Dynamic Path Exploration on Mobile Devices

Michael Birsak, Przemyslaw Musialski, Peter Wonka, *Member, IEEE,* and Michael Wimmer

Abstract—We present a novel framework for visualizing routes on mobile devices. Our framework is suitable for helping users explore their environment. First, given a starting point and a maximum route length, the system retrieves nearby points of interest (POIs). Second, we automatically compute an attractive walking path through the environment trying to pass by as many highly ranked POIs as possible. Third, we automatically compute a route visualization that shows the current user position, POI locations via pins, and detail lenses for more information about the POIs. The visualization is an animation of an orthographic map view that follows the current user position. We propose an optimization based on a binary integer program (BIP) that models multiple requirements for an effective placement of detail lenses. We show that our path computation method outperforms recently proposed methods and we evaluate the overall impact of our framework in two user studies.

Index Terms—Tourist Guide, OpenStreetMap, Exploration, Binary Integer Program

1 Introduction

Due to their low price and immense versatility, mobile devices (e.g., smartphones, tablets) have become a central element in our daily lives. We not only use them for making phone calls or messaging, but also for gaming, taking photos and for navigation.

When investigating an unfamiliar region of a city, users may additionally use the data connection of their devices and tourist applications (apps) like Yelp or TripAdvisor to find out about points of interest (POIs) in their environment. To this end, these apps usually allow limiting the search result to some preferred categories, e.g., restaurants or sights, and present the found POIs in ascending order with respect to the distance to the current position. After an interesting POI is chosen, the mobile device is used as a navigation system to find the shortest path to the chosen POI.

We see two drawbacks with this common approach that we would like to improve on: (1) It is not guaranteed that also the path itself to the POI is interesting, while there might be another path with a slightly longer distance providing places which are worth seeing; and (2) in recent work the destination selection and navigation were strictly separated and done sequentially. However, often the user does not know in advance which POIs will be the destination, but she is more interested in exploration.

In this work, we present a novel framework to overcome the mentioned limitations:

- In order to address problem (1), we compute an interesting path through a whole set of POIs instead of computing the probably uninteresting shortest path to one particular POI. To this end, we query the rating values of all POIs and seek a path to an initially undefined POI that maximizes the sum of quality values of visited POIs while at the same time staying below a maximum path length. Alternatively, the user can set a fixed destination POI (e.g., a sight) and use the system to find a path to this POI that is close to POIs of other categories (e.g., restaurants). We discuss the details in Section 4.
- Given an interesting path and a set of POIs, we address problem (2) by focusing on the environment of the path by improving the map cutouts that represent the moving viewport onto the map within a dynamic visualization. This visualization is presented to the user as the final output of our system. We find the positions of the map cutouts by computing another path that describes the smooth motion of the map cutouts during the visualization and seek a solution that emphasizes the environment of the first path, thereby showing essential information about the surrounding POIs in rectangular entities called detail lenses. We discuss the computation of the path for the map cutouts in Section 5 and the optimized detail lens placement in Section 6.

Figure 1 shows a common use case of our system. We evaluate the preference for our visualization techniques and the acceptance rate of our system in two user studies and present the results in Section 9. Finally, we show several results of our framework and also discuss its limitations in Section 10.

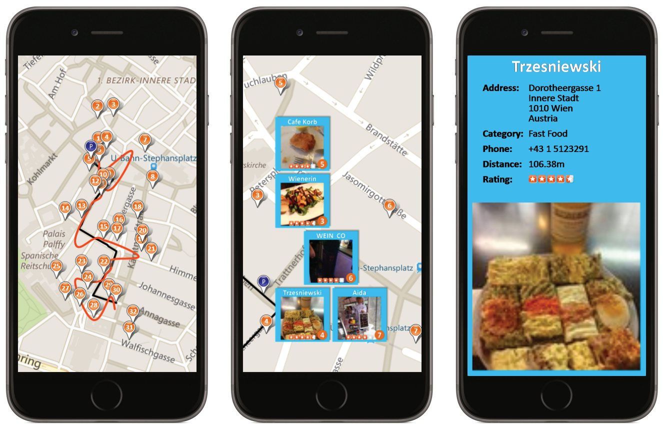
- *Michael Birsak, Przemyslaw Musialski, and Michael Wimmer are with TU Wien. E-mail: {birsak, musialski, wimmer@cg.tuwien.ac.at}*
- *Peter Wonka is with Arizona State University and King Abdullah University of Science and Technology. E-mail: peter.wonka@asu.edu*

2 Related Work

We tackle a variety of problems in our framework. In detail, our approach is related to path computation and map generation techniques. This section presents the related work partitioned into these categories.

Path Computation. While the classical shortest path problem can generally be solved efficiently [1], finding a high-quality path under a possible maximum distance constraint is substantially more challenging and is directly related to the longest path problem that is known to be NP-hard [2]. Quality is an application-dependent metric, e.g. determined by the analysis of photo locations in proximity to the paths [3], [4], [5], [6] or the retrieval and weighting of ranking values according to the distance to surrounding POIs [7]. Due to the computational complexity of finding the optimal high-quality path, researchers came up with a variety of methods to compute an approximative solution.

Previously published methods tackle the high-quality path computation problem using dynamic programming [3], adapting a shortest path algorithm [6], and genetic algorithms [8]. We will compare against these three methods in Section 9.1.



Trzesniewski

Address:	Dorotheergasse 1 Innere Stadt 1010 Wien Austria
Category:	Fast Food
Phone:	+43 1 5123291
Distance:	106.38m
Rating:	★★★★★

Fig. 1: Dynamic Path Exploration on demand. (left) Our system automatically retrieves a set of points of interest (POIs) in the environment of the user and computes an interesting path through these POIs (shown in black) as well as a separate path for the map cutouts representing the motion of the viewport during the visualization (shown in orange). (middle) The path and the POIs are then presented within a dynamic visualization based on the previously computed map cutouts, and using so-called detail lenses to show essential information about the POIs. (right) The user can click onto the detail lenses to get more information.

Map Generation. We structure the related work in map generation into two categories. *Static* methods produce printable results and *dynamic* methods target output devices with a display.

Static Methods. Although we present a dynamic method that targets mobile devices, many of the principles that motivated our design decisions were developed for a static output. For example, finding an optimal placement of detail lenses is related to the classical map labeling problem. Current methods usually try to follow some approved guidelines (e.g. avoidance of mutual overlap of labels or overlap of map features) to find a coherent layout. Two representative papers in this field are from Imhof [9] and Hirsch [10] who did pioneering work in the direction of design guidelines for map labeling and automating the labeling process.

Subsequently published methods proposed to tackle the map labeling problem using simulated annealing [11] or gradient descent methods [12]. Our method is most related to the work by Zoraster [13], [14] and Birsak et al. [15] who proposed an integer programming approach. Our proposed approach can be seen as an extension of previous work to dynamic maps, including a novel splitting technique to accelerate the computation.

There are multiple other interesting problems for static map generation. One problem is the simplification (abstraction) of cartographic data. For example, previous work studied abstractions for route maps [16], tourist maps [7], or destination maps [17].

