Applying trajectory mining in medical image data

Roland Ellerweg[1], Katharina Hofer[1], Artur Khromov[1], Peter Voigt[2] and Sebastian Fischer[3]

[1]Fraunhofer FIT
Sankt Augustin, Germany
roland.ellerweg@fit.fraunhofer.de

[2]University Hospital Leipzig
Leipzig, Germany
peter.voigt@medizin.uni-leipzig.de

[3]University Hospital Frankfurt
Frankfurt, Germany
sebastian.fischer@kgu.de

ABSTRACT

During evaluation of CT or MR images radiologists navigate through a volume in different orientations in order to detect a disease. While doing so, they leave a trail, which might hold valuable information for other clinicians. Unfortunately, current systems do not analyze this trail for certain motions or potential patterns. In this work we developed and implemented different strategies to infer the manifestation of a disease from the trail of inspection. Furthermore we evaluate the effectiveness of these strategies by conducting an experiment in which clinicians had to find a tumor in several cases. The results suggest that inferring suspicious areas from the trail is possible.

Author Keywords

Radiology; Data Mining; Trajectory; Medical Imaging; User Experiment.

ACM Classification Keywords

H.2.8. Database Applications: Data Mining

H.1.2 User/Machine Systems: Human Factors

J.3 Life and Medical Sciences: Medical Information Systems

INTRODUCTION

In the field of radiology, imaging methods such as computer tomography (CT) or magnetic resonance imaging (MRI) are used to help clinicians in diagnosing diseases. In addition they can be used to guide certain therapies such as minimal invasive cancer therapies.

MR images are acquired in different planes by either acquisition in a certain orientation (esp. axial, sagittal, coronal) or reconstructed from a 3D volume dataset like CT in imaging. This allows the user to navigate the anatomical structures by either clicking or scrolling in one of the generated planes. Typically in a DICOM viewer axial, sagittal and coronal orientations are illustrated next to each other and a click in one window basically represents a "jump" to the corresponding location in the other planes. Scrolling in one of the planes lets the program load the next slice for the requested direction.

While the clinician navigates through a case he produces a trail. This trail consists of a sequence of points visited at a certain time. In physics this sequence is called a trajectory and in recent years, particularly with the introduction of location aware devices, scientists try to apply data mining methods to get information out of trajectories [4,6,7,8,9,10,11,12,13,14]. This information can help in the classification of trajectories, the detection of outliners, or the prediction of the next location. Unfortunately trajectory mining methods have not been applied in the medical context.

In our work we implement different strategies to infer the manifestation of a disease from the trajectories a clinician produces while working with a DICOM viewer. Among the strategies are common trajectory data mining methods known from literature but also custom ones we developed in this work. We evaluated all strategies in an experiment where we asked several clinicians to localize malignancies in CT datasets to receive real world trajectories.

Finding a disease from trajectory data has several potential applications. In the clinical workflow diagnostic findings are predominantly documented in an either handwritten or digital form. Clinicians interested in an already diagnosed case use this form to get a first overview of an image dataset. In a second step, they open the images in a DICOM viewer and navigate to the diagnostic findings mentioned in the form. By analyzing the trajectories this whole process can be simplified as it would enable the implementation of new interfaces which suggest certain regions in a volume to the clinician, similar to the "customers who were interested in this item were also interested in …" dialog known from web shops. Hence a clinician recapitulating an already diagnosed case would not spend so much time to search for the main findings in a volume dataset.

Apart from the benefits to the clinical workflow inferring the location of a disease from trajectory data can also serve as an input for segmentation algorithms. These algorithms typically require a user to select a seed point which marks some region of interest within the images. Consecutively an algorithm of the typically "region growing" type is launched from this seed point input to calculate the volume of e.g. a tumor, lesion or organ. Replacing this manual seed point selection with the inferred location could simplify this step.

Finally the knowledge about the location of a disease could be used to optimize the image-downloading process in teleradiological scenarios. Therefore all slices showing the disease are downloaded first. In a second step all other slices are downloaded. This would be useful in regions with weak internet connection.

The rest of this paper is structured as follows. Section 2 describes related work, particularly in the area of trajectory data mining in greater extent. Section 3 explains the

PervasiveHealth '17, May 23–26, 2017, Barcelona, Spain

ACM ISBN 978-1-4503-6363-1/17/05…$15.00
https://doi.org/10.1145/3154862.3154889

methodology of the experiment but also introduces new strategies to find point of interests that were not mentioned in related work. Section 4 presents the results from the experiment conducted. Eventually conclusions are drawn in section 5.

