This is a preliminary version of an article published by
P. Mildner, F. Claus, S. Kopf, W. Effelsberg:
Navigating Videos by Location.
Proc. of ACM Workshop on Mobile Video (MoVid), February 2013.

Navigating Videos by Location

Philip Mildner
University of Mannheim
Mannheim, Germany
mildner@informatik.uni-mannheim.de

Frederik Claus
University of Mannheim
Mannheim, Germany
fclaus@mail.uni-mannheim.de

Stephan Kopf
University of Mannheim
Mannheim, Germany
kopf@informatik.uni-mannheim.de

Wolfgang Effelsberg
University of Mannheim
Mannheim, Germany
effelsberg@informatik.uni-mannheim.de

ABSTRACT

The combination of images with geographical information has gained a lot of attention; systems like Google Street View (GSV) have become an integral part of our daily routine and uploading and sharing geotagged images becomes more and more popular. With videos, however, this trend has not started yet. In this paper, we propose a model that combines videos with location data in interactive video tours. An introduced virtual map layer abstracts from single traces to compose one continuous video tour from an arbitrary set of input videos. This allows for an automatic selection of video files with an optimal coverage rate while minimizing transitions between videos. We implemented our model in a web application that enables users to create their own video tours and to freely change both time and location of the video stream.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based services; I.4.9 [**Image Processing and Computer Vision**]: Applications

General Terms

Algorithms, Design

Keywords

Video Navigation, Geotagging, Interactive Video Tour

1. INTRODUCTION

GSV[1] is popular, very popular. A lot of people are using it on a daily basis to survey a particular destination or area before they actually arrive. They have specific questions such as: 'How steep is the hill?' or 'Is the area wheelchair accessible?'. And there are many more things people are interested in. In many of those cases, a picture from GSV is sufficient. In many others, it is not and other novel projects have evolved. Wheelmap[2] is one of them. Its goal is to categorize the wheelchair friendliness of places. In the case of the wheelchair friendliness it is easy to create a map that marks spots as friendly, partially friendly, or not friendly. But there are more peer groups, who have special needs or are interested in other features of places.

What happens when the information represented cannot be qualified precisely? In the case of 'How steep is the hill?', a quantification can be done in percent or degree. Although this might seem like a good solution, steepness might be judged differently by different people. Another way which might make the retrieval of such information significantly easier is to provide video content about a specific location. Users who watch the video can make their own decision about the steepness.

Sharing a video and location information is still very complicated; a video is typically published on video hosting platforms such as YouTube or uploaded to a personal blog. The exact location(s) of the video must be added in either the description of the YouTube video or the blog text. Online Maps like Google Maps (GM) can be used to display the traces of the video, but can only be embedded in the blog article and not in the description of the YouTube video. This is a nonsatisfying solution, because users cannot see the current position on the map and navigate the video along with it. Another service is needed to provide users a platform to upload videos that are displayed not in a list but on a map with videos from other users. It might be comparable to a GSV with videos. However, just displaying those videos on a map is not enough. Imagine using GSV, an employee A lives on 1st Avenue and works on 3rd Avenue and employee B lives on 2nd Avenue and works on 5th Avenue. Both employees upload videos of their path to work. It would be nice for people living on 1st Avenue and working on 4th Avenue to see the entire path from 1st Avenue to 4th Avenue using the videos from employee A and employee B. This makes it easy to cover all subsets of routes from an already existing video, without having to record every element of the subset (1st $\rightarrow$ 2nd, 2nd $\rightarrow$ 3rd, etc) of the video.

The contribution of this work is to present a way to connect traces to other traces, make them routable, and find an

[1] `http://maps.google.com/streetview`

[2] `http://www.wheelmap.org`

MoVid '13, February 26-March 1, 2013, Oslo, Norway.

optimal path in terms of high video coverage and little trace changes between two points. A web application is used for interaction with the system. It supports the playback of a path, offers a video upload option for users, and makes route navigation by time and geographical position possible. The format of location information is specified loosely; therefore, it is possible to input any location data in any projection, and if necessary, extensions are supplied.

