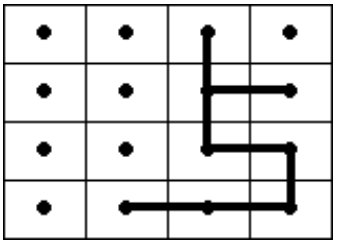

Fig. 7c: Interpretation of adjacency matrix after image processing

Advantages of this method:

- It does not require the processing of original image except cropping on marginal black pixels;
- Density of points on the recognizable image may be less than the density of points on the reference image;
- It is less sensitive to minor changes (within the same rectangle) and preserves the overall structure of the image;
- Selected characteristic points allow differentiating a wide range of images.

5 Results

To test our methods we require some training set. We simulated vertical projections of corresponding point clouds. The only way to do it is to change reference images somehow to make them similar to projections from 3D point cloud. We decide to take black pixels from reference images and change their location with the help of probability theory (Fig. 8a and 8b). We predefine to argument p and r. First argument is responsible for density of pixels, second argument is responsible for location change. For every black pixel on the image we generate random variable V that is evenly distributed on the interval [0;1]. If $V < p$ than we changed the location for the pixel of interest to $(x_0 + t, y_0 + t)$ where t is integer random variable that is evenly distributed on the interval $[-r; r]$. If $V \geq p$ then we delete the pixel of interest, i.e. replace it with white pixel. So if $V = 1$ we do not delete any black pixel, if $V = 0$ we delete all black pixels. For test and training sets we use for p integer values from 50 to 100, and for r we use integer values from 1 to 4.

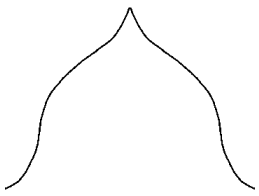

Fig. 8a: Example of reference image

Fig. 8b: Image after simulation vertical projection with p=50, r=4

The perceptron showed the best results. For domes we divided images on 20x20 rectangles and used perceptron with 28 neurons in hidden layer, it recognized correctly 13 of 14 images at average. For roofs we divided images on 20x20 rectangles and used perceptron with 15 neurons in

hidden layer, it sometimes made only 1 mistake for roofs generated with r=4, but mostly it recognized all roofs with no mistake. For crosses we divided images on 20x20 rectangles and used perceptron with 15 neurons in hidden layer, it recognized all crosses with no mistake.
Counter propagation neural network and mass center calculation method also showed good results. At average for domes 12 of 14, for crosses 6 of 7 and for roofs 5 of 6 images were recognized correctly. For the neighborhood radius we used the value equal to 10% of the diagonal length of the image.
Unfortunately, counter propagation neural network and critical points method showed poor results. For reference images it allows to correctly recognize all the images but for simulated images it mistakes for 3 and more images for every structural element. It could be explained by the fact that for proper angle of slope calculation curves on the image should have width of 1 or 2 pixels. Otherwise proposed method of slope angle calculation shows wrong results.

6 Conclusion

Obtained results tell us that not always we could determine structural element of the church correctly with 100% accuracy. But proposed methods showed their efficiency and more than that it is only the first steps and there is some room for further enhancements. For example, new algorithm for slope angle calculation in counter propagation neural network and critical points method could be developed. If we would be able to differentiate outer pixels out of others those method could be successful. As for counter propagation neural network and mass center calculation method there could be developed some algorithms that resolve issues with unsymmetrical results (for example, if point B would be in the uppermost black pixel, point C with equal possibility could be in the left side and on the right side of the image; Fig. 3b). And for perceptron there are plenty of opportunities. We could increase the number of hidden layers, we could change neuron activation functions and we could enhance method of rectangle adjacency determine.

7 Acknowledgements

We would like to thank the students of Ivanovo State Power Engineering University – Anatolii Bolshakov and Maksim Lobanov - for program realization of our algorithms.

8 References

AHN, S., 2004: Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space. Lecture Notes in Computer Science **3151**, Springer Berlin Heidelberg, 125 pages.

ALBY, E., & GRUSSENMEYER, P., 2012: From Point Cloud to 3D Model, Modelling Methods Based on Architectural Knowledge Applied to Fortress of Chatel-sur-Moselle (France). International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **39** (B5), 75-80.

ALEXANDRE, L., 2014: 3D Object Recognition using Convolutional Neural Networks with Transfer Learning between Input Channels. Intelligent Autonomous Systems **13**, Springer International Publishing, 889-898.

CANCIANI, M., FALCOLINI, C., SACCONE, M., & SPADAFORA, G., 2013: From Point Clouds to Architectural Models: Algorithms for Shape Reconstruction. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **40** (5/W1), 27-34.

CICHY, R. M., KHOSLA, A., PANTAZIS, D., TORRALBA, A., & OLIVA, A., 2016: Deep Neural Networks Predict Hierarchical Spatio-temporal Cortical Dynamics of Human Visual Object Recognition. arXiv:1601.02970, http://arxiv.org/pdf/1601.02970v1.pdf.

HUBER, P., 1964: Robust Estimation of a Location Parameter. Annals of Mathematical Statistics **35** (1), 73-101.

KOROVIN, D., CHIZHOVA, M., BRUNN, A. & STILLA, U., 2016: Probabilistic Reconstruction of 3D Buildings using Bayesian Nets. Submitted manuscript for the journal Photogrammetrie Fernerkundung Geoinformation (PFG).

MARSHALL, D., LUKACS, G. & MARTIN, R., 2001: Robust Segmentation of Primitives from Range Data in the Presence of Geometric Degeneracy. IEEE Transactions in pattern analysis and machine intelligence **23** (3), 304-314.

RABBANI, T. & VAN DEN HEUVEL, F., 2005: Efficient Hough Transform for Automatic Detection of Cylinders in Point Clouds. Proceedings of the ISPRS Workshop Laser scanning 2005, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences Volume **36** (3/W19), 60-65

ROSENBLATT, F., 1957: The Perceptron: a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory.

RUMELHART, D.E. & MCCLELLAND, J.L., 1986: Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1 – Foundations, MIT Press Cambridge MA, USA.

SCHNABEL, R., WAHL, R. & KLEIN, R., 2007: Efficient Ransac for Point-Cloud Shape Detection. Computer Graphics Forum **26** (2), 214-226.

VOSSELMAN, G., GORTE, B., SITHOLE, G. & RABBANI, T., 2004: Recognizing Structure in Laser Scanner Point Clouds. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences **46**, 33-38.

YUMER, M. & KARA, L., 2012: Surface creation on unstructured point sets using neural networks. Computer-Aided Design **44**, (7), July, 644-656.