More recently, Birsak et al. [15] proposed a system for the generation of customized tourist brochures that contain routing information for multiple destinations, which try to find a compromise between path-length and clarity of the navigational instructions in a static brochure. Additionally, they also utilize the concept of detail lenses to display additional information about POIs. In this work we continue the idea of detail lenses that are used to amplify particular points of interest on the map.

Another branch of research focused on the problem of automatic generation of multi-destination paths combined with concise and convenient visualization of such information in order to make it compact and usable. For instance, Karnick et al. [18] introduced a system to visualize routes using local detail lenses placed automatically on a map canvas. Zheng et al. [19] proposed a system for trip planning along routes with sightseeing qualities. Their system produces routes that are enriched by nearby POIs.

Dynamic Methods. While static maps are aimed for printout, dynamic maps are usually displayed on a digital screen and adapt the shown content according to the interaction of a user. For example, Been et al. [20] compute a dynamic map labeling for a pannable and zoomable map. They assign priority values to the labels and present a method that avoids mutual overlaps during interaction by omitting a placement of labels that cause overlaps with other labels of higher priority. While our method follows a very similar goal to present information about map features in a dynamic way, we avoid a mutual overlap of detail lenses by utilizing the chronological dimension to show them at the same position but in different periods of time.

Another method for interactive labeling has been proposed by Götzelmann et al. [21], which is based on heuristic optimization using so-called *layout agents*. This method also maintains coherence across frames in a dynamic setup.

Fekete and Plaisant [22] explored the capability of "excentric labels". Their method allows the investigation of features within a user-defined neighborhood without extensive use of zoom operations which makes a retrieval of information more efficient.

Our approach is particularly related to the work by Chittaro [23] since we also tackle the *presentation problem* of mobile devices. We present a new approach for this problem and propose a system that shows details in an area that is too large to be shown on the screen within a coherent dynamic visualization. To this end, we resort to a grid-based mathematical optimization similar to Birsak et al. [15] and Rylov et al. [24].

Recently, Wang et al. [25] introduced a system for automatic generation of hierarchical structures for route maps that help users to move back and forth across multiple views with different scales.

In general, in this paper we propose a fully automatic method driven by binary-integer optimization which only needs a weak specification of the actual routing goal. In fact, the user can only specify a high-level semantic description of the desired activity (e.g., dining), and perhaps the longest walking distance (e.g., 500 meters), and the system tries to find a path with the maximal walking quality. This path is then presented to the user within an appealing dynamic visualization.

3 System Overview

The input to our framework is the current position of the user $\mathbf{p}(0)$ and a set of POIs $P = \{P_1, \ldots, P_n\}$. Optionally, one of these POIs can be designated as destination. Our system produces as output:

1) An interesting path $\mathbf{p}(t)$ leading to a specific POI that is either chosen by our method or by the user (Section 4).
2) Another path $\mathbf{m}(t)$ that describes the camera motion in the visualization and that is designed to emphasize interesting regions in the environment of the path $\mathbf{p}(t)$ (Section 5).
3) The optimal placement of rectangular entities called detail lenses inside the map cutouts. The detail lenses show the most essential information about the POIs (Section 6).
4) A dynamic visualization using the previous steps, which is presented to the user on a mobile device like a smartphone or tablet (Section 7).

Figure 2 shows an overview of our system.

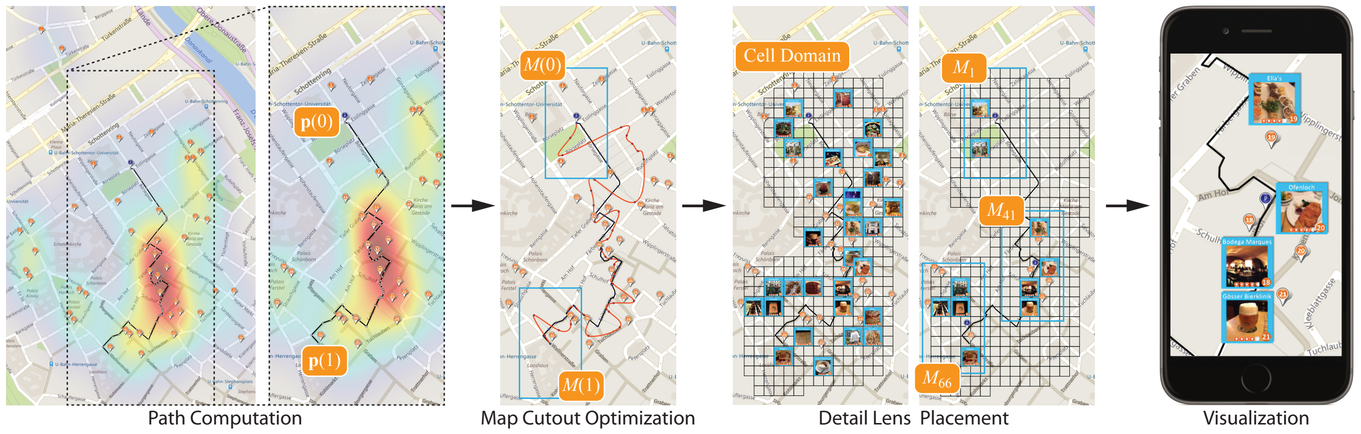


Fig. 2: Overview of our system. (left) After the POIs are queried from an online database (e.g., Yelp) we compute an interesting path $\mathbf{p}(t)$ (shown in black). The grey dots along the path refer to the positions of the nodes $\mathbf{p}_i$. (middle-left) Taking the POIs P and the path $\mathbf{p}(t)$ as inputs, we compute another path $\mathbf{m}(t)$ (shown in dark orange) for the map cutouts $M(t)$ (shown in light blue). The red dots on $\mathbf{m}(t)$ refer to the set $\mathfrak{C}$ of control points, the blue dots on $\mathbf{p}(t)$ are the corresponding set of foot points $\mathfrak{F}$. (middle-right) We partition the map into a Cartesian grid, denoted as C and aim for a dynamic placement of rectangular entities, which we call detail lenses. (right) The output of our framework is shown on a mobile device within a dynamic visualization in order to give guidance to the user.

4 Path Computation

The pedestrian path $\mathbf{p}(t)$ we seek should meet the following requirements: (1) It should be *interesting*, which we model similar to the work by Grabler et al. [7] by closeness to high-quality POIs. (2) The walking distance should stay below a threshold that is chosen by the user, e.g. 20% more than the distance of the shortest path between the current position and the dedicated destination.

In order to compute a path with these requirements, we consider the road-network graph $G = (V,R)$ with nodes $v_i \in V$ and road segments $r_{i,j} \in R$ representing the edges of the graph and seek a path from a source node s to a destination node d. We denote $q(v_i)$ to be the quality value of v_i and $\delta_{i,j}$ to be the walking distance of the road segment $r_{i,j}$. For easier notation, we refer to the nodes of G by using just the index, e.g. i, and to the edges by using the incident nodes, e.g. $\{i, j\}$. In our optimization, we seek a path that maximizes the sum of quality values of visited nodes while at the same time staying below a maximum distance.

We start with a computation of the areas which are influenced by the POIs (Section 4.1). After that we identify the nodes corresponding to the POIs within the road-network graph as well as a destination POI with high rating which is located in an area that is influenced by many POIs (Section 4.2).