This paper is structured as follows. Section 2 presents related work. In Section 3 an overview of our model and the scenario is given. The following four Sections present our solutions for modeling, overlaying, connecting, and routing of traces. Section 8 then gives an overview over the web application we built. An evaluation follows in Section 9 and Section 10 concludes the paper.

2. RELATED WORK

With the increasing amount of multimedia content that is available on the Web, multimedia information retrieval has been an active research area in the past years [6]. A use case for this field is the provision of navigation within videos by using computer vision techniques [4, 5] or content analysis approaches [8]. Instead of only relying on the information and meta data of the video itself, additional information can be included in the retrieval process to improve existing solutions. As smartphones are becoming more and more popular, one natural scenario is the combination of mobile video with the position and/or compass data of the devices. Among existing work are the automatic detection of transportation modes through raw Global Position System (GPS) data in [14] or a visual editor for GPS raw data in [11]. In [10], video analysis and GPS data are combined to provide a mobile localization and tracking functionality. By also including compass data it is possible to track view angles of recorded videos in order to identify view scenes within a video [2].

Several applications have been proposed that make use of the comprehensive analysis and available meta data. Lu et al. considered positions of geo-tagged photos at touristic locations to automatically create route recommendations for tourists [7]. Google co-founder Larry Page started GSV by driving around San Francisco Bay Area and recording parts of the city with a video camera [1], because sharing a video virtual tour seems much more intuitive than taking hundreds of pictures to create an image virtual tour. A related topic is covered in [3] where geographical trajectories are segmented so that they can be used to create a certain route. An application for annotation and navigation within single videos is presented in [12]. In [13], Zhang et al. propose an application for the automatic creation of video summaries based on geographical information. E. g., a video summary can be created out of several input videos that all include a certain landmark. In this way, tourists can get a better impression of a scene than only looking at pictures.

3. SCENARIO

This paper presents a novel approach to the field of video composition and navigation. Instead of presenting videos that only can be navigated by adapting the time axis, our application also incorporates position information. In this way, users are able to navigate through multiple videos by specifying the geographical position they want to see. This is in contrast to existing work where either single videos are considered or no navigation by location is possible. Consider the following sample scenario: Multiple users want to share a video of their favorite path through Central Park. They start recording a GPS eXchange Format (GPX) trace with *Open GPS Tracker*[3] and start recording a video. They then walk their favorite route until the end, where they stop recording both, the video and the GPX trace. The application created in this work displays all traces of all users in one map and it is possible to select two points from all of these traces. These two points mark the start and end of the virtual video tour. The application then builds an optimal path from the graph of all traces and displays the videos associated with the traces to the user. The resulting path does not have to fully overlap with one of the source videos; instead, a video stream following the specified route is created dynamically.

Ten routes in the city of Mannheim, Germany have been used. The majority of the routes were recorded on foot, while three routes were recorded with the streetcar as well. In total, we recorded 25 GPX traces and 25 videos the way described above. Figure 1 shows three routes mapped to an Open Street Map (OSM) projection[4]. This visualization already connects adjacent points of the trace. If a user selects one of the first points in front of the *castle* and one in front of the *water tower*, the application may take half the trace of route 3 (from the *castle* to the *market place*) and the whole trace of route 2 (from the *central place* to the *water tower*). The user is presented with the video from route 3 to the middle, which then switches to video from route 2.

Several of the potential problems are obvious: First, traces, which should overlay, do not overlay. For example, traces from *castle* to *central place* and from *castle* to *market place* should overlay. Second, traces are not connected. The trace from *castle* to *central place* should be connected to the one to *water tower*. Last, not all points of the traces are visible. A user should only be able to click points of the traces, as it is not clear where exactly the user has been between two points.

4. MODELING TRACES

The first step is the representation of traces and points of traces. One needs to find a model to represent the data in a way that is understandable and routable while keeping the integrity of the data, even when adding several slightly different traces. Figure 1 shows the traces as a path. However, the path is still lacking points for the user to select. Therefore, we introduce a model that abstracts from the actual traces to generic points on the map.

Modeling the traces describes the process of creating a database model that holds all necessary information. Figure 1 shows three different traces marked with different colors. Several points of the traces overlap with other traces, and adjacent points are not connected to each other. In our model, we generally use directed graphs for representation of traces and routes.

