Discovering Geo-Magnetic Anomalies: a Clustering-Based Approach

Elena Volzhina, Andrei Chudin, Boris Novikov, Natalia Grafeeva, Elena Mikhailova
Saint Petersburg State University
St. Petersburg, Russia
lenavolzhina@ya.ru, andrei.chudin@gmail.com, {b.novikov, n.grafeeva, e.mikhaylova}@spbu.ru

Abstract—**The paper investigates the application of clustering algorithms to data obtained by magnetometry in the places of archaeological excavations. The goal was to enable automatic detection in order to find possible anomalies in the data that could indicate the presence of potential archeological sites. Analysis of the results of following experiments with clustering algorithms is presented: *k*-means, *c*-means and DBScan. The algorithms were run on a number of datasets, and visual assessment by experts confirmed the high level of accuracy of the results.**

I. Introduction

The application of geophysical methods to conducting archaeological explorations has come to be a separate research area where observation and data interpretation techniques are specific to archaeological research objects. The study of such objects involves finding, identifying and mapping remains of ancient buildings or sites of past human activity. As a rule, such places are hard to locate on the surface of the earth as a result of land reclamation and other processes, whether caused by man or nature. This leads to the expansion of a search area and complicates planning of the archaeological excavations.

Magnetometry [3] is one of the most common non-destructive detection methods currently used in archaeology. The method is quite simple: with the help of a magnetometer, the values of magnetic field are measured in a number of pre-determined points. The obtained data is then processed resulting in a readable map, by looking at which an expert could easily recognize an anomaly, that is, an area or areas, where the value of field magnitude differs from the average value for the entire search area. Such anomalies may have appeared in the search area for various reasons. The hidden presence of an archaeological object of interest is only one reason, while a frustratingly large number of anomalies is caused by metallic debris left from agricultural activities, military operations or magnetometer measurement error [1], [4].

Currently, there is not a single automated technique capable of identifying such anomalies as those described above. It still takes a human expert with extensive experience in magnetometric detection to discover them. The present research is an attempt at automating anomaly identification. In order to accomplish that, we evaluate the applicability of several clustering methods [10].

II. Related works

Presently, archaeological research employs large amounts of indirect digital data: a variety of space imagery, aerial photographs and results of magnetometric surveys [1]. Remote sensing and geophysical methods allow to obtain a large amount of data without having to make excavations or even going to places, and take advantage of such openly available sources such as GoogleEarth. The question, however, is how to find effective ways to process such large quantities of data.

Remote sensing methods [3] include deciphering aerial photos, analyzing satellite images, cartography, view shed analysis, point-to-point visibility analysis, photo-modelling and laser surface scanning (light detection and raging). Remote sensing methods allow to achieve a big overall picture of the archaeological site or object of interest, or the area around it. However, given the large scale, none of the above methods ensure elaborate detail.

Geophysical methods [3], such as magnetic and electrical explorations, as well as the use of a ground penetrating radar, involve using relevant equipment on location. On the other hand, these methods provide much more information about what is located under the ground without requiring time-consuming and costly excavations. The most versatile and effective of them is the magnetic method that is discussed in this article.

By far the most widely applicable methods to process the magnetic data are based on the view of earth's magnetic field at a specific point as the sum of three components: the regional field caused by the impact of geologic objects, local anomalies caused by archaeological objects, as well as noise associated with equipment error. Such methods as filtering, averaging observations and trend analysis [4] are used to highlight the captured data that points out to a local anomaly. The use of all of these methods require that researchers actively participate in tuning parameters and analyzing results for accuracy. Many methods do not distinguish anomalies, but merely facilitate further "manual" analysis of data by removing noise from it, which, ultimately, is the task of an expert with experience handling such data.

III. Methodology

The experimental evaluation of several algorithms was

ISBN 978-952-68397-6-9 (paperback), ISBN 978-952-68397-7-6 (PDF)

based on real data:

- 0305 – a Bronze Age settlement on the Tarkhankut Peninsula;
- 0510 – the outskirts of a Hellenistic settlement on the Tarhankut Peninsula;
- 1606 – the outskirts of a Hellenistic settlement on the Tarhankut Peninsula;
- 2405 – part of a Hellenistic vineyard at Cape Oirat, Tarhankut Peninsula [2];
- 2904 – a Bronze Age settlement called Uschelnoe at Lake Donuzlav;
- 3103 – a Hellenistic vineyard in the Ortli estate, Tarhankut Peninsula [2].