Finally, we compute a piecewise linear pedestrian path $\mathbf{p}(t)$ that prefers road segments which are incident to nodes with a high quality value (Section 4.3). The path consists of a sequence of nodes $\mathfrak{P} = \Big(\mathbf{p}(0) = \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{n_{\mathfrak{P}}-1}, \mathbf{p}_{n_{\mathfrak{P}}} = \mathbf{p}(1)\Big)$, and is parameterized according to the normalized distance for $0 \leq t \leq 1$.

4.1 POI Area of Influence

Given the current position of the user $\mathbf{p}(0)$ and the set of queried POIs P, we first define an area of influence by assigning a 2d Gaussian G_i to each of the POIs P_i, whose location is defined by Mercator-projected coordinates denoted by $Merc(P_i)$. We set the center $\mu_i = (x_i, y_i)$ of G_i to be at $Merc(P_i)$ and the amplitude $A_i = G_i(\mu_i)$ to represent the rating (the higher, the better) of P_i. The rating, which we denote by $q(P_i)$, usually ranges from 1 (bad) to 5 (excellent) and is retrieved in our framework from Yelp.

4.2 Source Node and Destination Node

For each POI we identify the corresponding node in the graph, which is either the node $k \in V$ for which the distance between k and the position of the POI is minimal, or a new node that is found by perpendicularly projecting k onto the closest edge $\{i, j\}$, thereby splitting it into two edges $\{i,k\}$ and $\{k, j\}$. The source node s is identified similarly. We denote the set of vertices corresponding to the POIs as $V_P \subseteq V$, re-enumerate all vertices $v \in V$ for easier notation such that v_k corresponds to the POI P_k and assign the quality values $q(v_k) = q(P_k)$.

The destination node d can optionally be chosen by the user. The more interesting case, however, arises when the choice is left to our system. In this case, we do not randomly take one highly rated POI since this could provide a POI in an uninteresting surrounding region with sparsely positioned POIs. In fact, we only consider the subset of highly ranked POIs, which we denote by V_d, but instead of a random choice we take the POI that lies in the region which is most influenced by other POIs, or formally:

$$d = \arg\max_{k \in V_d} \left\{ \sum_{i=1}^{n} G_i \left(Merc\left(P_k\right) \right) \right\}.$$

Note that this formulation does not prevent the destination d to be located in the vicinity of the source node s. Although we could not observe any problems with this formulation in our experiments, it could be easily extended to only consider POIs with a certain minimum distance to s. Alternatively, the threshold defining the maximum walking distance can be set to a big value to compute a reasonable long pedestrian path whose source and destination are close to each other. We refer the interested reader to the supplementary materials for an example in which the distance between the user position and the destination is very small.

4.3 Best Path Estimation

Similar to Kachkaev and Wood [6] we want to find a high-quality path that does not exceed a maximum walking distance and define the quality of a path $q(\mathbf{p})$ as the sum of rating values of visited

POIs. We denote the distance of the shortest path by $\delta_{\min}$ and the maximum walking distance by $\delta_{\max}$.

We formulate the problem of finding a path with the mentioned requirements as a binary integer program (BIP). The main motivation to choose integer programming instead of using simulated annealing [11] or an evolutionary algorithm [8] are the advantages of BIP compared to these methods. It is exact and we can make use of a lot of research optimizing BIP solvers to our benefit, e.g., there are standard solvers like Gurobi that are highly optimized. The problem is still challenging and exact solvers, e.g. building on branch-and-bound, can hit a wall and they are not suitable when the problem gets too large. Then either an approximate BIP solver has to be used, or an alternative approximation technique, e.g. shortest path [6] or dynamic programming [3]. In our problem instance we can observe that the solution quality of approximate solvers degrades significantly, while BIP is still able to compute solutions quickly (cf. Section 9.1).

We introduce a binary variable $x_{i,j}$ for each road segment $r_{i,j}$ where $x_{i,j} = 1$ when the path contains the road segment $\{i, j\}$ and $x_{i,j} = 0$ otherwise. Additionally, we introduce a binary variable y_i for each vertex $i \in V$ where $y_i = 1$ when i is visited and $y_i = 0$ otherwise.

We further formulate the objective function as

$$\max_{y_i} \sum_{i \in V_P} q(P_i) y_i$$

and introduce the following hard constraints:

C1: Both s and d are incident to one edge of the path.

$$\sum_{j \in N(s)} x_{s,j} = 1,\ y_s = 1 \text{ and } \sum_{j \in N(d)} x_{d,j} = 1,\ y_d = 1$$

C2: Each vertex (except s and d) is incident to 0 or 2 edges.

$$\sum_{j \in N(i)} x_{i,j} = 2y_i;\ i \in V \backslash \{s, d\}$$

The variables y_i act as control variables and guarantee that the sum corresponding to the incident edges of a vertex i is either 0 ($y_i = 0$) or 2 ($y_i = 1$) but can not be a different value.

C3: The walking distance must stay below $\delta_{\max}$.

$$\sum_{\{i,j\} \in R} \delta_{i,j} x_{i,j} \leq \delta_{\max}$$

C4: The problem of separate closed circles that was subject in recent work concerning the traveling salesman problem [26] is addressed in our framework by the introduction of a so-called *lazy constraint*. To this end, the routine that starts the process of finding a solution is passed a function pointer. Each time a solution subject to the constraints **C1** to **C3** is found, this function is called and the solution is analyzed if it consists any edge cycles. If this is the case, an additional constraint is added which every upcoming proposed solution has to meet. To give a short example, let O be the set of $|O|$ edges forming an edge cycle within a proposed solution. Then the following constraint is added:

$$\sum_{\{i,j\} \in O} x_{i,j} < |O|$$

4.4 Process of finding a Solution

We solve the optimization problem of finding an interesting pedestrian path using the Gurobi library, a specialized (mixed) integer programming solver, which leverages a linear-programming based branch-and-bound algorithm to find a good feasible solution. We use a default value of $1 \cdot 10^{-4}$ for the Gurobi parameter *MIPGap*, which defines the maximum difference between the lower and upper objective bound at termination. This ensures that the delivered solution is close to optimal.

In Figure 3, a comparison of the path $\mathbf{p}(t)$, that is computed with our method, and the shortest path is shown. All paths shown in this paper were computed optimally in less than 5 seconds. Note that the path with $\delta_{\max} = 1.5 \cdot \delta_{\min}$ shown rightmost in Figure 3 corresponds to the highest quality although the path only crosses a small part of the region of highest POI density that is indicated by the heat map. However, the heat map only shows the values of the sum of Gaussians G_i while the quality of the paths is not defined by the line integral under the Gaussians but is defined by the sum of ranking values of actually visited POIs.

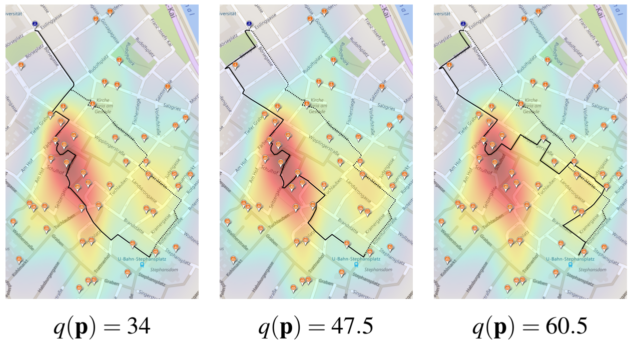


Fig. 3: Comparison of paths which were computed by our system (black) and the shortest path (dotted) to a destination that was chosen by the user. The three shown solutions (from left to right) correspond to maximum walking distances $1.1 \cdot \delta_{\min}$, $1.3 \cdot \delta_{\min}$ and $1.5 \cdot \delta_{\min}$ respectively. The shortest path has a quality of 8.5.

