Dokumentation der Applikation, sowie Beschreibung der Softwarearchitektur und Algorithmen.

Erstellung eines Webdokuments zur Vorstellung der Projektergebnisse:

**Motivation**:

* Informelle Beschreibung des Problems

People like to watch videos but only if a video shows interesting things or locations. With the MediaQ project we soon will have the opportunity to watch videos from all over the world. The problem is that you don’t know what a video shows before you have watched the video. But it would be much better if you know before whether the video shows the place you are looking for or if you can just search for videos that show the place at least for a period of time. Therefore we had the idea to connect the videos with the underlying geographically information. This way we have the opportunity to search for certain points of interest (POI) like sights, parks or restaurants.

* Reale Anwendungsszenarien (Beispiele)
* Someone is telling about an amazing place you have never been before. No Problem. Just search for all videos that show exactly this place.
* …

**Problemdefinition - Formale Definition des Problems:**

On the one hand we have the video database of MediaQ. There we have all the information about the videos like trajectories, trajectory points or the perspective. On the other hand we have services like the Google Places API that returns information about places (establishments, geographic locations, or prominent points of interest). The problem is now to connect the information of both systems to a single application.

**Algorithmische Lösung:**

* Formale Lösung ohne Implementierungsdetails:

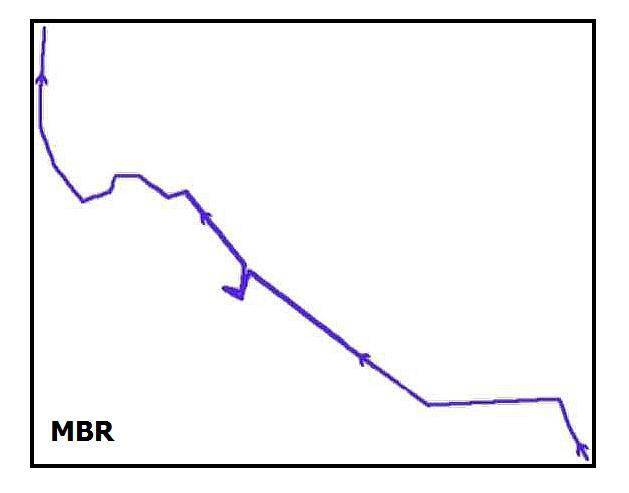
Two different search requests:

1. **Find all videos that show a given POI**

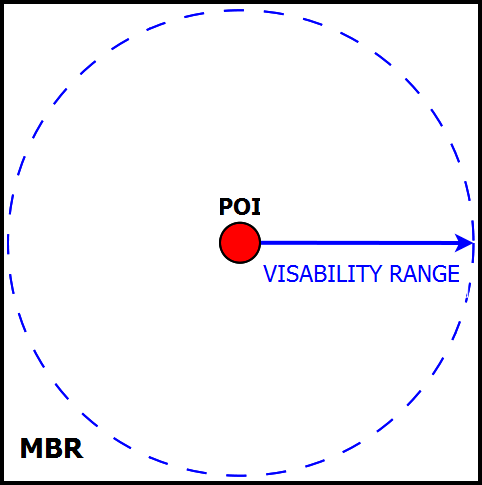
* Determine the coordinates (latitude, longitude) of the POI via Google Places API

Filtering with R-tree

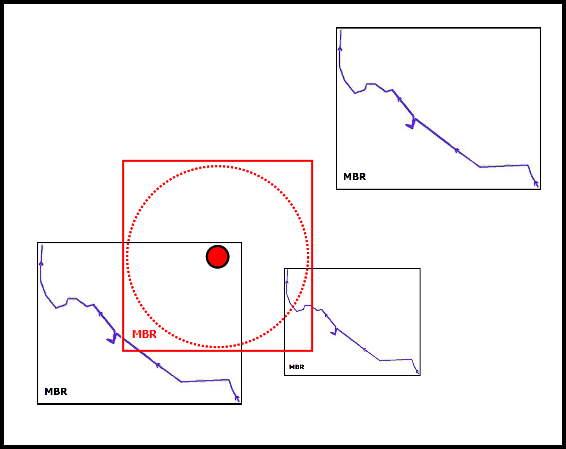
* R-tree contains all videos as minimal bounded rectangles (MBRs) 🡪 trajectory of a video can simplified be shown as a MBR



* POI with visibility range can also be shown as a MBR



* R-tree range query with the POI-MBR



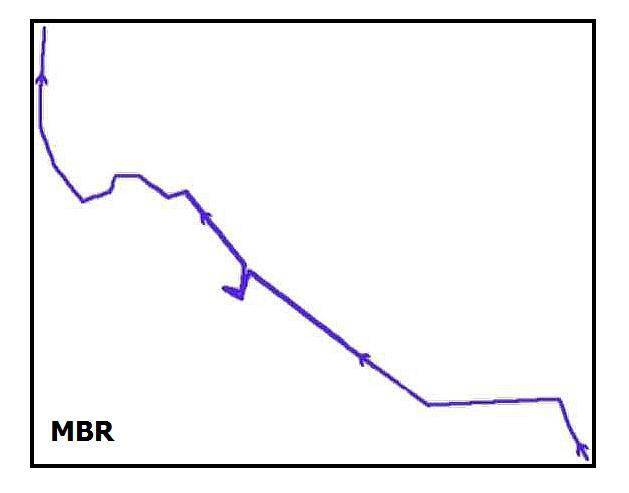
* Results in video candidates

Refinement

* For each video candidate determine if the POI is visible in a trajectory point of the video
* If it’s true then the video is in the result list for the given POI

1. **Find all POIs that are shown in a given video**

* Trajectory of a video can simplified be shown as a MBR



Filtering with Google Places API

* Get all POIs located in the video-MBR
* Results in POI candidates

Refinement

* For each trajectory point of the video determine which of the POIs candidates is visible
* Result set with all visible POIs
* Pseudocode

**Technische Umsetzung:**

* Google Places API:

Service that returns information about Places (geographic locations or prominent points of interest)

* R-tree:

R-tree for spatial queries

* UML
* Verwendete Frameworks
* …

**Evaluation gegenüber Baseline:**

**Downloads:**

* Präsentation
* Quellcode
* Programm - Einbindung als Webapplikation