The measurements of the magnetic field (the module of magnetic induction vector, to be more precise) were taken in the nodes of a square grid sized 50 by 50 meters, the interval between the nodes being 0.5 meters. Anomalies caused inherent in this method of shooting (for example, changes of the general magnetic field of the earth) were already accounted for and filtered out from the data.

Unfortunately, we were unable to obtain data that could be mapped to the real artifacts found in various places, so the data had to be marked manually with the help of experienced professionals who visually evaluated eye-catching places in the data as anomalous. Such marking does not claim to be absolutely accurate or precise, but it can give us an idea of how close the outcome an algorithm is to what is seen by an expert looking at the data. Marking was performed for each dataset twice: First, the most significant and noticeable anomalies were marked (let us call it an in-depth version); then, the in-depth version was complemented by marking less obvious and eye-catching anomalies (an initial version).

In order to tell the difference between an expert opinion and the outcome of an anomaly detection algorithm proposed in this work, we used the MSE (mean square error) metric, calculated by the formula:

$$MSE = \sum_{i=0}^{N} (l_i - f_i)^2/N$$

where N is the number of points in dataset,

l_i – is the number 0 or 1, corresponding to the absence or presence of anomalies in case of manual labeling by an expert,

f_i – is the number between 0 and 1 describing the probability of occurrence of an anomaly at a given point.

Many algorithms used in our work required defining a similarity measure for elements. A similarity measure in our case is based on the distance between points. Points are vectors of characteristics

$$x = (x_1, x_2, \dots, x_n)$$

Suppose we want to calculate the distance between points x and y. The most common formulae for this are:

- Euclidean distance:

$$\rho(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Squared Euclidean distance:

$$\rho(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2$$

- Manhattan distance

$$\rho(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

All the metrics presuppose that each characteristic is normalized to a common scale, for example, all vector components are between 0 and 1. The metric selection affects results, so all the algorithms were tested using each of the metrics. The dataset points are 3-demension vectors: the x and y coordinates and the *value* of the magnetic field.

To detect an anomaly, we used the following algorithms:

- k-means
- c-means
- DBScan

In general, a clustering algorithm takes a set of n-dimensional points as input. Then, the algorithm splits the dataset into clusters containing similar elements [5].

In this work, k-means and c-means algorithms we used in the same manner as the generally accepted one. We expected that the algorithms would be able to highlight abnormal areas as clusters. Then, we intended to repeatedly run these algorithms and calculate the probability of dataset points being included in anomalous clusters at different values of the algorithm parameters.

The DBScan algorithm differs from k-means and c-means in that it can detect points not belonging to any cluster, which makes DBScan more noise-resistant. DBScan is highly dependent on its parameter values, but, on average, it can steadily group areas devoid of abnormalities into clusters (for example, flat plains). It is based on the assumption that we used DBScan as follows:

- At the first stage, the DBScan algorithm was repeatedly run with different input parameter values. Points, that could not be included in any cluster, were treated as suspicious and belonging to an anomaly, and the probability of falling into an abnormal area was calculated for each point. The obtained results, except for those of unsuccessful runs, were compared with the marked data and evaluated with the help of the MSE metric.
- At the second stage, the points with the probabilities obtained at the previous stage were processed by the DBScan clustering algorithm in order to split all the points into clusters. The resulting clusters with non-zero probability corresponded to the areas where the

archaeological artifacts were believed to be. Silhouette index was used for assessing the accuracy of clustering.

IV. Experiments. Anomaly detection

This section describes the application features and the outcomes of the above-mentioned clustering algorithms.

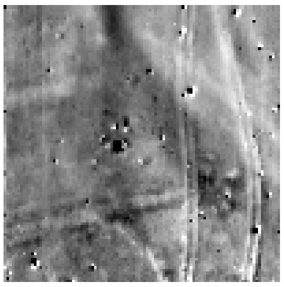


Fig. 1. Initial data

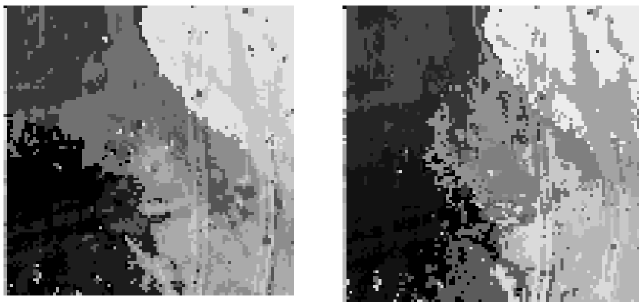


Fig. 2. First cluster separation

A. K-means algorithm

The k-means algorithm splits the input set of points into separate clusters in order to achieve maximum variability of the average values of clusters. For the algorithm to start splitting the points, we specify k – the predefined number of clusters and an initial point (centroid) for every cluster. Next, all the remaining points are assigned to the nearest clusters, and centroids are calculated anew as the average value of each cluster. This process repeats itself until no more points can be assigned to a different cluster.