4.1 Virtual and real layer

It is possible to separate between points of a trace and

[3] `http://code.google.com/p/open-gpstracker/`

[4] For license terms see `http://www.openstreetmap.org/copyright`.

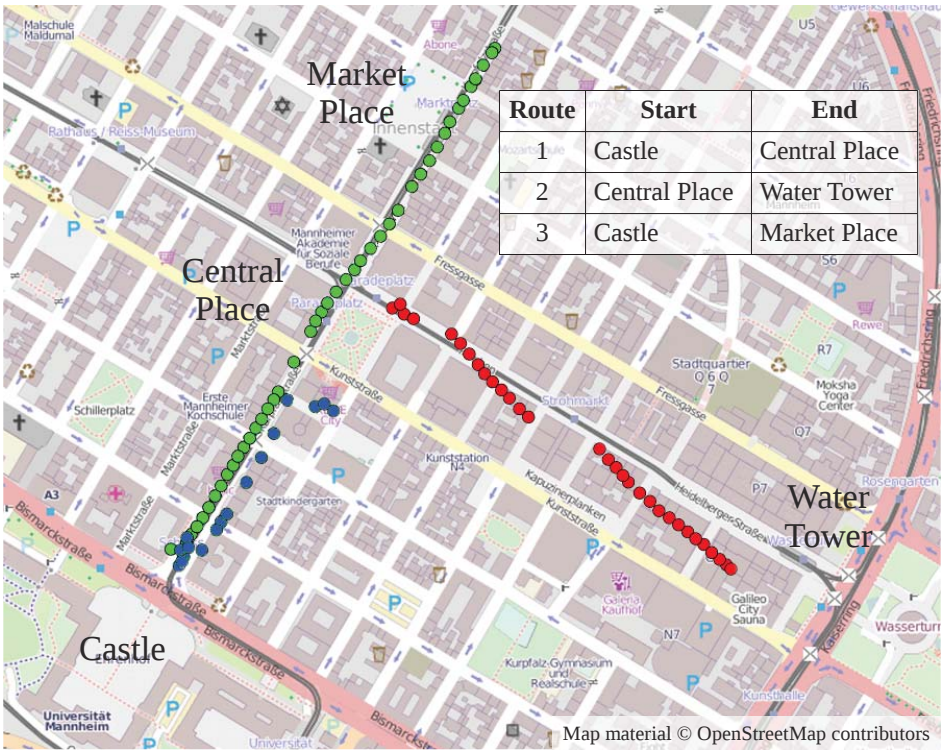

Figure 1: Three GPS traces are visualized with different colors. The trace fom central place to water tower shows errors due to GPS inaccuracies.

points of the map. Points of the map should be unique and build up an availability map for videos. Therefore, they are residing in the so called real map layer. Points of a trace, on the other hand, can be considered as virtual points. They are virtual, because a point of a trace describes a point of the map plus additional information (time and video), similar to meta data. Every point of a trace must belong to one point of the map. The connection between the points can either be made between two points of a trace or between two points of a map. Figure 2 shows the layer architecture, which depicts two simple traces. Between both traces, the connection is made via a connection of the map.

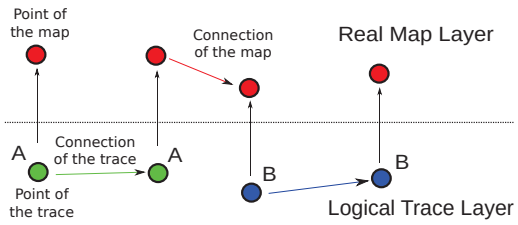

Figure 2: Separation of real layer and logical layer. A point of the logical trace may be connected to other points of the logical trace or the real map. On the other hand, a point on the real map may only be connected to other points on the map.

4.2 Definition of the trace model

A point on the map will be referred to as MapPoint and a point of the trace as TracePoint. A MapPoint stores its latitude and longitude and is capable of holding several TracePoints in a composition relationship, because multiple videos can use the same point. A TracePoint stores videotimestart and videotimend of that point. The videotimestart is the time in seconds that has already passed since the beginning of the video. The videotimeend is the time in seconds until the point is left again. This is necessary to prevent the same TracePoint from appearing multiple times, when the user stops moving and stands still on one spot while taking a video. For a sequence of almost identical coordinates in the trace, there will only be one TracePoint with the videotimestart of the first and the videotimeend of the last point of the sequence.