RELATED WORK

Most data mining methods on trajectory data came up with location-aware devices about a decade ago. The GPS-Module of these devices stores triples representing the coordinates as a point P(x|y) together with a timestamp t. A series of triples forms a spatial trajectory which is used for data mining. There are various application scenarios such as the prediction of mobility [6, 7, 12] or the recommendation of suitable locations in a particular context [4,13].

Under the hood these applications use stay point detection-algorithms [6, 8], sometimes also referred to as point of interest extraction [13]. The idea behind these algorithms is to find a meaningful place S in one or more spatial trajectories. Li proposed a first stay point algorithm in 2008[8] (cp. Fig. 1). Basically, Li's algorithm checks first if a sequence of points on the trajectory is within a given distance and secondly, if the points have been visited within a certain time threshold. If both checks succeed a stay point is calculated as the mean of the point sequence. Eventually the algorithm continues on with the processing of the remaining points of the trajectory until all stay points are detected.

```
Input: A GPS log P, a distance threshold distThreh
       and time span threshold timeThreh
Output: A set of stay points SP={S}
1. i=0, pointNum = |P|; //the number of GPS points in a GPS logs
2. while i < pointNum do,
3.     j:=i+1;
4.     while j < pointNum do,
5.         dist=Distance(pi, pj); //calc. the distance between two points
6.         if dist > distThreh then
7.             ΔT=pj.T-pi.T; //calc. the time span between two points
8.             if ΔT>timeThreh then
9.                 S.coord=ComputMeanCoord({pk | i<=k<=j})
10.                S.arvT= pi.T; S.levT=pj.T ;
11.                SP.insert(S);
12.            i:=j; break;
13.        j:=j+1;
14. return SP.
```

Figure 1. Stay point detection algorithm [4].

Li successfully used the algorithm to find meaningful places of users. However, as the algorithm applies the threshold only between a starting point and its successors not all necessary points are considered for a stay point. E.g.: if a trajectory contains four points P1, P2, P3 and P4 and the time and distance thresholds apply between P1 – P3 and between P2 – P4 only one stay point for P1 – P3 would be created. P4 might be taken into account for a second stay point or is ignored. The stay point detection proposed in [12] solves this issue by building a cluster of points first. All points of the cluster are than compared to the next point on the trajectory. In case of the four points mentioned previously, a cluster for P1-P3 is created first following the idea of the stay point algorithm above. Secondly, the threshold between P4 and all points of the cluster are evaluated. Since the thresholds apply between P2 and P4, P4 is added to the cluster. Eventually a stay point is calculated for all points in the cluster.

Another idea to extract a stay point from a trajectory is presented in [13]. Here, from a set of trajectories the start- and end-points are filtered first. Secondly, points that are within a certain distance threshold ϵ are clustered. The more points fall within the cluster the more interesting is the location.

All stay point algorithms have advantages in different use cases. For instance, if the points are very dense in a given trajectory one might not want to create a huge cluster from which a stay point is created. On the other hand, if most of the points are dense, but there are bigger gaps between point groups one might favor the cluster based method. However, the implementation of stay point detection algorithms on the trajectories which clinicians produce while working on a DICOM series raises several questions. Firstly, the presented algorithms use only 2D locations whereas clinicians browse through 3D data. Hence the algorithms need to be adapted accordingly. Secondly, it is unclear which stay point algorithm has the better performance in matters of accuracy. Thirdly, descent time and distance thresholds have been established in GPS based applications whereas for the present use case these thresholds are unknown. Lastly, other approaches such as the amounts of visits at a certain location need to be evaluated against the presented algorithms.

In the following the setup of an experiment conducted for this work is described. The output of this experiment answers the questions above.

METHODOLOGY

To find out whether or not stay point detection algorithms can identify the location of a disease from motion patterns we employed the following methodology:

- Create a scenario (1)
- Create a questionnaire (2)
- Select and extent a DICOM viewer (3)
- Select an environment (4)
- Implement stay point detection algorithms (5)

Steps 1 - 4 form the basis of a user experiment which we conducted to collect trajectories. Therefore steps 1 and 2 describe the content of the experiment and how test subjects were acquired. Step 3 and 4 cover the technical aspects, such as the software/hardware setup. Eventually step 5 deals with the implementation of the stay point algorithms mentioned in related work and the implementation of other stay point detection algorithms we developed in this work. In the following each step of the methodology will be described in greater detail.

Create a Scenario (1)

To acquire trajectory data for the stay point algorithms we formulated a scenario in a first step. In the scenario five different tumor cases were presented to the user. Users were asked to find the tumor and take a screenshot once the tumor is visible. We briefed the users and let them know, that all cases contain at least one tumor in either the liver, the lungs or the kidneys. They did not know however which particular organ was affected in a case. Hence, they had to scan all organs.