How can k be defined in the k-means algorithm? One way to do it is to start with a large value for k and keep removing centroids (that is, reducing k) until it no longer reduces the description length. Another way is to start with one cluster, then keep splitting clusters until the points assigned to each cluster have a Gaussian distribution. Still another way is this: The algorithm can be executed on the same data many times, but with different k values; and the most suitable outcome can then be chosen.

This method has several peculiarities. First, as mentioned above, there are different ways to define the distance between points. Second, k-means procedure works better when you select good initial points for cluster centroids. The following two approaches to initialization were tested in the present research:

- Choosing k random points as cluster centroids and distributing all points among clusters with the closest centroids;
- Randomly distributing points among clusters, and then finding cluster centroids as the average value of all the points in a cluster;

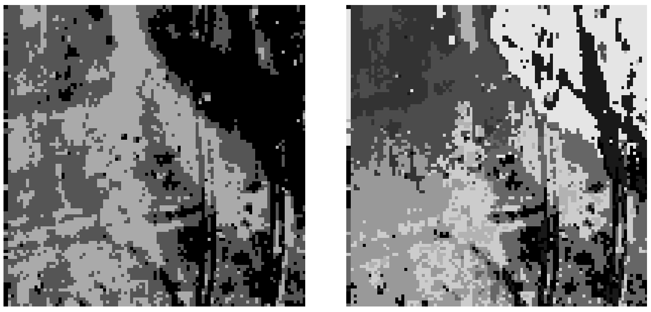


Fig. 3. Second cluster separation

No variety of this algorithm performed well on the available data. Above are the initial data (Fig.1) and some results of the partitioning into clusters (Fig.2, Fig.3). In Fig.1, points are coloured in different shades of grey, and the larger the value of the magnetic field at a point, the lighter is the shade of grey of its pixel. In the figure illustrating the grouping into clusters, each cluster is mapped to a unique shade of gray, and all the dots in a cluster are coloured in the same way.

The image resulting from the second initialization looks better compared to that resulting from the first initialization since the latter adversely affects the final clusters. At the same time, the second method causes clusters to be close to each other in the center of the image. In addition, while the algorithm was being tested at multiple values of k, it was seen that one of the clusters was absorbing the others until all the points belonged to it. The pictures show that, although the outcome of -means does reflect an approximate shape of the road, it is unable to extract the road as a whole into a separate cluster. Other abnormalities, that are visible to the naked eye, are not congregated in proper clusters either. The overall results were so unsatisfactory that we decided not calculate the mean square error.

B. C-means algorithm

The c-means algorithm is a fuzzy clustering algorithm [7]. Its purpose is the same as that of the k-means algorithm: Distributing the input points into clusters so that the midpoints of different clusters differ as much as possible. The k-means algorithm assigns a point to a single cluster, whereas c-means allows one point to belong simultaneously in two or more clusters. The belonging degree of the point i to cluster j is characterized by the value of $\mu_{ij} \in [0,1]$. This value has the following property for all i: $\sum_{j=0}^{k} \mu_{ij} = 1$, where k is the number of clusters. To store these values for all N points, we use the matrix μ with N rows and k columns. This matrix is called a distribution matrix. Just like the previous algorithm, c-

means requires a hypothesis about the number of clusters k. Also, parameters $\epsilon > 0$ and $m > 1$ need to be selected. The former parameter is necessary for stopping the algorithm as soon as the changes of the distribution matrix between two iterations become negligible. The latter parameter m is called a coefficient of fuzziness and determines how fuzzy partitioning will be. The larger the value of m is, the smaller the value of μ_{ij}will be. If m is close to 1, partitioning becomes similar to the outcome of the k-means algorithm. The c-means algorithm isdescribed below:

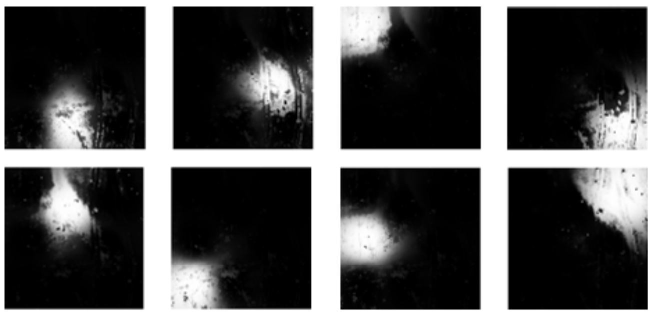


Fig. 4. C-means results (*k=8, m=1.5*)

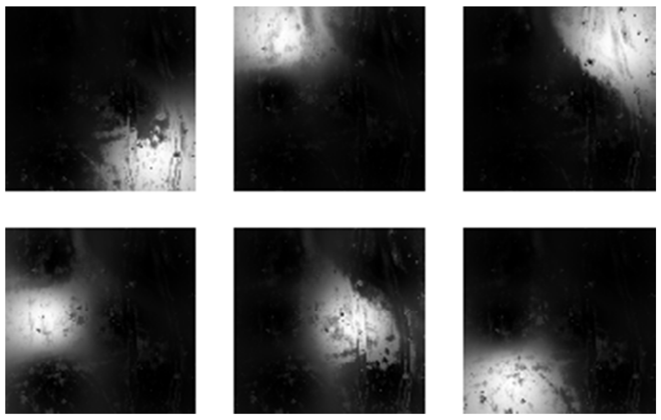


Fig. 5. C-means results (*k=6, m=1.8*)

1) Distributing point to clusters in a random way.

2) Finding centroids in each cluster by calculating the average value of all relevant points, with each point considered with a weight equal to its degree of belonging to the cluster:

$$center_j = \frac{\sum_{i=0}^{N} \mu_{ij}^{m} x_i}{\sum_{i=0}^{N} \mu_{ij}^{m}}$$

3) Recalculating the distribution matrix according to the formula:

$$\mu_{ij} = \frac{1}{\sum_{c=0}^{k} (\frac{\rho(center_j, x_i)}{\rho(center_c, x_i)})^{2/(m-1)}}$$

4) Stopping the algorithm if the matrix distribution has changed less than ϵ, or, if the number of the iterations exceeds the maximum possible number of iterations. Returning to Step 2 if none of the above conditions was satisfied.

The outcome of the algorithm is the current partitioning of points into clusters.

Initialization was performed by choosing random points as cluster centers. The algorithm was run repeatedly with different values of m (the fuzziness factor) and k (the assumed number of clusters). Fig.4 – 7 present the results. Each figure consists of k images, and every image represents one cluster. Each point of the input set corresponds a pixel colored with a shade of gray, and the lighter a pixel is, the higher the belonging degree to the cluster is.

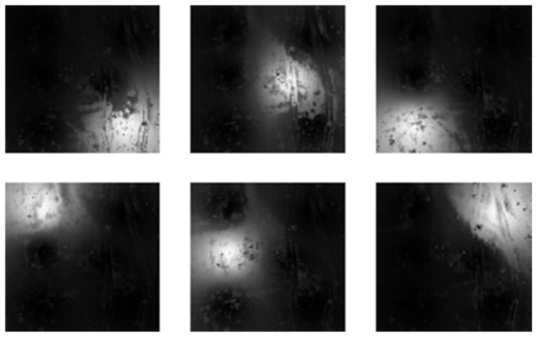


Fig. 6. C-means results (*k=6, m=2*)

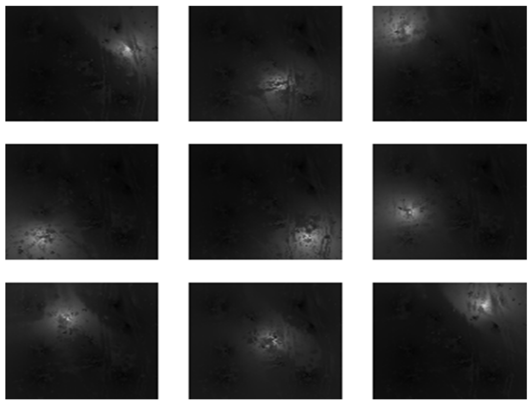


Fig. 7. C-means results (k=9, m=3)

This algorithm showed no satisfactory results on our data. The k-means and c-means algorithms proved to be extremely dependent on the k-parameter and on the initial initialization of clusters. Also, the outcome of this algorithm was as poor visually as that of the previous algorithm, and we decided not to provide MSE estimates.