A MapPointConnection connects two MapPoints as a directed edge and is used to bridge the gap between the end of one trace and the start of another. A TracePointConnection on the other hand should connect two TracePoints and because TracePoints are connected to MapPoints, they establish a connection between two MapPoints. Additionally a TracePointConnection must store the video that the trace belongs to. A Video is stored as a simple data transfer object.

5. OVERLAYING TRACES

After the specification of our model, it is now necessary to overlay traces that represent the same route. The traces must be overlaid to represent the same sequence of connections on the map layer. Figure 1 shows the trace layer of three different routes. In our example, the route from *castle* to *central place* must be overlaid with the one from *castle* to *market place*. To clarify: Every TracePoint of the trace

from the *castle* has to share the same MapPoint with other TracePoints from different traces in the same area.

While trying to overlay the traces, our system has to deal with the imprecision of the satellite navigation and the fact that two people walking on the same pavement will never truly walk the exact same path (for example, walking further on the left, or further on the right side of the pavement). It is also important to determine whether or not two different traces really depict the same or an opposite path. Another problem arises when a user stands still while recording. The resulting trace will not have the same point, but slightly different points, over and over again. The application needs to have capabilities to deal with these sources of error.

A naive approach would be to just append new traces to existing ones if they are in a certain distance of the first trace. This, however, is not a feasible solution as there is no guarantee that the first trace fits correctly to an available map.

A good solution is to only derive a MapPoint from a Point of Interest (POI) that is already present in an available resource. OSM[5] defines data by three basic data types: Node, Way, and Closed Way. Nodes are used to mark POIs like shops or restaurants, but they also define Ways. The OSM database holds a large amount of Nodes collected by the community. These points can be used as MapPoints and nearby TracePoints can be appended to them. The process of resolving location information into POIs is called reverse geocoding. Plain reverse geocoding will lead to a number of problems when one trace skips points or ends at a point that was skipped by all other traces. This will produce dead-ends that are hard to spot and make routing impossible. Therefore, it is necessary to build a directed graph with the right OSM dataset, for example highway=electrified for the train transportation mode. Every trace starts on one Node of the graph and must follow the graph successively in the direction of travel or stay at the current Node. It is necessary to implement a heuristic that can detect when the direction of travel in the OSM graph and the trace are opposing and in that case force the trace on the correct side of the street. Coping with traces that have a high inaccuracy is non trivial and will require a sophisticated look-ahead and look-behind heuristic to always choose the best node. This solution is reliable in cases where enough data is available in the database and will deliver perfect solutions when the quality of the trace is good enough. It is also possible to create a routable copy of the OSM data[6] in a local database, instead of building the graph in memory. As this solution offers a good compromise between implementation effort, performance, and usability of the created graph we chose this approach for our model.

6. CONNECTING TRACES

After modeling traces and overlaying traces, the next step is to connect traces to other traces in close proximity. Going back to Figure 1, the trace from the *castle* to the *central place* needs to be connected with the one to the *water tower*. As there is no video stream available for these MapPointConnections, the cost of the transition, i. e., the distance between them, should be minimized.

The problem is very similar to the previous one: The concept of proximity or 'how to find videos in close proximity' and how to connect them. But there is an additional constraint that needs to be considered: A trace end should not be connected to another trace end. This can be phrased more abstract as, an edge should not be added when the target MapPoint has an outgoing degree of 0. Solving this problem requires a strategy to ensure that every MapPoint is connected to every other MapPoints in reach, while trying to minimize the number of MapPointConnections needed. If the number of MapPointConnection is greater than needed, routing might skip a certain distance and will produce an incomplete virtual tour and user experience. A simple solution would be to just connect each MapPoint with an outgoing degree of 0 to the nearest MapPoint with an incoming degree of 0 so that start and end of traces are connected. This solution, however, is far from optimal as it may introduce big gaps between videos and it would not be possible to do a transition in the middle of a trace (see Figure 3 left).