The test data for the experiment were taken from the image repository of the EU-funded project "GoSmart". GoSmart is a web platform, which enables clinicians to simulate minimally invasive cancer treatments. The project contains approximately 160 cases with tumors in different organs such as liver, lung or kidney. All data within the GoSmart project was pseudonymized. Furthermore participating patients agreed on the usage of their data for scientific purposes.

From the approximately 160 tumor cases five were selected randomly for the experiment – two kidney cases, one lung case and two liver cases. The kidney cases each comprised a single renal cell carcinoma which was clearly visible by a strong contrast enhancement and a tumorous appearance. The liver cases contained multiple big metastases, in one case they were even strongly contrasted by a chemotherapeutic agent. The lung case contained a solitary lung metastasis

Create a Questionnaire (2)

Candidates of the experiment were obliged to have skills in the diagnosis of tumors in particular organs. To obtain only candidates with these skills we developed screening questions to record demographic information such as current position, work experience and self-rating in tumor detection. All candidates had to fill out this screener prior to the experiment. Figure 2 shows an example of a screening question.

(3) On a scale from 1 – 5, how would you rate yourself in the diagnosis of tumors in liver, lung and kidney (1: very bad, 5: very good):

- ☐ 1 (very bad)
- ☐ 2 (bad)
- ☐ 3 (average)
- ☐ 4 (good)
- ☐ 5 (very good)

Figure 2. Demographic questions.

Apart from the demographics we also wanted to get more information about how users are working with DICOM viewers in a specific context and how they would react to new interface concepts, such as the "customers who were interested in this item were also interested in …"-dialog mentioned before. For the former we developed questions which described a particular case, e.g.: a HCC case. Then we asked the candidates which plane (axial, coronal, sagittal) they would use to find the tumor (cp. Fig. 3). For the latter we developed a simple paper prototype and asked the users if they consider it as helpful in their daily work (cp. Fig. 4).

(6) Given a patient with a hepatocellular carcinoma (HCC), which of the following windows would you use to diagnose the tumor (multiple answers allowed):

- ☐ Axial
- ☐ Sagittal
- ☐ Koronal
- ☐ 3D

Figure 3. Question for disease depending direction.

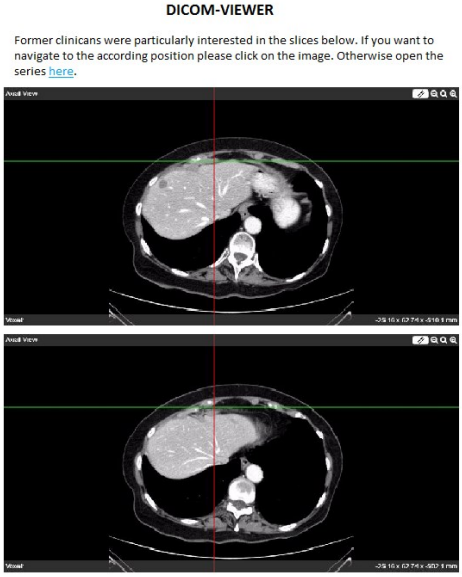

Figure 4. Screen prototype.

Select and Extent a DICOM Viewer (3)

The main criteria of the DICOM viewer selection were ubiquitous access to the images, ease of use and extensibility as components to track the movements of clinicians had to be facilitated. We compared several DICOM viewers from different studies and papers [1,2,3,5]. Eventually we selected the DICOM viewer architecture published in [2]. The architecture excels in teleradiologic scenarios which is particular useful for the acquisition of test subjects in remote locations. Figure 5 shows the frontend of the implemented viewer.

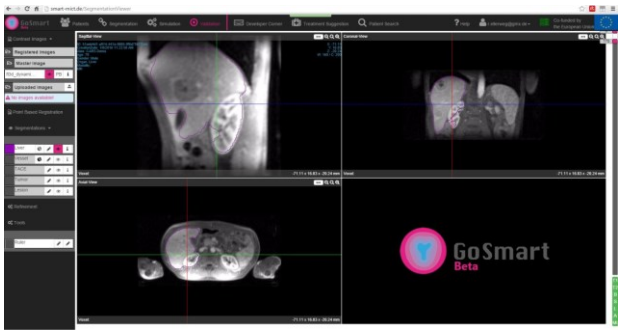

Figure 5. Frontend of the implemented viewer [2].

To cater for the experiment we extended the viewer architecture by several modules. These modules are:

Movement tracker module: The responsibility of the movement tracker module is to record the movement of the user while working on a case. To achieve this goal it contains a data structure to persist the position $P(x|y|z)$, a timestamp and metadata about the case such as case id and series id. The data structure is filled on the client side using Typescript. Once a user switches to another case all data is send to a server which persists the acquired trajectory in an SQL database.