It became clear at this stage that anomalies could not be sufficiently segregated in clusters in spite of their significant difference from points in the surrounding areas. After the results were analyzed, it became apparent that points without anomalies tended to group themselves into clusters, but anomalous points moved erratically between adjacent clusters because they were not close enough to any of those clusters. It was such erratic points that gave us an idea of how to proceed with our research.

C. DBScan algorithm

The DBScan algorithm is a density algorithm based on the assumption that the density of points is approximately the same within a cluster, but it is higher than that outside the cluster. In addition, if there is noise in the data, the point density in these areas is much lower than that in any cluster. The algorithm requires that points that have a sufficient number of neighbors should be grouped into clusters. In the algorithm, there are two types of points to be assigned to clusters: internal points and boundary points [8]. The algorithm takes into account the fact that boundary points have fewer neighbors than internal ones. What is required to run the algorithm is to choose eps (the radius of the neighborhood) and $minPts$ – a minimum number of neighbors. In order to test the algorithm on our data, we used its implementation in the fpc library of the R language [11].

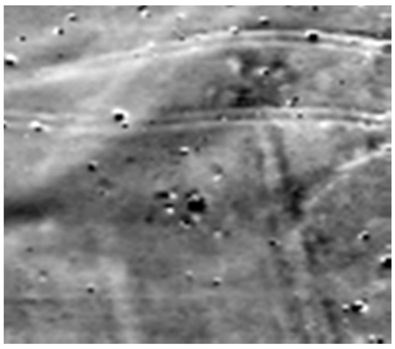


Fig. 8. Initial data

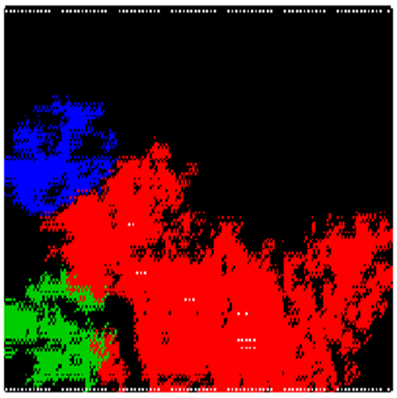


Fig. 9. DBScan results ($eps = 0.5, minPts = 300$)

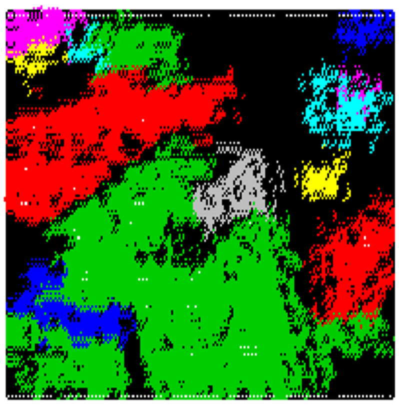


Fig. 10. DBScan results ($eps = 0.25, minPts = 50$)

This implementation allows a user-defined distance metric between points, but the default metric is Euclidean distance. It is this metric that was used in the experiment. The algorithm was run many times with different values of eps and $minPts$. In most cases, all points were either treated as noise, or assigned to one large cluster. Points were treated as noise if there were too many of them within a relatively small area and, vice versa, points were assigned to one large cluster if there were relatively few of them in a large area. However, there were a few algorithm runs that produced more interesting splitting results. These are presented below in Fig.8 – 12. Different colors denote different clusters; boundary points and noise are colored black.

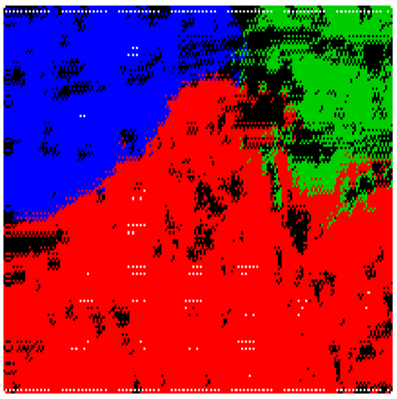


Fig. 11. DBScan results ($eps = 0.2, minPts = 460$)

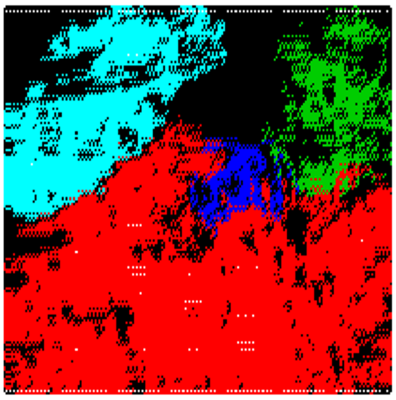


Fig. 12. DBScan results ($eps = 0.15, minPts = 250$)

The pictures show that clusters are areas with more or less homogeneous structure, and anomalies are often marked as boundary points and noise points. In particular, in Fig.11 and 12, we can see shapes of the road, and Fig.9 and 10 show a large light spot of elongated shape at the bottom. Thus, by using multiple runs of the DBScan algorithm, discarding invalid outcomes and analyzing the best of them, we can at least detect areas where there is likely to be some kind of an anomaly.