5 Map Cutout Optimization

Regular path visualizations typically place the user position in the middle of the screen. In our system, we want to make better use of screen estate to show important surrounding areas. We do this by adjusting the *map cutouts* used for the visualization. In particular, we compute a separate path $\mathbf{m}(t)$, which describes the motion of map cutouts $M(t)$. We define the map cutout $M(t_0)$ as the rectangular region in the Mercator-projected map (e.g., Bing Maps) that corresponds to the visible map area during the visualization on screen at time t_0.

In order to identify a suitable structure of map cutouts, we experimented with different zoom levels in an early phase of our framework. The implications of changing zoom levels and strategies to handle overlapping map labels were presented by Been et al. [20]. Their labels occupy a fixed size on the screen which makes a removal of unimportant labels during a zoom-out necessary. However, they mainly penalize a mutual overlap of labels but not an overlap of important map content. The occluded map area would increase proportional to the increasing distance to the map representing the zoom-out. Such important map content could be vital for proper navigation as it is the case in our system and should therefore not be occluded.

Therefore, we propose to use labels, or in our case detail lenses, which do not correspond to a fixed size on screen but to a fixed map region. One consequence of this design choice, however, is the apparent shrinking of labels during a zoom-out. We therefore propose to optimize the labeling (cf. Section 6) for a suitable zoom level and to use different levels of detail for the labels. Since one particular zoom level implies a constant width and height of all map cutouts $M(t)$, we can refer to these fixed values by W and H respectively. We could empirically identify a zoom level of 0.694 meters per logical pixel to bring satisfying results and justified our design choice by receiving very positive response from the users of our system (cf. Section 9.3).

Similar to $\mathbf{p}(t)$, $\mathbf{m}(t)$ should satisfy a number of requirements:

1) It should be smooth in order to avoid discontinuities in the movement of the map cutouts.
2) It should pass through regions with high POI density.
3) The movement of the cutout should in general follow the movement of the user (i.e., no backward-movement while the user moves forward). We choose to formalize this by restricting $\mathbf{m}(t)$ to lie on a line orthogonal to $\mathbf{p}(t)$.

We start by computing the POIs that are close enough to $\mathbf{p}(t)$ to be theoretically visible in at least one map cutout $M(t)$. Next, we identify regions (called *hot spots*) with a high density of potentially visible POIs and compute a cubic spline that touches the hot spots. Finally, we choose a parametrization for $\mathbf{m}(t)$ that satisfies requirement 3. The details about the computation are given in the following paragraphs.

5.1 Potentially Visible POIs

We start the map cutout optimization by finding those POIs which have a high probability of being visible in the final visualization. Due to the design of our optimization approach, it is not possible to tell immediately which POIs will be visible and which will not. We therefore identify the set of *potentially* visible POIs, which we denote by $P_v \subseteq P$. To this end we project each POI $p \in P$ orthogonally onto the closest point on $\mathbf{p}(t)$ to identify the point $\mathbf{p}_p$. We only add p to the set P_v if we can find a map cutout that allows visibility of both p and $\mathbf{p}_p$. This can be easily discovered by testing if the difference of the x-values of p and $\mathbf{p}_p$ is $\leq W$ and the difference of the y-values is $\leq H$ (cf. Figure 4).

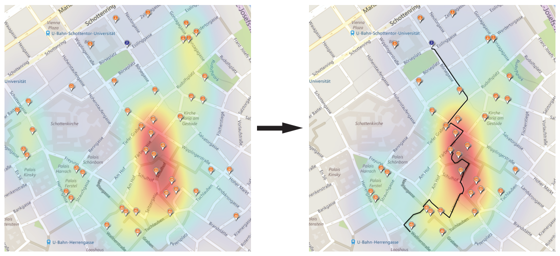


Fig. 4: The set of POIs P is reduced to the set P_v of potentially visible POIs. We identify elements of P_v by finding those POIs that have a certain proximity to the path (shown in black).

5.2 Map Cutout Path

With the potentially visible POIs at hand, we now seek the path $\mathbf{m}(t)$ that describes the motion of the map cutouts $M(t)$. We start by taking the sequence of nodes $\mathfrak{P}$ along the path $\mathbf{p}(t)$ and also identify the parameters $t_{\mathbf{p}_i}$ of the nodes $\mathbf{p}_i$ with $\mathbf{p}_i = \mathbf{p}(t_{\mathbf{p}_i})$.

We tackle point (1) of the mentioned requirements by defining $\mathbf{m}(t)$ as a cubic spline, with control points $\mathfrak{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_{n_\mathfrak{C}})$. Point (2) is considered by positioning these control points at the POI *hot-spots* near $\mathbf{p}(t)$. With hot-spots we refer to points in the environment which provide a local maximum of the sum of Gaussians G_i. The requirement mentioned in point (3) is satisfied only approximately for simplicity reasons. In particular, we enforce it only at the control points $\mathbf{c}_i$.

For the computation of the cubic spline representing $\mathbf{m}(t)$, we do not directly set the location of the control points $\mathfrak{C}$, but initially determine a set of regularly distributed points $\mathbf{f}_i$ along the path $\mathbf{p}(t)$. There is, however, an inherent connection between the points $\mathbf{f}_i$ and the control points $\mathbf{c}_i$, since each $\mathbf{f}_i$ has – after the locations of the control points $\mathbf{c}_i$ are determined in the next step – its corresponding control point $\mathbf{c}_i$ and defines the parameter t that is assigned to $\mathbf{c}_i$.

We position the points $\mathbf{f}_i$ on the path $\mathbf{p}(t)$ such that $\mathbf{f}_i = \mathbf{p}(i\Delta t)$ and choose a regular parameter interval Δt between two points $\mathbf{f}_i$ and $\mathbf{f}_{i+1}$ that yields a walking distance of about 40 meters, which we found to be a good trade-off between control and providing enough freedom to the cubic spline. Since the number of points $\mathbf{f}_i$ equals the number of control points $\mathbf{c}_i$, the parameter interval Δt between the points $\mathbf{f}_i$ defines the number of control points $n_\mathfrak{C} = 1 + {}^1\!/\!_{\Delta t}$.