Screenshot module: The screenshot module enables the user to take screenshots of the current screen. To implement it we cared for easy access of the necessary controls, as a weak usability would have a great impact on the movement pattern. A usability engineer verified the usability of the controls. Once a screenshot is taken it is saved on the server side together with the current position. The stored data is used to build the ground truth for the algorithm validation.

Data acquisition module: The data acquisition module prepares the data for the processing by different algorithms. Therefore the data saved by the movement tracker module is loaded first. Secondly, the trajectory data is enriched by the ground truth which is defined as a bounding box around the screenshot position. The size of the bounding box is based on typical tumor sizes found in our test data (~20mm). With this information we can check, if the calculated stay point is close to the point the user found interesting. This does not necessarily need to be the tumor but some other structure which might be suspicious. Finally the data is returned to the caller of the module.

Select an Environment (4)

The server which hosted the web-based DICOM viewer had two Intel Xeon CPU E5-2665 processors with a 2.4GHz clockrate and 32 GB of RAM. The operating system was windows server 2012. Internet information services (IIS) was used as a web server.

The experiment was done with two participants from the University Medical Center in Leipzig and three from the University Hospital in Frankfurt.

Implement Stay Point Detection Algorithms (5)

To evaluate the different stay point algorithms we implemented the ones mentioned in related work section - Li's stay point algorithm and the extension using clusters - first. Therefore we extended them to care for 3D volumes instead of 2D maps which basically involved changes in methods calculating distances or mean coordinates. After that we implemented a testbed in which we tried different distance threshold which correspond to the minimum and maximum tumor size (20mm and 30mm). For the time threshold we analyzed the data and found good results for 3, 4 and 5 seconds.

In a next step we implemented further strategies to find stay points. It follows a more detailed description of these strategies.

Outside Ball Algorithm

The idea of the Outside Ball Algorithm is based on the assumptions, that the movement of clinicians is usually mainly in one direction. Without loss of generality it is assumed, that the movement is basically only in z-direction. Then, tuples $(1/z_i, t_i)$ are considered, where z_i refers to the position at time t_i for each $i=1, \ldots n+1$. Out of these tuples difference tuples $v_i = (1/s_i, q_i)$ are built by

$$s_i = z_i - z_{i+1}$$

$$q_i = t_i - t_{i+1} \text{ for each } i=1, \ldots, n.$$

The reason for building the tuples like that is, that some common ground has to be found to measure the difference between the tuples. Since q_i has to be big for getting a stay point but on the other hand s_i has to be small, $1/s_i$ is getting big too in case of stay points. By using the tuples v_i we can look for huge differences. To measure the differences, norms, i.e. the Euclidean norm $||.||_2$, can be used. Each $||v_i||_2$ which tells something about the difference to the neighbor then can be used – if big enough – to find a stay point.

Input: List of tuples $(1/z_i,t_i)$ for i=1,…,n+1
Constant a_1 for calculating threshold T
Output: List of stay points (x_i,y_i,z_i) for j=1,…m
1. **for** i=1,…,n
2. calculate tuples $v_i:=(1/(s_i-s_{i+1}),t_i-t_{i+1})$
3. **end**
4. Calculate threshold T = mean(v_1,…,v_n)+a_1*std(v_1,…,v_n)
5. **for** i=1,…,n
5. **if** $||v_i||_2 > T$ **then**
6. add (x_i,y_i,z_i) to list of stay points
7. **end**
8. **end**
9. return list of stay points

Figure 6. Outside Ball Algorithm.

The calculation of the threshold T is semi-automatic. On the one hand, it depends on the trajectory, on the other hand, some a-priori chosen parameter a_1 comes into play. The idea

behind that choice, is to use the distribution of the tuples v_i for calculating the threshold in order to simplify the choice of the a_1. A suitable choice of the constant a_1 is expected to be in the set {2,3,4}.

The reason for adding (x_i,y_i,z_i) in line 6 of the Outline Ball Algorithm is to enforce that only points (slides in the viewer) which have been visited can be labeled as stay points.

Remark: In order to avoid the labeling of points as stay points in case of interruption of the clinician (e.g. by a telephone call) a modified threshold can be used (line 5 of the algorithm) by

$T_1(a_1) < ||v_i||_2 < T_2(a_2)$

and $T_i(a_i) = mean(v_1,\ldots,v_n)+a_i*std(v_1,\ldots,v_n)$ *for* $i=1,2$

where mean is the mean value and std the standard deviation. This can be imagined as circular ring around each v_i, where all such circular rings with bigger then T_1 and smaller then T_2 are marked as stay point. Since the norm $||v_i||$ increases if q_i increases, an increased time difference (e.g. due to a phone call) isn't considered any longer as stay point.