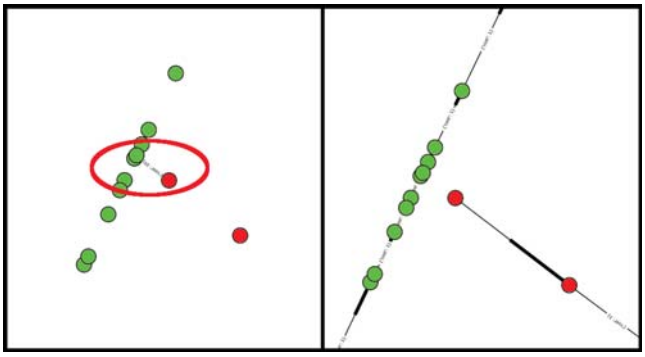

Figure 3: Two traces component problem and solution by adding a virtual connection to the map layer

There needs to be a way to tell that a certain Point cannot reach another Point. To solve this problem, our solution takes graph components into account [9]. For each component the minimal distance to all other components is computed. This is done by comparing each MapPoint to all MapPoints of another component. Between the pair of points with the minimal distance a MapPointConnection is inserted resulting in a *virtual connection* of these two components. Figure 3 shows two sample traces on the left and the optimal connection on the right which connects them.

7. ROUTING TRACES

After modeling, overlaying, and connecting traces, the graph is usable for navigation. The next step is to find an optimal path from the start to an end point while trying to maximize video coverage and minimize trace changes.

Transition to a different trace is only possible via a MapPointConnection or a TracePointConnection. While the video coverage decreases when a MapPointConnection is used for changing traces, it does not decrease while using a TracePointConnection. Consequently, TracePointConnections should be cheaper to use, but switching to another trace cannot be completely free, as this violates the goal of trying to minimize trace changes.

To find the optimal path, we use a modified Dijkstra single source shortest path algorithm. Instead of iterating over

[5] http://www.openstreetmap.org/
[6] http://pgrouting.org/index.html

the points, we need to iterate over the connections. Every MapPoint has to add an edge pointing to itself to symbolize a change of traces. This new edge must be cheaper than a MapPointConnection and more expensive than a TracePointConnection. The cost of a TracePointConnection is defined as 1. In cases where there is no adjacent TracePointConnection available a MapPointConnection is used instead. Here the cost is determined by the distance of the MapPointConnection to reflect the increased costs compared to transitions between traces.

In contrast to the original Dijkstra where a vertex is more important than an edge, vertices are of almost no meaning in our model. They are representations of start and end points of the route, which solely consists of TracePointConnections and/or MapPointConnections. In addition, the predecessor of every Connection needs to be stored, because there could be more than one Connection preceding it. Instead of only taking costs of transitions into account for the route calculation, the predecessor is used to differentiate transitions within a trace from transitions between neighboring traces.

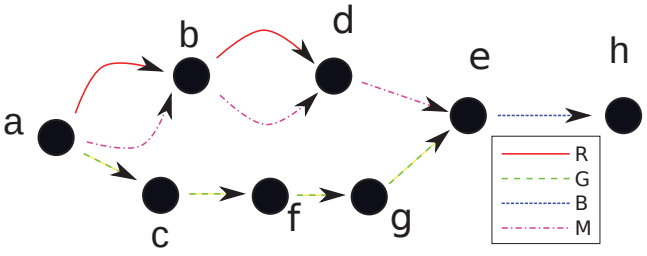

Figure 4: Sample graph with four traces

To demonstrate the algorithm, take a look at Figure 4. It shows a sample graph $G = (V, E)$ with $E = \{a, ..., h\}$ and $V = \{ab_R, ab_M, bd_R, bd_M, de_M, ac_G, cf_G, fg_G, ge_G, eh_B\}$, where nm_G is a TracePointConnection between n and m belonging to trace G. We now want to find the optimal path from a to h with the modified Dijkstra algorithm. First, the cost of all TracePointConnections are set to ∞, except for ab_R, ab_M, ac_G which depict the first possible trace sections originating from the starting position:

Vertex	Cost	Predecessor
$S : \{ab_R, ab_M, ac_G\}$	1	*None*
$\{v : v \in V, v \notin S\}$	∞	*None*