After the evaluation of the points $\mathbf{f}_i$, we calculate the control points $\mathbf{c}_i$. At each point $\mathbf{f}_i = \mathbf{p}(i\Delta t)$ we identify the ancestor node $\mathbf{p}_j = \mathbf{p}(t_{\mathbf{p}_j})$ with $t_{\mathbf{p}_j} = \max\{t_{\mathbf{p}_k} \mid t_{\mathbf{p}_k} \leq i\Delta t\}$ and descendant node $\mathbf{p}_{j+1}$ and denote the normal to the path at $\mathbf{f}_i$ as $\mathbf{n}_i = \frac{(\mathbf{p}_{j+1}-\mathbf{p}_j)^\perp}{\|\mathbf{p}_{j+1}-\mathbf{p}_j\|}$. Next, we restrict the control point $\mathbf{c}_i$ that corresponds to $\mathbf{f}_i$ to lie on the line $\mathbf{l}_i(\lambda) = \mathbf{f}_i + \lambda\mathbf{n}_i$ with $\lambda \in \mathbb{R}$, thus respecting requirement (3) for this control point. Since each control point $\mathbf{c}_i$ is connected to its corresponding point $\mathbf{f}_i$ perpendicularly to the path, we refer to the points $\mathbf{f}_i$ as the foot points $\mathfrak{F} = (\mathbf{f}_1, \ldots, \mathbf{f}_{n_\mathfrak{C}})$. Since the position of the user should be visible at any frame of the visualization, we further restrict the parameter λ to be inside the interval $\left[\lambda_i^{\min}, \lambda_i^{\max}\right]$, where $\lambda_i^{\min}$ and $\lambda_i^{\max}$ correspond to the minimum and maximum value of the parameter λ respectively such that the user position at $\mathbf{p}(i\Delta t)$ is still visible inside the map cutout $M(i\Delta t)$. To identify the best location for $\mathbf{c}_i$ corresponding to the POI hot spot on the line segment $\mathbf{l}_i(\lambda)$, thereby considering requirement (2), we define a quality measure $q_i(\lambda)$ as

$$q_i(\lambda) = \xi \sum_{j=1}^{n} G_j(\mathbf{l}_i(\lambda)) + (1-\xi) G_{\mathbf{f}_i}(\mathbf{l}_i(\lambda))$$

where $\xi \in [0,1]$ is a weighting parameter (usually set to 0.8) to define the importance of the map cutout offset towards the POIs in the environment, and $G_{\mathbf{f}_i}$ is a 2d Gaussian centered at $\mathbf{f}_i$ which acts as a counterpart to the POI Gaussians G_j and forces the map cutout to have the user position in its center at sections of the path where there are no POIs in the environment. We then position $\mathbf{c}_i$ at the point on $\mathbf{l}_i(\lambda)$ that corresponds to the biggest quality value:

$$\mathbf{c}_i = \mathbf{l}_i(\lambda_i^*) \,\text{with}\, \lambda_i^* = \arg\max_{\lambda} \{q_i(\lambda)\}\,; \lambda \in \left[\lambda_i^{\min}, \lambda_i^{\max}\right].$$

In order to make sure that the footpoints indeed correspond to the control points in the animation, we assign the parameter values $i\Delta t$ to the control points $\mathbf{c}_i$ so that $\mathbf{m}(i\Delta t) = \mathbf{c}_i$ and evaluate the now fully defined cubic spline $\mathbf{m}(t)$. The values λ_i^* are found by a linear search. Figure 5 illustrates the computation of $\mathbf{m}(t)$.

As already mentioned, it is not guaranteed that all the POIs which were initially added to the set P_v of potentially visible POIs are also visible in at least one map cutout $M(t)$. Those POIs whose corresponding map pins are not fully visible in at least one map cutout $M(t)$ are removed from the set P_v. The resulting set of remaining POIs to be used in the next section is denoted as P^*. For ease of notation, we re-index the remaining POIs in P^* such that $P^* = \{P_1, \ldots, P_{n^*}\}$.

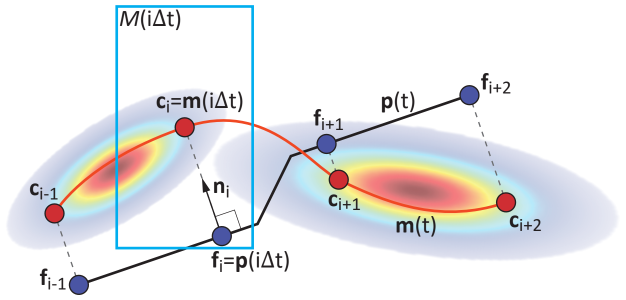


Fig. 5: Computation of the map cutout path $\mathbf{m}(t)$ whose purpose is to emphasize interesting regions in the environment of $\mathbf{p}(t)$ during the dynamic visualization.

6 Detail Lens Placement

With the computed paths $\mathbf{p}(t)$ and $\mathbf{m}(t)$ at hand, we can already present a dynamic visualization to the user that emphasizes regions of high POI density by shifting the midpoint of the viewport corresponding to the current map cutout towards those regions, while at the same time making sure the user position remains visible. This visualization, however, does not give any information about the POIs (e.g. name, category, rating), which is important to give the user a clue whether or not the POIs should be visited.

We therefore enhance this visualization with rectangular entities which we call detail lenses. Each POI corresponds to exactly one detail lens, which – in our implementation – contains the name of the POI as well as a photo and a rating value. Technically, we cast the problem as an optimization task based on a binary integer program (BIP). The goal is to find a dynamic layout in which the detail lenses are positioned as close as possible to the corresponding POIs, occlusions of important areas are avoided, and the time span in which the information is visible is maximized.

6.1 Path Discretization

To simplify the optimization approach for the detail-lens placement, we discretize the path $\mathbf{m}(t)$ by splitting it into consecutive parts of equal arc length. We usually choose this arc length to be about 40 meters and denote the resulting sequence of nodes along the curve $\mathbf{m}(t)$ as $\mathfrak{M} = \left(\mathbf{m}(0) = \mathbf{m}_1, \ldots, \mathbf{m}_{n_\mathfrak{M}} = \mathbf{m}(1)\right)$ with corresponding map cutouts $\left(M(0) = M_1, \ldots, M_{n_\mathfrak{M}} = M(1)\right)$. We further evaluate the parameter values $t_{\mathbf{m}_\tau}$ such that $\mathbf{m}_\tau = \mathbf{m}(t_{\mathbf{m}_\tau})$ and refer to the index τ as the *frame number* (or shortly just *frame*) in the discrete optimization. $\tau_{\max} = n_\mathfrak{M}$ is the last frame.

6.2 Cell Domain

To allow an easier and structured placement of detail lenses, we follow the design guidelines of recent work (see [15] and [24]) and partition the map into a 2d Cartesian grid and choose the side length of the cells so that a small integer number of cells results in well-visible detail lenses. We identified a value of about 40 meters to deliver good results. To keep the following optimization problem as compact as possible, we do not consider an infinite extent of the Cartesian grid but identify the set of those cells that are relevant to the visualization. To this end, we first identify the rectangular blocks of connected cells that are completely visible inside each discretized map cutout M_τ, which we denote by C_τ. We further denote the x- and y-coordinate of one particular cell c as x_c and y_c respectively. The whole cell domain consisting of the union of all visible cells considered in the optimization is denoted by C (cf. Figure 2).

6.3 Detail Lenses

After the cell domain has been identified, we choose a layout for the detail lenses that allows an easy match to the structure of the 2d Cartesian grid. To this end, we restrict the detail lenses to have a side length that corresponds to an integer multiple of cells. We call W_L and H_L to the number of cells along the horizontal and vertical axis respectively. Usually, we use square detail lenses by defining their size to be 2 cells both along the horizontal and vertical axis ($W_L = H_L = 2$). In combination with our standard values for the cell size (40m by 40m) and a zoom level of 0.694 meters per logical pixel, this yields a visible side length on the iPhone 6+ of about 20mm, which we found to be a good trade-off between readability and space consumption. We also evaluated our design choice in a user study (cf. Section 9.3).