The drawpoints of the Outside Ball Algorithm is, that only each point and its neighbors are considered. That means no clusters can be found. Nevertheless, in the given application the crucial point is to find slides with suspicious structures and not to find clusters.

The use of different norms, i.e. especially the use of the supremum norm or the Manhattan norm is possible.

Outside Ball for three-dimensional points

The presented outside ball algorithm is based on the assumption that clinicians mainly move in one direction. However, an extension to three-dimension (or even two) is possible if this assumption should turn out to be wrong. In the extension, tuples *(1/x,1/y,1/z,t)* are considered. The difference tuples are defined by

$v_i=(1/(x_i-x_{i+1}),1/(y_i-y_{i+1}),1/(z_i-z_{i+1}),t_i-t_{i+1})$ for $i=1,\ldots,n+1$.

The algorithm is then analogue to the Outside Ball Algorithm given above. Assuming, that from step i to $i+1$ mainly only one direction changes, $||v_i||_2$ gets especially large if the difference of the change in the point is very small, or the time difference is very high.

Remark: In order to avoid a division by zero, the following strategy – explained for the case $x_i=x_{i+1}$, $y_i\neq y_{i+1}$ $z_i\neq z_{i+1}$ – is used. In that case v_i is set to

$v_i = (1e16, 1/(y_i-y_{i+1}),1/(z_i+z_{i+1}),t_i-t_{i+1})$.

Trajectory Subsection Analyzer

First we prepare the data for analyze. For this purpose, we delete the stable positions (the coordinates do not change) from the axis if the user scrolls the different axis. We call this process data normalization, and deleted points – false stable points. After data is normalized, we calculate the first derivate: $x' = (x_{i+1} - x_i)/(t_{i+1} - t_i)$, where x_i refers to the position on the X-axis at time t_i. The derivate shows us the speed of changing. It could be negative or positive. Negative sign shows that the user scrolls down, positive – up. Then we assume that the searching procedure consists from constant changing of direction and most interesting coordinates are visited more often. We use the derivation to find most visited segments and segment with the lowest speed of scrolling. By combining this information, we can determine the segments of interests and assume that tumors are within the segments of interest. We take the last point of each segment as a position of tumor. The final step is to combine the results from all axes. We do it by building all possible tuples (x_i, y_i, z_i) from the previous step and select those that belong to the user's trail.

Input: List of tuples (x_i,y_i,z_i) with corresponding time values t_i for i=1, … n
Output: List of stay-points S with (x_j,y_j,z_j) for j=1,…m

1. Calculate the reduced data u,v and,w for each component by
2. for i=1,…,n-1 do
3. If $x_i \neq x_{i+1}$ then u.add(x_i)
4. If $y_i \neq y_{i+1}$ then v.add(y_i)
5. If $z_i \neq z_{i+1}$ then w.add(z_i)
6. If $x_i == x_{i+1}$ and $y_i == y_{i+1}$ and $z_i == z_{i+1}$ then
7 u.add(x_i); v.add(y_i); w.add(z_i)
8. Calculate the approximation of change for each reduced data by
9. for l=1,…,L-1 do
10. $u_l' = (u_{l+1}-u_l)/(t_{l+1}-t_l)$
11. $v_l' = (v_{l+1}-v_l)/(t_{l+1}-t_l)$
12. $w_l' = (w_{l+1}-w_l)/(t_{l+1}-t_l)$
13. Find indices where sign in u_l' changes and set them in $L_{index} = (l_1,\ldots, l_q)$
14. Find repeated segments R = $((u_1, u_2), (u_3, u_4), \ldots, (u_{r-1}, u_r))$ by
15. $r_{prev} = [u_1, u_{last}]$
16. for l= l1,…,lq-2 do
17. $r_{temp} = [u_l, u_{l+1}] \cap [u_{l+1}, u_{l+2}]$
18. If $r_{temp} \cap r_{prev} \neq 0$ then $r_{prev} = r_{temp}$
19. Else R.add(r_{prev}); l--;
20. Find average speed of changing S = $(s_1, \ldots, s_{q-1})$ by
21. for l= $l_1,\ldots,l_{q-1}$ do
22. s_l = mean($[u_l, u_{l+1}]$)
23. Find intersection between repeated segments and slowest segments P = $((u_1, 24.u_2), (u_3, u_4), \ldots, (u_{p-1}, u_p))$
24. Create list X of possible stay points in x-direction X = $(u_2, \ldots, u_p)$
26. Repeat steps 13-23 for V and W to build Y, Z
27. Check each combination of tuples (x,y,z) from (X,Y,Z) if it is a trajectory point.
28. Each (x,y,z) which is a part of the trajectory, is added to the list S of stay points

Figure 7. Trajectory Subsection Analyzer.