The algorithm iterates over all Connections starting with the Connection with the lowest cost. After the first step, the cost of the adjacent edges are lowered to 2. The predecessors of bd_R and bd_M differ, as the algorithm favors routes on the same trace over trace changes:

Vertex	Cost	Predecessor
bd_R	2	ab_R
bd_M	2	ab_M
cf_G	2	ac_G

Now bd_R, bd_M and cf_G all have cost of 2. The next table shows all their adjacent Connections. de_M could be reached both from bd_R and bd_M. Unlike in the previous step, the transition from bd_R to de_M necessarily requires a trace change, because there is no alternative trace section. bd_M is therefore favored as no trace change is required here:

Vertex	Cost	Predecessor
de_M	3	bd_M
fg_G	3	cf_G
ge_G	4	fg_G

There are now only two Connections left that determine the cost and predecessor of eh_B, de_M and ge_G. ge_M sets the cost of eh_B to $4 + 1 = 5 < \infty$. de_M sets the cost to $3 + 1 = 4 < 5$. The optimal path can be discovered by going in reverse from eh_B and reading the value of the predecessor. This will result in $R : (eh_B, de_M, db_M, ab_M)$ with a total of one trace change and a distance of $|R| = 4$ (see Table 1).

Vertex	Cost	Predecessor
ab_R	1	*None*
ab_M	1	*None*
bd_R	2	ab_R
bd_M	2	ab_M
de_M	3	bd_M
ac_G	1	*None*
cf_G	2	ac_G
fg_G	3	cf_G
ge_G	4	fg_G
eh_B	4	de_M

Table 1: Result of the algorithm based on the graph shown in Figure 4

At the moment, the optimal path can only be calculated between two points. This is not as restrictive as it sounds, because every path $(a_1, \ldots, a_i)$ with $i > 2$ can be split up into $(a1, a2), (a2, a3), \ldots, (a_{i-1}, a_i)$. The shortest path algorithm must be repeated $i - 1$ times to compute a route that connects all i points.

8. IMPLEMENTATION

We implemented our model as an HTML5 web application using Django[7] as web application framework. Reverse geocoding of coordinates must be done via a stand-alone OSM eXtensible Markup Language (XML) file or with OSM data in the database. A setup for a productive environment will require a routable copy of the OSM in a database to ensure fast response times. Osm2pgrouting[8] is a tool which populates a database with the given OSM data using the types and function defined by OpenGIS[9].

For the frontend, OSM and its JavaScript framework OpenLayers[10] have been chosen instead of GM. In contrast to GM, OpenLayers is a powerful framework that can display several vector layers with selectable features and custom layout on top of a base layer. Figure 5 shows a screenshot of the frontend. In the application, all available connections are displayed in red. Currently, the user selected half of the trace from *castle* to *central place* and half of the trace from *central place* to *water tower*. The optimal path is displayed in green and the current playing segment is displayed in blue. Navigation within the video stream can be achieved

[7] http://www.djangoproject.com
[8] http://pgrouting.org/docs/tools/osm2pgrouting.html
[9] http://www.opengis.org
[10] http://wiki.openstreetmap.org/wiki/OpenLayers