We create one detail lens L_i for each $P_i \in P^*$. The placement of a lens L_i at a grid cell $c \in C$ with coordinates (x_c, y_c), also written as $L_i(x_c, y_c)$, is defined so that the lower left corner of L_i coincides with the lower left corner of the cell c.

6.4 Binary Integer Program

We formulate the problem of optimized detail lens placement as a binary integer program. Our design choice to use integer programming is mainly motivated by similar arguments as were presented in Section 4.3. We introduce a set of binary variables v_{τ,i,x_c,y_c}, where $v_{\tau,i,x_c,y_c} = 1$ when the detail lens L_i is visible at frame τ and placed at grid cell c and $v_{\tau,i,x_c,y_c} = 0$ otherwise.

We further formulate the objective function as

$$\max_{v_{\tau,i,x_c,y_c}} \sum_{\tau=1}^{\tau_{\max}} \sum_{i=1}^{n^*} \sum_{c \in C} q_{\tau,i,x_c,y_c} v_{\tau,i,x_c,y_c}$$

where q_{τ,i,x_c,y_c} is the quality value (the higher, the better) for placing the detail lens L_i at frame τ at cell c. We follow approved guidelines proposed in [9] and intend to place detail lenses close to their corresponding POIs, but avoid obstructing important map detail. For this, we define the quality value as

$$q_{\tau,i,x_c,y_c} = d(L_i(x_c,y_c), P_i)^{-1} (1 - o(L_i(x_c,y_c)))^e$$

where $d(L_i(x_c,y_c), P_i)$ is the Euclidean distance between the midpoint of L_i placed at grid cell c and the POI P_i, both in meters with respect to the Mercator-projected map, and $o(L_i(x_c,y_c))$ is a function returning a value $\in [0,1]$ referring to the amount of occlusion of important map content when L_i is placed at cell c. With important map content we refer to roads which are crucial for navigation and emphasize the penalization of the occlusion of such areas by introducing the exponent e, which we usually set to 10. We refer the interested reader to the supplementary materials to obtain details how to compute $o(L_i(x_c,y_c))$.

In order to enhance the performance of the process of finding a solution, we exclude all variables from the definition of the BIP which are known to be 0. In detail, we exclude variables that would lead to an overlap of important information (pedestrian position pin or POI pin) by a lens and variables that would lead to a placement of detail lenses which are (partially) outside of the screen. For all the presented results we rigorously make use of variable exclusion since we observed a significant performance boost when the number of variables was kept as low as possible.

6.5 Hard Constraints

We introduce the following hard constraints for our optimization problem. Notice that some variables which are referenced in the definition of the hard constraints may not exist due to the variable exclusion. Those variables are then simply ignored.

C1: Each detail lens L_i is shown at most once at a time.

$$\sum_{c\in C} v_{\tau,i,x_c,y_c} \le 1;\ \tau\in[1,\tau_{\max}];\ i\in\left[1,n^*\right]$$

C2: No two detail lenses may overlap.

$$\sum_{i=1}^{n^*}\sum_{x=x_c-W_L+1}^{x_c}\sum_{y=y_c-W_L+1}^{y_c} v_{\tau,i,x,y}\le 1$$
$$\tau\in[1,\tau_{\max}];\ c\in C_\tau$$

With the $\sum_{\tau=1}^{\tau_{\max}}|C_\tau|$ hard constraints of type **C2** we avoid an overlap of any pair of detail lenses. This goal is achieved by allowing at most one variable v_{τ,i,x_c,y_c} that is affecting a grid cell c at frame τ to be 1. Note that it is not sufficient to restrict only the variables directly referring to c, since lenses which are bigger than 1×1 can affect a grid cell also when they are placed in a neighbor cell. Therefore we have to restrict the whole area of neighbor cells in which a placement of a lens L_i would occupy the grid cell c.

C3: To avoid jumps between different positions, each detail lens is shown at just one position on the map.

$$\sum_{\tau'=\tau+1}^{\tau_{\max}}\sum_{c'\in C\backslash c} v_{\tau',i,x_{c'},y_{c'}} - \mathrm{M}\cdot(1-v_{\tau,i,x_c,y_c})\le 0$$
$$\mathrm{M}=\tau_{\max}\cdot|C|;\ \tau\in[1,\tau_{\max}-1];\ i\in\left[1,n^*\right];\ c\in C$$

With the $(\tau_{\max}-1)\cdot n^*\cdot|C|$ hard constraints of type **C3** we ensure that if a detail lens L_i is shown at frame τ at cell c, thereby setting $v_{\tau,i,x_c,y_c}=1$, it is restricted to stay at c and can therefore not be shown at any cell $c'\in C\backslash c$ for any future frame τ'. For this definition we introduce a big constant M that helps us to achieve the desired goal: When a detail lens L_i is shown at frame τ at cell c, thereby setting $v_{\tau,i,x_c,y_c}=1$, the term including M becomes 0, thereby forcing all the other variables in the sum to be 0 in order to meet the constraint. When $v_{\tau,i,x_c,y_c}=0$, the constraint is always met.

C4: To avoid any flickering artifacts caused by permanent appearance and disappearance, each detail lens is shown in just one contiguous block of frames. To meet this constraint, we have to introduce a set of new binary variables d_{τ,i,x_c,y_c}, which represent the absolute value of differences between neighboring variables v_{τ,i,x_c,y_c} and $v_{\tau-1,i,x_c,y_c}$ with respect to the time axis. Further, we have to introduce the following constraints to the BIP:

$$v_{\tau-1,i,x_c,y_c}-v_{\tau,i,x_c,y_c}\le d_{\tau,i,x_c,y_c}\ge v_{\tau,i,x_c,y_c}-v_{\tau-1,i,x_c,y_c}$$
$$i\in\left[1,n^*\right];\tau\in[2,\tau_{\max}];c\in C$$

With these new variables at hand, we can introduce the actual constraints. A more detailed derivation of the hard constraints of type **C4** are given in the supplementary material.

$$v_{1,i,x_c,y_c}+\sum_{\tau=2}^{\tau_{max}} d_{\tau,i,x_c,y_c}+v_{\tau_{\max},i,x_c,y_c}=2$$
$$i\in\left[1,n^*\right];\ c\in C$$

The BIP can be solved using the presented hard constraints in order to get a satisfactory result for the placement of the detail lenses. We can however accelerate our approach significantly by the introduction of two strategies which will be presented in the next two sections.

6.6 BIP Simplification

Although the goals of all the presented hard constraints are needed to get the desired result, it is possible to simplify the whole binary integer program by merging the majority of the constraints. In detail, it is possible to merge the hard constraints **C1**, **C3**, and **C4**:

$$\sum_{c\in C}\left(v_{1,i,x_c,y_c}+\sum_{\tau=2}^{\tau_{\max}} d_{\tau,i,x_c,y_c}+v_{\tau,i,x_c,y_c}\right)=2;\ i\in\left[1,n^*\right]$$

The merge arises from the hard constraints **C4** by defining one constraint for a detail lens L_i not per grid cell c, but to have just one constraint per detail lens by taking the sum over all the left-hand sides of **C4** corresponding to L_i, and keeping the result on the right-hand side of the equation to be 2. A proof of the equivalence of the individual and merged constraints is given in the supplementary material.