Last point visited strategy

The last point visited strategy assumes that the user closes the viewer session once he found the desired information. Therefore the last location the user visited is simply taken as a stay point. An already visible disadvantage of this strategy is that is cannot detect multiple stay points.

Most visited points strategy

The most visited point strategy assumes that the user navigates through more interesting regions more often than

in less interesting regions. Therefore all locations gain a counter which increments once the user visits it. The points which were visited most are considered as a stay point.

RESULTS

In the result section we first discuss the results of the screener. Secondly we compare and present the results of the stay point algorithms mentioned in the previous chapter.

Screener Results

To gain real world trajectory data we acquired five users from two healthcare institutions. The majority of users worked as radiologists (4). One user worked in another medical field but used radiologic images on a regular basis. The work experience of the users ranged from 1-3 years (2), 3-5 years (2) and > 5 years (1). Furthermore the users rated their tumor diagnostic skills with below average (1), average (1), good (2) and very good (1).

To diagnose a HCC the users stated they would preferably use the axial plane (1), the axial and coronal planes (2), or the axial, sagittal and coronal planes (2). In case of a herniated disc the majority of users prefer the axial and sagittal plane (4). One user stated he would only use the sagittal plane. No one considered a 3D view as helpful in diagnostics.

The interface prototype which suggests a particular location to the user was considered as useful (3) or very useful (1). One user stated that there is no use in such a dialog. It turned out that this user did not consider this dialog as a possibility to jump to a certain location and hence was worried that he would miss auxiliary findings.

Algorithm Evaluation

Table 1 shows the overall accuracy of the different stay point algorithms. In the table we refer to the stay point algorithms mentioned in related work by their original title and the main author in brackets. For the algorithm introduced by Li et.al. [8] this is "StayPointDetection (Li)", for the extension using clusters [12] it is "ParkingCandidateDetection (Yuan)". Further algorithms introduced in methodology section simply keep the name.

In case the algorithm was executed using different parameter sets we mention those in the 2nd column and explain the parameters in the table legend. The following columns show how many stay points were found inside (*Positives P*) or outside (*False Positives FP*) the ground truth defined in the previous section, as well as the corresponding accuracy rate (*R*). We consider a stay point as positive if at least one direction, either X, Y, Z, falls within the ground truth. This corresponds to the behavior of the clinicians who use one direction at a time to search for a disease and hence most of the time only one plane actually shows the disease. Since a high stay point accuracy does not necessarily mean that all screenshots were found, we indicate further how many were found by an algorithm (*Found F*), how many were not found (*Not Found - NF*) and the screenshot accuracy rate (*R*). In total 49 screenshots were taken by the clinicians.

At best, the stay point detection introduced by Li et. al. [8] was able to detect 58.09% stay points correctly. Furthermore the algorithm could identify 44 out of the 49 screenshots taken. Yuan's parking candidate detection performed slightly better in the stay point accuracy rate (66.67%). Unfortunately less than half of the taken screenshots were detected (42.86%). Due to the dense trajectory points the algorithm often builds only one huge cluster from which the stay point is calculated.

In comparison to the other algorithms, the Outside Ball algorithm found much more potentially stay points. The result shows, that the Outside Ball 1D algorithm perform worse, than the Outside Ball 3D algorithm, but at least better than expected, since only one clinician in the experiments used the z-axis only. Nevertheless, due to the structure of the algorithm (for 1D), it is sufficient to have a bigger time difference between two neighboring points to ensure, that a tuple will be marked as stay point. The best rate for the Outside Ball 1D algorithm is about 61.73% stay point accuracy with a 89.8% rate for the detection of all screenshots. The Outside Ball 3D for the best choice of the parameters yield 65.15% (stay point accuracy) with 89.8 % (detection of all screenshots). Even though, the parameters a_1 and possibly a_2 have to be set, the range for setting a_1 is quite clear, whereas the setting of a_2 can be avoided.

The trajectory subsection analyzer found 100 stay points from which 64 were at least in one direction close to the screenshot. The screenshot accuracy rate was at 81.63%. It is remarkable that the algorithm achieved the quite good rates without any parameterization.