Here is how the DBScan algorithm was used in our work for finding anomalies: It was run multiple times with different parameters eps and $minPts$; then, the points that were most frequently labeled as noise were analyzed. In addition, the results, where a large percentage of points was related to noise, were ignored. If, for example, 40% of points were noise ($p = 0.4$), the run was considered unsuccessful. It was noticeable that the larger the value of p was, the more sensitive to anomalies the algorithm became. On the other hand, the algorithm execution time increased. Fig.13 - 18 below present the results of the algorithm on different data sets.

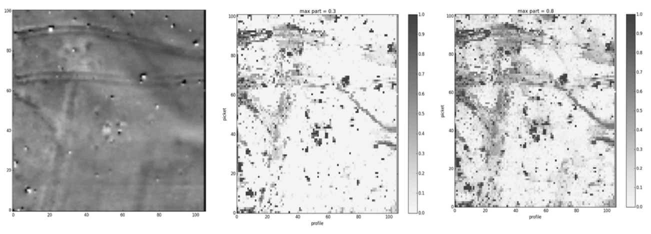

a) Initial data b) $p = 0.3$ c) $p = 0.8$

Fig. 13.Results for the parameters $p = 0.3$ and $p = 0.8$ on area A

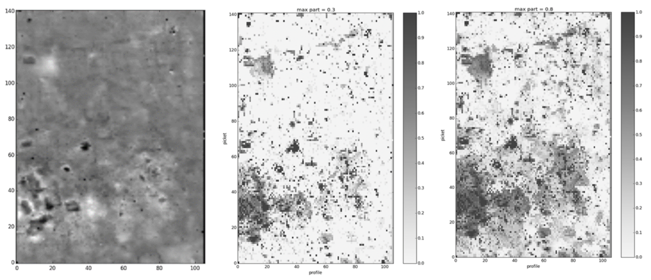

a) Initial data b) $p = 0.3$ c) $p = 0.8$

Fig. 14.Results for the parameters $p = 0.3$ and $p = 0.8$ on area B

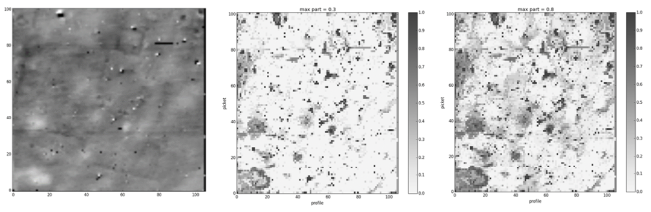

a) Initial data b) $p = 0.3$ c) $p = 0.8$

Fig. 15. Results for the parameters $p = 0.3$ and $p = 0.8$ on area C

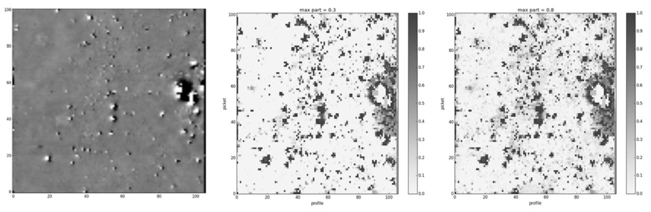

a) Initial data b) $p = 0.3$ c) $p = 0.8$

Fig. 16. Results for the parameters $p = 0.3$ and $p = 0.8$ on area D

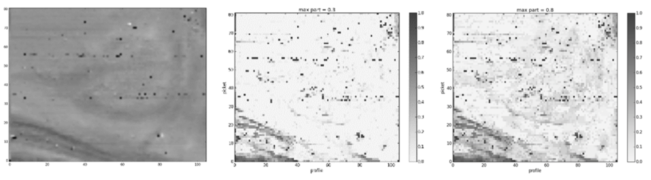

a) Initial data b) $p = 0.3$ c) $p = 0.8$

Fig. 17. Results for the parameters $p = 0.3$ and $p = 0.8$ on area E

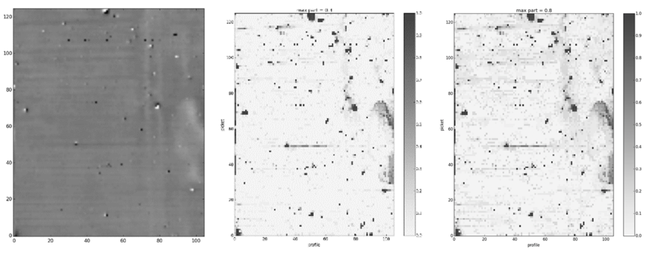

a) Initial data b) $p = 0.3$ c) $p = 0.8$

Fig. 18. Results for the parameters $p = 0.3$ and $p = 0.8$ on area F

From a visual analysis of the results we can make the following conclusions about the algorithm:

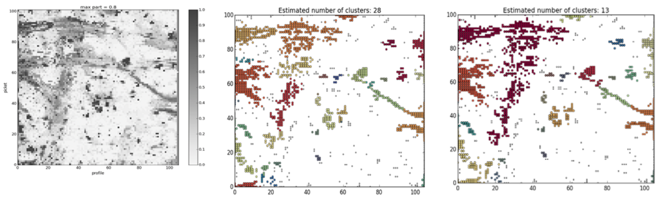
- The can algorithm easily find strongly pronounced local anomalies;
- It detects large local anomalies only partially (for example, as shown in the right part of Fig.16.a). The algorithm invariably marks the border points as anomalous, but, since the middle of the anomalous area has almost identical values, these values group themselves into a cluster and are not treated as noise;