Note that merging the hard constraints of type **C1**, **C3**, and **C4** does not influence the complexity of the problem or the accuracy of the solutions. However, we could observe a significant boost concerning the runtime and therefore present all the timings in this paper with respect to the merged definition.

6.7 Incremental Approach

There is still room for improvement since we can divide the whole problem into a sequence of small problems with boundary constraints. This not only decreases the computation time for the whole problem, it is further possible to present solutions of subproblems to the user before all subproblems are solved. The division into subproblems particularly makes sense if not the whole solution is needed immediately, but it is sufficient to provide a partial solution corresponding to the beginning of the path first, and then to gradually provide solutions to the following parts.

We use an incremental approach and start with a subdivision of the sequence of nodes $\mathfrak{M}$ into a sequence of n_I subsequences of nodes

$$\mathfrak{M}^{n_I}=((\mathbf{m}_1,\ldots,\mathbf{m}_{K_1}),(\mathbf{m}_{K_1},\ldots,\mathbf{m}_{K_2}),\ldots,(\mathbf{m}_{K_{n_I-1}},\ldots,\mathbf{m}_{\tau_{\max}}))$$

such that the last node in each subsequence is the first node in the next subsequence and denote the indices K_1 to K_{n_I-1} to be the *key frames*. Usually, we choose the number of subsequences n_I to be about 4 to 8 and the key frames such that the number of nodes in each subsequence is approximately the same.

Then we initially create a BIP, which we denote as BIP_1, similar to the definition proposed in Section 6.4, but only include variables and hard constraints which correspond to the frames $\tau \in [1, K_1]$. The solution, which can be evaluated much faster due to the smaller complexity of BIP_1, can then be directly presented to the user. While this part is visualized, we create and solve BIP_2 for the second part corresponding to frames $\tau \in [K_1, K_2]$. This process is then continued until a solution for BIP_{n_I} for the last part corresponding to the frames $\tau \in [K_{n_I - 1}, \tau_{\max}]$ is found.

Since the end of one part and the beginning of the next part must perfectly fit in order to avoid any visualization artifacts in the transition, we have to introduce a few more hard constraints for all BIP_k with $k > 1$ to get a smooth transition between the results of two consecutive binary integer programs BIP_k and BIP_{k+1}:

C5: If L_i is visible in the last frame of BIP_k, due to the one-frame-overlap it must be visible in the first frame of BIP_{k+1}.

$$\begin{gathered} \text{if } v_{K_k,i,x_c,y_c} = 1 \text{ in BIP}_k \\ \text{then } v_{K_k,i,x_c,y_c} = 1 \text{ in BIP}_{k+1} \\ k \in [1, n_I - 1]; i \in [1, n^*]; c \in C \end{gathered}$$

Note that this is not an informal description of a hard constraint including an if-statement, since this constraint is set for BIP_{k+1} depending on the outcome of BIP_k.

C6: If a detail lens L_i is not visible in the last frame of BIP_k but was already visible in any frame of any BIP_j for $1 \le j \le k$, then it will remain invisible for all BIP_j for $j > k$.

$$\begin{gathered} \text{if } v_{K_k,i,x_c,y_c} = 0 \text{ in BIP}_k \text{ and } \exists \tau \le K_k - 1 \middle| v_{\tau,i,x_c,y_c} = 1 \\ \text{then } v_{\tau,i,x_c,y_c} = 0 \text{ in BIP}_j \text{ for } j > k \text{ and } \tau > K_k \\ k \in [1, n_I - 1]; i \in [1, n^*]; c \in C \end{gathered}$$

6.8 Process of finding a Solution

Again, we solve the optimization problem of placing the detail lenses using the Gurobi library (cf. Section 4.4). When a solution for the BIP is found, the resulting layout ensures that each detail lens L_i is shown as long as possible and as close as possible to its corresponding POI P_i, thereby avoiding any mutual overlaps or occlusions of any map pins.

7 Visualization

In our current implementation, we number the map pins w.r.t. the appearance time of the corresponding lenses and always fade in and fade out each corresponding pair (map pin and lens) simultaneously. However, in the future we want to explore the impact of different numbering orders, e.g. according to geographic coordinates or in the chronological order in which the POI positions get visible.

For the final visualization, the viewport defined by the map cutout layout follows the path $\mathbf{m}(t)$, while the user position is shown at $\mathbf{p}(t)$ using a standard map pin. The parameter t is either defined by the real position of the user who follows the path $\mathbf{p}(t)$, or it is automatically increased by our system according to a user-chosen speed function $s(t)$. In our experiments we usually choose a constant speed function $s(t) = c$ to resemble a uniform walking speed (e.g., $c = 5\frac{m}{s}$). However, our framework could be easily extended to allow non-uniform speed functions $s(t)$ that allow a slow-down in interesting regions and a speed-up in sparse regions.

To visualize the detail lenses, the values of the binary variables as they are delivered by the BIP are taken into account. Let L_i be a detail lens whose visibility at grid cell c is evaluated between frames τ and $\tau + 1$. The trivial case is when both $v_{\tau,i,x_c,y_c} = v_{\tau+1,i,x_c,y_c} = 1$, which implies that L_i is visible in the parameter interval $\left[t_{\mathbf{m}_\tau}, t_{\mathbf{m}_{\tau+1}}\right]$. When $v_{\tau,i,x_c,y_c} = 0$ and $v_{\tau+1,i,x_c,y_c} = 1$, the detail lens L_i changes its state from invisible at $t_{\mathbf{m}_\tau}$ to visible at $t_{\mathbf{m}_{\tau+1}}$. We use a simple linear interpolation between according values of the alpha channel or scale factors 0 and 1 to get a smooth transition between the visibility states, but to avoid any overlaps with any other lenses, we usually start the transition not at $t_{\mathbf{m}_\tau}$ but at $\frac{t_{\mathbf{m}_\tau} + t_{\mathbf{m}_{\tau+1}}}{2}$. For similar reasons we finish the transition at $\frac{t_{\mathbf{m}_\tau} + t_{\mathbf{m}_{\tau+1}}}{2}$ if a detail lens L_i changes its state from visible at $t_{\mathbf{m}_\tau}$ to invisible at $t_{\mathbf{m}_{\tau+1}}$. However, in a post-processing step we check if the time span of visibility can be increased without introducing any overlaps, and perform an early fade-in between $t_{\mathbf{m}_\tau}$ and $\frac{t_{\mathbf{m}_\tau} + t_{\mathbf{m}_{\tau+1}}}{2}$ and a late fade-out between $\frac{t_{\mathbf{m}_\tau} + t_{\mathbf{m}_{\tau+1}}}{2}$ and $t_{\mathbf{m}_{\tau+1}}$ whenever this is possible.

Figure 6 shows a small example of a visualization and how the transition of visibility states allows the utilization of the chronological dimension to better exploit available map space and to avoid visible overlaps. The example shows the detail lenses L_{23} and L_{26} that would partially overlap if they would not be shown after each other.