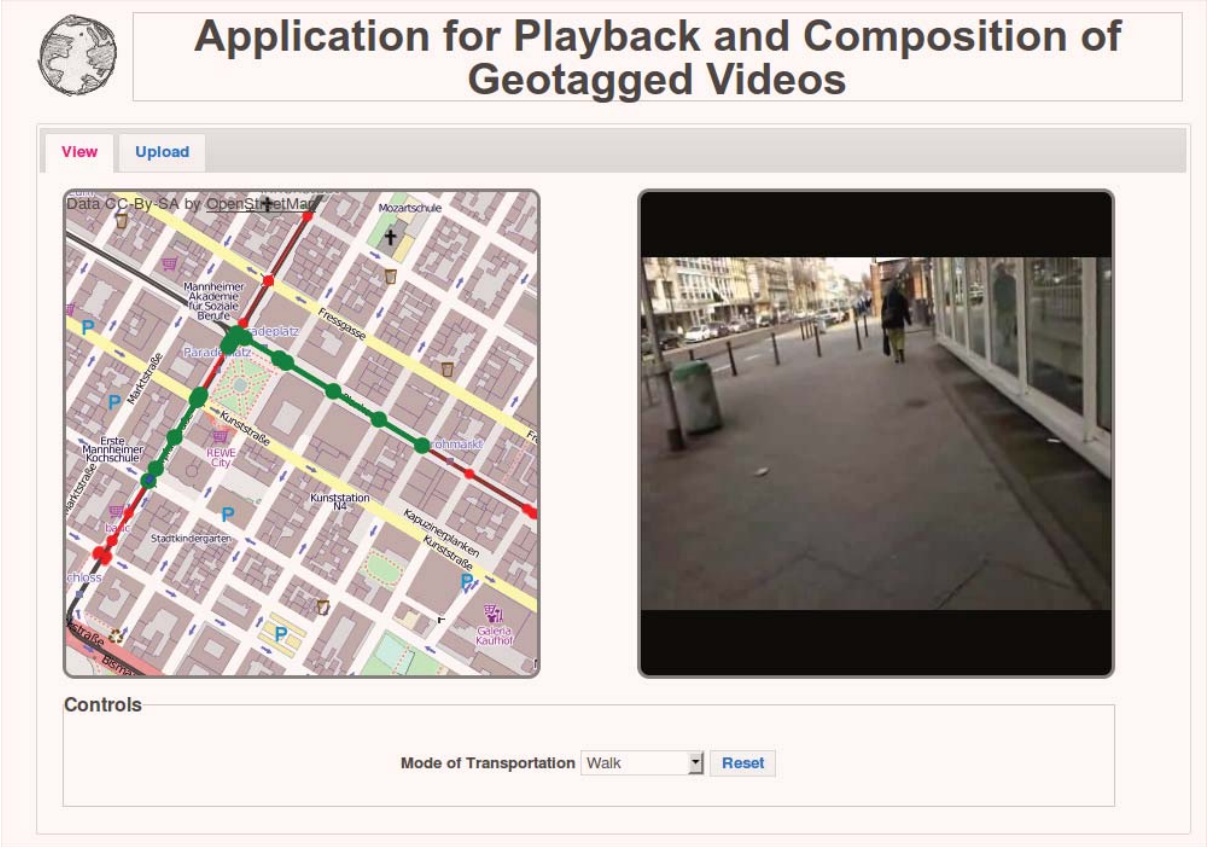

Figure 5: Screenshot of the web frontend showing the map view and a video

by either adapting the time slider or by selecting a specific segment of the calculated route. A prototype of our application is publicly available[11].

Our application supports customization of the selection process. Users are able to specify the modes of transportation, i. e., foot, train, bicycle, and motor vehicle when looking for a video tour. This rudimentary scheme can be extended by including further categories, like time of day, view angle, or video quality. Based on the specified criteria the best suited route is calculated out of all available routes. In case no route can be found given one combination of parameters, a notification is displayed to that the user can adapt the request.

9. DISCUSSION

To get a first insight into our system, we evaluated it with a set of 25 recorded traces. As our model abstracts from real traces it has to make sure that videos showing the same route are mapped to the same `Point`s. On the other hand, videos showing different routes should result in separate routes on the map. To measure this, we calculate the deviation of the video routes to both the GPS traces and the map projection. The original paths were extracted from the videos by manually aligning the routes with the OSM map. Results show that the average error caused by GPS inaccuracies is 15.8 m. In 23 of 25 traces the original path could be restored by our mapping. However, there are two situations where the projection does not deliver correct results. First, the algorithm is not able to compensate GPS errors bigger than 40 m resulting in a wrong path. Second, if there is no path present in the OSM map, the algorithm cannot reconstruct the actual route. This especially occurs on squares like the market place where there is no clearly defined path.

Apart from the projection of all individual traces, our system also has to correctly connect traces in order to create video tours with as few transitions as possible. While short distances without any videos can be skipped, too big gaps should be avoided so that users can retain the impression of one contiguous tour. We therefore estimate the maximum distance for these `MapPointConnection`s. We set differing values for different transportation modes to incorporate the movement speed. Consequently, the maximum distance for videos by foot was set to 50 m, while the value for streetcar videos was set to 500 m. These values provide a good compromise between connectedness of the routes and quality degradation due to gaps in the video stream.