- As shown in Fig.17, the algorithm does a good job of detecting the long anomaly in the lower left corner, but it fails to detect the less bright circle-shaped anomaly at $p = 0.3$. At $p = 0.8$, however, at least the lower semi-arc becomes detectable. The loss of such anomalies can be avoided by processing the data in two stages: first, bright anomalies should be detected and left out of the anomalous area which is being looked at. The removal of bright anomalies makes it possible to then detect medium bright anomalies because they became more noticeable.

The results of the algorithm were then compared with the hand-crafted marking. As was mentioned earlier, the marking was presented in two versions: the in-depth version that contained only the most visible anomalies and the initial version where all eye-catching abnormal areas were marked. The idea behind this comparison is to see to which extent the opinion of an expert in the field coincides with the outcome of the algorithm. The MSE metric was used for the comparison. Table I shows the results of the comparison for all the datasets, for two marking versions (the in-depth one, the initial one), at the algorithm parameters $p = 0.3$ and $p = 0.8$.

TABLE I. THE RESULTS OF THE METRICS MSE

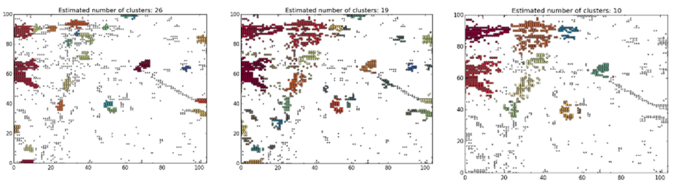
MSE	0305	0510	1606	2405	2904	3103
In-depth & p=0.3	0.0635	0.0610	0.0481	0.1188	0.0455	0.0325
Initial & p=0.3	0.1205	0.0706	0.0578	0.1167	0.0765	0.0560
In-depth & p=0.8	0.0706	0.0701	0.0593	0.1387	0.0797	0.0405
Initial & p=0.8	0.1115	0.0773	0.0670	0.1298	0.0961	0.0591

At the previous stage, the probability of the presence of an anomaly was calculated for each point in the studied area. In order to extract individual anomalies from the dataset, it made sense to use the clustering algorithms to separate the abnormal areas from each other. In other words, the task was to cluster a set of points, each of which is described by three values: x, y are coordinates, and $freq$ is the probability of an anomaly. Since anomalies often have a complex shape, the DBScan algorithm was utilized again. Earlier in this work, we were interested in points outside clusters; this time, the algorithm is going to be used for clustering anomalous points themselves. What was chosen as the initial data was those points that tended to be treated as noise ($freq > 0.4$). It remains to understand how to set the values of eps and $minPts$ coefficients used in the algorithm. Fig.19 shows some examples of running the algorithm.