The last point visited strategy performed best in stay point accuracy rate (92%). Apparently the probability that the user closes the DICOM-viewer while looking on a suspicious structure is high. This is even the case, when time stamps for screenshot and for closing the DICOM-viewer lie further apart. Hence the user revisited the suspicious coordinates before exiting the session. Unfortunately, the screenshot accuracy rate of 63.27% is weaker than the one of other algorithms. This is due to the fact that the last point visited strategy can only detect one screenshot per case as a maximum. Similarly the most visited points strategy achieved a quite high stay point accuracy (79.17%) but performed weak in the detection of all screenshots (71.34%).

Overall the suitability of the presented algorithms largely depends on the use case. If the goal is to find nearly all structures a user found interesting one might want to use the outside ball 3D algorithm which performed best in the tumor accuracy rate and also achieved good results in the stay point accuracy rate. In case good parameter settings are not known one can use the trajectory subsection analyzer. Finally, if one stay point is enough per case the last point visited strategy can be used which achieved the highest stay point accuracy rate.

For the implementation of new interfaces which suggest certain locations to the user both, stay point accuracy rate

and screenshot accuracy rate matter. Hence the outside ball algorithm is the most appropriate for this use case. If two images are selected by this algorithm at random (e.g. Outside Ball 3D with parameters a_1= 5.0; a_2=1e16), the probability of selecting one image with a suspicious structure is 87% (1-0,3485^2). If three images are selected the probability even raises to 95,76% (1-0,3485^2).

Name	Parameters	Stay Point			Screenshot		
		P	*FP*	*R (in %)*	*F*	*NF*	*R (in %)*
StayPoint Detection (Li)	δ= 30 mm ; τ= 5000 ms	81	64	55.86	41	8	83.67
	δ= 30 mm ; τ= 4000 ms	100	77	56.50	44	5	89.80
	δ= 30 mm ; τ= 3000 ms	115	84	57.79	44	5	89.80
	δ= 20 mm ; τ= 5000 ms	79	57	58.09	40	9	81.63
	δ= 20 mm ; τ= 4000 ms	99	81	55	44	5	89.80
	δ= 20 mm ; τ= 3000 ms	124	106	53.91	45	4	91.84
ParkingCandidate Detection (Yuan)	δ= 30 mm ; τ= 5000 ms	15	9	62.50	20	29	40.82
	δ= 30 mm ; τ= 4000 ms	16	8	66.67	21	28	42.86
	δ= 30 mm ; τ= 3000 ms	14	10	58.33	18	31	36.73
	δ= 20 mm ; τ= 5000 ms	14	8	63.64	21	28	42.86
	δ= 20 mm ; τ= 4000 ms	15	8	65.22	21	28	42.86
	δ= 20 mm ; τ= 3000 ms	14	10	58.33	18	31	36.73
Outside Ball 1D	a_1= 2.0; a_2=1e16	406	342	54.27	45	4	91.84
	a_1= 3.0; a_2=1e16	396	342	53.66	45	4	91.84
	a_1= 4.0; a_2=1e16	363	225	61.73	44	5	89.80
	a_1= 5.0; a_2=1e16	357	225	61.43	44	5	89.80
	a_1= 2.0; a_2=100	405	342	54.22	45	4	91.84
	a_1= 3.0; a_2=100	395	342	53.96	45	4	91.84
	a_1= 4.0; a_2=100	362	225	61.67	44	5	89.80
	a_1= 3.0; a_2=1000	396	342	53.66	45	4	91.83
Outside Ball 3D	a_1= 2.0; a_2=1e16	325	195	62.50	46	3	93.88
	a_1= 3.0; a_2=1e16	271	156	63.47	46	3	93.88
	a_1= 4.0; a_2=1e16	230	130	63.89	44	5	89.80
	a_1= 5.0; a_2=1e16	200	107	65.15	44	5	89.80
	a_1= 2.0; a_2=100	317	194	62.04	46	3	93.88
	a_1= 3.0; a_2=100	263	155	62.92	46	3	93.88
	a_1= 4.0; a_2=100	222	129	63.25	44	5	89.80
	a_1= 3.0; a_2=1000	271	156	63.47	46	3	93.88
Trajectory subsection analyzer	-	64	36	64.00	40	9	81.63
Last point visited strategy	-	23	2	92.00	31	18	63.27
Most visited points strategy	-	95	25	79.17	35	14	71.43

Table 1. Stay Point Detection Results.