Overall, the system proved to be easy-to-use in the data acquisition phase. As only a video and a GPX file are needed, traces can be recorded with any smartphone or camera without the need of installing additional software. The web application can already be used to create individual video tours, however, it is not clear how users will accept the generated video stream as it may include sudden changes in viewing angle and video continuity. E. g., one user might

[11] `http://ls.wim.uni-mannheim.de/de/pi4/research/projects/geotagged-videos`

prefer longer video segments in a poor video resolution to maximize video consistency while another user might maximize resolution at the cost of more frequent transitions. A solution here would be to include more customization options for the tour creation process so that users can set their preferences.

10. CONCLUSION

In this paper, we presented an approach for the creation of video tours that can be navigated by time as well as location. Similar to geotagged images, our model incorporates GPS traces that are recorded along with videos. These traces are projected to a map by abstracting from the actual traces to generic routes on the map. In this way, similar videos can be aggregated and routes spanning over several input videos can be created. We implemented our model in a publicly available web application. Results of a preliminary evaluation show that in most cases our algorithm correctly projects the traces to the map. We plan to perform a larger evaluation where we want to measure the perceived quality of the generated videos. Furthermore, we want to optimize the projection algorithm and include more metrics for customizing the created video streams.

11. REFERENCES

[1] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. a. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: Capturing the world at street level. *Computer*, 43(6):32 –38, 6 2010.

[2] S. A. Ay, R. Zimmermann, and S. H. Kim. Viewable scene modeling for geospatial video search. In *Proceeding of the 16th ACM international conference on Multimedia – MM '08*, pages 309–318. ACM Press, 2008.

[3] M. Buchin, A. Driemel, M. van Kreveld, and V. Sacristán. An algorithmic framework for segmenting trajectories based on spatio-temporal criteria. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '10*, pages 202–211. ACM Press, 2010.

[4] D. B. Goldman, C. Gonterman, B. Curless, D. Salesin, and S. M. Seitz. Video object annotation, navigation, and composition. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 3–12. ACM, 2008.

[5] S. Kopf, S. Wilk, and W. Effelsberg. Bringing videos to social media. In *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, pages 681 –686, July 2012.

[6] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1):1–19, Feb. 2006.

[7] X. Lu, C. Wang, J.-m. Yang, Y. Pang, and L. Zhang. Photo2Trip: Generating Travel Routes from Geo-Tagged Photos for Trip Planning. In *Proceedings of the international conference on Multimedia - MM '10*, pages 143–152. ACM Press, 2010.

[8] K. Schoeffmann, M. Taschwer, and L. Boeszoermenyi. The video explorer: a tool for navigation and searching within a single video based on fast content analysis. In *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, MMSys '10, pages 247–258. ACM, 2010.

[9] C. Vasudev. *Graph Theory with Applications*. New Age International, 2006.

[10] Q. Wang, W. Zeng, and A. G. Lobzhanidze. Mobile Media in Action: Remote Target Localization and Tracking. *IEEE Multimedia*, 19(3):74–80, July 2012.

[11] J. Weitkämper and T. Brinkhoff. XFormsGI – Extending XForms for Geospatial and Sensor Data. In J. Ware and G. Taylor, editors, *Web and Wireless Geographical Information Systems*, volume 4857 of *Lecture Notes in Computer Science*, pages 76–93. Springer Berlin / Heidelberg, 2007.

[12] B. Zhang, Q. Li, H. Chao, B. Chen, E. Ofek, and Y.-Q. Xu. Annotating and navigating tourist videos. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '10*, pages 260–269. ACM Press, 2010.

[13] Y. Zhang, G. Wang, B. Seo, and R. Zimmermann. Multi-video summary and skim generation of sensor-rich videos in geo-space. In *Proceedings of the 3rd Multimedia Systems Conference on - MMSys '12*, pages 53–64. ACM Press, 2012.

[14] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma. Understanding transportation modes based on GPS data for web applications. *ACM Trans. Web*, 4(1):1:1–1:36, Jan. 2010.