a) anomaly probability b) eps =3, $minPts$ = 9 c) eps = 4, $minPts$ = 12

Fig. 19. Example of running DBScan for the final anomaly detection



a) $silhouette$=0.542 b) $silhouette$=0.603 c) $silhouette$=0.623

Fig. 20.The best running examples

It is obvious that different values of the parameters will lead to different algorithm outcomes, and not all of them will be of satisfactory quality. For instance, it can be seen in Fig.19.c that most anomalies grouped themselves into one large cluster on the left. The work [9] lists 11 indices that it can be utilized to evaluate the quality of clustering in terms of compactness and distribution of clusters. The index selected for the case at hand is the silhouette index that is based on the information about the average distance between points within a cluster and between clusters. The shorter the average distance between points in a cluster is, and the longer the distance between clusters themselves is, the closer the index is to 1. If the quality of splitting is low, the value of the index will be negative and close to -1. Look at Fig.20 below for the best examples of algorithm runs in this respect.

As shown the examples in Fig.20, the use of the silhouette index for optimization causes only small-size and compact anomalies to merge into clusters, with a significant proportion of points being treated as noise. The latter factor can also be taken into account with the help of the following formula:

$$sil\_part = silhouette \cdot (1 - \frac{noisepoints}{allpoints})$$

Unfortunately, we have no marked data, that is, information about the location of real anomalies, and this makes it hard to find a good detection algorithm for anomalies, as it is difficult to evaluate obtained various results. If such marked data was available, it would be possible to estimate various outcomes by using such numerical characteristics as precision and completeness. What we would have to do would be to run algorithms on marked data and assess the outcome in terms of precision and completeness.

VII. Conclusion

In this work, we have made an empirical attempt at understanding how clustering algorithms can be used in magnetometric analysis. This paper looks at the applicability of three algorithms: k-means, c-means, and DBScan. The first two algorithms did not demonstrate convincing results, while the results of applying the DBScan algorithm were quite interesting.

When applied for the first time, the DBScan algorithm detected some points that were outside any of the clusters. These points were considered to be of interest because of the assumption: The higher the probability of the presence of an anomaly at a point is, the more often the algorithm will treat this point as noise. Then, the DBScan algorithm was reapplied to points that, with high probability, were part of noise, with the view to obtaining clusters consisting of such abnormal points. Since we are not in possession of marked data, we cannot calculate the quality of anomaly detection; however, the algorithm was run on many datasets, and a visual assessment by experts confirmed a sufficient level of accuracy. As an application for the future, this approach can be used to split anomalies into types (e.g., metal objects, walls of buildings, etc.) and focus on a particular type of them.

References

[1] T.N. Smekalova, F.N. Lisetsky, O.A. Marinina, A.V. Chudin, A.S. Garipov, “A study of spatial Organization of Ancient Land-use by Geoarchaeological Methods”, *The Bulletin of Archaeology, Anthropology and Ethnography*, 2015, No. 1(28), pp. 150-160.

[2] T.N. Smekalova, B.W. Bevan, A.V. Chudin, A.S. Garipov, “The Discovery of an Ancient Greek Vineyard”, *Archaeological prospection,* 2015, DOI: 10.1002/arp.1517.

[3] T.N. Smekalova, E.B. Yatsishina, F.N. Lisetsky, A.V. Chudin, A.S. Garipov, A.E. Pasumansky, R.S. Ketsko, “High Technology in Archaeology”, *Bosporus Studies*, Kerch, 2015.

[4] I.N. Koshelev, V.P. Dudkin. “Methods of Comprehensive Interpretation of the Results of Magneto-metric Studies of Archaeological Sites”, *East European Archaeological Journal*, (3(16)), May-June 2002.

[5] A.A. Barseghyan, *Analyzing Data and Processes.* St. Petersburg, 2009.

[6] J.A. Hartigan and M.A. Wong. “A K-means Clustering Algorithm”, *Journal of the Royal Statistical Society*, Series C (Applied Statistics) 28.1 (1979): 100–108.

[7] J Valente de Oliveira, Witold Pedrycz. *Advances in fuzzy clustering and its applications.* Wiley Online Library, 2007.

[8] M. Ester, “A density-based algorithm for discovering clusters in large spatial databases with noise”, *In Proceedings of 2-nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996.

[9] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao and Junjie Wu, “Understanding of internal clustering validation measures”, *In Data Mining (ICDM), 2010 IEEE 10th International Conference*, pages 911–916. IEEE, 2010.

[10] Anil K. Jain and Richard C. Dube, *Algorithms for Clustering Data.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[11] Hennig, C. *fpc: Flexible Procedures for Clustering.* R package, version 2.1, 2015.