CONCLUSION AND FUTURE WORK

In this paper we implemented and evaluated stay point detection algorithms to infer suspicious structures from trajectory data a clinician produces while using a DICOM viewer. The main conclusions of this evaluation can be summarized as follows:

- Clinicians appreciate interfaces which suggest them a good starting point in a case. The presented stay point algorithms, particularly the outside ball algorithm or the parameter less trajectory subsection analyzer are a first starting point for the development of such interfaces. The development and evaluation of such interfaces in real world scenarios are subject to our ongoing research.
- Stay point detection algorithms which use a cluster approach are barely usable to infer suspicious structures. Due to the density of the trajectory points these algorithms retrieve often one stay point only for a whole case. An improvement could be the introduction of weights for the trajectory points as it will decrease the density. This is subject to ongoing research.
- The users tend to revisit suspicious structures in a case before they close the DICOM viewer. Hence the last position of a user A can be used as a starting point for a user B.
- In our work we tested the effectiveness of the algorithms separately. The results may improve if the results of different algorithms are combined.

ACKNOWLEDGEMENTS

This work has been funded by the European Commission under Grant Agreement no. 600641, FP7 Project Go-Smart and the German Federal Ministry of Education and Research (BMBF) (project HUMIT, http://humit.de/, grant no. 01IS14007A). We would like to thank all users from university hospital Leipzig and university hospital Frankfurt who participated in the experiment. Their help made this analysis possible.

REFERENCES

1. Filipe R Barra, Renato R. Barra, and Alaor B. Sobrinho. 2010. Freeware medical image viewers: can we rely only on them?. Radiol Bras. 43:313-318.
2. Roland Ellerweg, Dominic Reuter, and Phil Weir. 2016. Architecture of a web-based DICOM viewer showing segmentations and simulations. In *Proceedings of the 18th International Conference on e-Health Networking, Applications and Services (Healthcom'16)*, 1-5.
3. Roland Ellerweg, Dominic. Reuter, Elmar Stärk, Phil Weir. (2016). Design & Implementation of an on demand loading web based Dicom viewer showing anatomical structures. International Journal of Computer Assisted Radiology and Surgery. 11(1) pp. 178-179.
4. Riccardo Guidotti, Anna Monreale, Salvatore Rinzivillo, Dino Pedreschi, and Fosca Giannotti. 2015. Retrieving points of interest from human systematic movements. *Lecture Notes in Computer Science,* 294-308. DOI=http://link.springer.com/chapter/10.1007/978-3-319-15201-1_19
5. Daniel Haak, Charles E. Page, Klaus Kabino, and Thomas M. Deserno. 2015. Evaluation of DICOM viewer software for workflow integration in clinical trials. In *SPIE 9418 Medical Imaging*, pp. 94180O-94180O-9. DOI=http://dx.doi.org/10.1117/12.2082051.
6. Yuan N. Jing, Zheng Yu, Zhang Liuhang, Xie Xing, and Sun Guangzhong. 2011. Where to find my next passenger? In *Proceedings of the 13th International Conference on Ubiquitous Computing.* UbiComp'11, 109-118. DOI=http://doi.acm.org/10.1145/2030112.2030128
7. Henrik Kretzschmar, Markus Kuderer, and Wolfram Burgard. 2014. Learning to Predict Trajectories of Cooperatively Navigating Agents. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'14)*, 4015-4020.
8. Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. 2008. Mining user similarity based on location history. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems (GIS'08*). 34:1-34:10. DOI=http://doi.acm.org/10.1145/1463434.1463477
9. Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. 2009. WhereNext: A Location Predictor on Trajectory Pattern Mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'15)*, 637-646. DOI=http://doi.acm.org/10.1145/1557019.1557091
10. Susanta Satpathy, Lokesh Sharma, Ajaya K. Akasapu and Netreshwari Sharma. 2011. Towards Mining Approaches for Trajectory Data. International Journal of Advances in Science and Technology 2, 3: 38-43.
11. Md Reaz Uddin, Chinya Ravishankar, Vassilis J. Tsotras. 2011. Finding Regions of Interest from Trajectory Data. In *Proceedings of the IEEE International Conference on Mobile Data Management*, 39-48.
12. Xing Xie, Yu Zheng, Liuhang Zhang, and Nicholas J. Yuan. 2013. T-Finder: A Recommender System for Finding Passengers and Vacant Taxis. *J. IEEE Transactions on Knowledge & Data Engineering*. 25, 2390-2403.

DOI=http://doi.ieeecomputersociety.org/10.1109/TKDE.2012.153

13. Yang Ye, Yu Zheng, Yukun Chen, Jianhua Feng, Xing Xie. 2009. Mining Individual Life Pattern Based on Location History. In *Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware.* MDM '09, 1-10. DOI=http://dx.doi.org/10.1109/MDM.2009.11
14. Yu Zheng. 2015. Trajectory Data Mining: An Overview. ACM Transactions on Intelligent Systems and Technology (TIST) 6,13: 1-41. DOI=http://doi.acm.org/10.1145/2743025