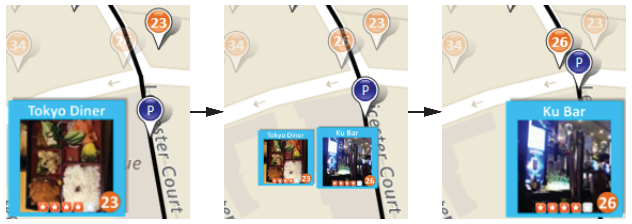


Fig. 6: By showing detail lenses in different periods of time, our visualizations utilize the chronological dimension to better exploit available map space. In this example, the detail lens L_{23} is shown first (left), but then disappears while at the same time the lens L_{26} appears (middle). After this transition, only L_{26} is shown (right).

We further analyzed the situation of a user that does not follow the path that was computed by our system but decides to choose a road segment into a different direction. This would require some kind of re-routing by a new call of our system with the chosen road segment as an initial constraint. Since this would however not implicate any new technical problems which are not considered in our framework, we did not further investigate this issue.

7.1 User Interaction

One of the general principles for standard map labeling addresses the problem of how to make it easier for users to recognize the labels for certain features in the map [9]. While these guidelines mainly focus on the distance and relative orientation between labels and features, other presented methods address this correspondence problem by using line segments to connect the corresponding objects [22], [27]. To avoid visual clutter, we desist from using such line segments but leverage the interactive component of our system to emphasize corresponding pairs of information. In detail, the user can easily find a detail lens for a particular map pin or vice versa a map pin for a detail lens by hovering with a pointer above one of the corresponding items. As a result, the opacity values of all other elements except this pair of interest are reduced to make an identification easy. Furthermore, the user can use a set of standard interface elements to control the speed and direction of the visualization. With these controls it is also possible to pause the animation to explore a static view in more detail (cf. Figure 7 middle and right). If the user wants to have more detailed information about a POI, he can click on a detail lens to be presented with a more detailed view (cf. Figure 1).

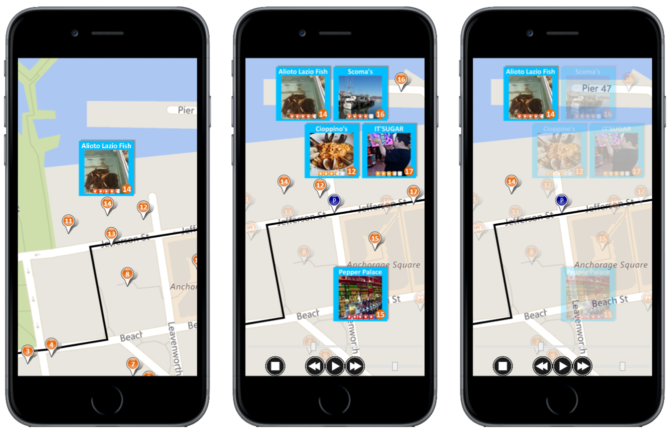


Fig. 7: (left) Re-implemented functionality of Yelp in our framework showing a detail lens when the corresponding map pin is clicked. (middle) Similar view but with our proposed visualization and visible control elements. (right) To easily recognize corresponding map pins and detail lenses the user can hover over one of the elements to reduce the opacity of all other elements.

8 Implementation

The current prototype of our framework is implemented with XAML and WPF in C#. In order to allow a pixel-accurate visualization on a PC monitor as it would be performed on a screen of a mobile device, we use the high-PPI map tiles of Bing Maps, which are designed for devices like smartphones and tablets. Further, we choose an area on the PC monitor for the visualization that exactly resembles the logical resolution of the corresponding mobile device.

The vector-based map data that we need to extract the route segments is obtained from the OpenStreetMap (OSM) database. All data about the points of interest is queried from YELP.

For the future, we want to port our framework to completely run on mobile devices and to replace the prototypic control elements discussed in Section 7.1 by intuitive gestures. We do not expect any performance issues since we plan to install Gurobi on a server, which is then called by its mobile clients.

9 Evaluation

9.1 Path Computation

We compared our approach for the path computation with several recently proposed methods. In detail, we compared to Mooney and Winstanley [8], Lu et al. [3] and Kachkaev and Wood [6] since they propose competing approaches to find a high-quality path. We tested all algorithms on a small toy example (cf. Figure 8) as well as in a real street network. The toy example represents a small graph consisting of five nodes and six edges, each having a distance value δ and quality value q assigned. Note that this

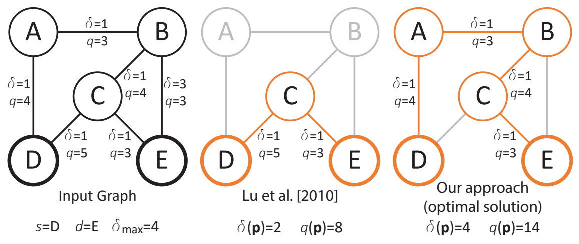


Fig. 8: (left) Toy example to demonstrate the performance of different path computation algorithms. (middle) The method from Lu et al. [3] only finds a suboptimal solution due to disrespect of the suboptimal partial solution for $\delta_{\max} = 2$ from D to B via A which is however part of the optimal solution (right) for $\delta_{\max} = 4$ from D to E that is successfully found by our approach.

differs from the problem definition stated in Section 4.3 where the quality values were assigned to the vertices. However, we choose this definition here to be consistent with the other methods. Further note that the toy example does not claim to be a representative small-scale model of a real street network but is only used to give the reader an impression of the shortcomings of the competing algorithms. The real street network in our comparison corresponds to our Vienna dataset shown in Figure 2. We tested the performance of all approaches for computing a high-quality path between two preselected nodes given a maximum walking distance $\delta_{\max}$.

Surprisingly, the dynamic programming approach of Lu et al. [3] does not produce competitive results, despite its very high running time and memory consumption. We illustrate the problem in Figure 8. Since their method only stores the path with highest quality between all node pairs, it misses the sub-optimal sub-path $D-A-B$ which is however part of the optimal solution. We also observe poor performance in our real-life scenario, particularly for maximum walking distances that are only slightly longer than the shortest path. Moreover, due to the high complexity level we observed runtimes taking several hours with a directly proportional relation between maximum path length and runtime (cf. Figure 9).

The approach of Kachkaev and Wood [6] leverages Dijkstra's shortest path algorithm in an iterative manner with special edge weights to find a high-quality path. The heuristic is fast, but it also does not perform as well as expected. In particular, our results in Figure 9 reveal that the heuristic does not work for long maximum walking distances. In addition, the method also fails on our toy example in Figure 8.

The only approach beside ours that was able to find the optimal solution in our toy example is the one proposed by Mooney and Winstanley [8]. Their method is based on an evolutionary algorithm that produces a predefined number of generations of feasible solutions. Each generation evolves from applying a mutation and crossover operator to the previous generation together with the best solutions found by random walk. The approach is able to find the optimal solution in the toy example. Also in a realistic scenario their approach finds paths of good quality. However, we could observe a bad runtime performance for short maximum walking distances (cf. Figure 9). The main problem here is that it is difficult to find a path by random walk that meets very strict maximum walking distance constraints specified by the user.

As can be seen in Figures 8 and 9, our approach is the only one that always finds the optimal solution in a reasonable time ($< 5s$), making it suitable for the real-time requirement of our setup.

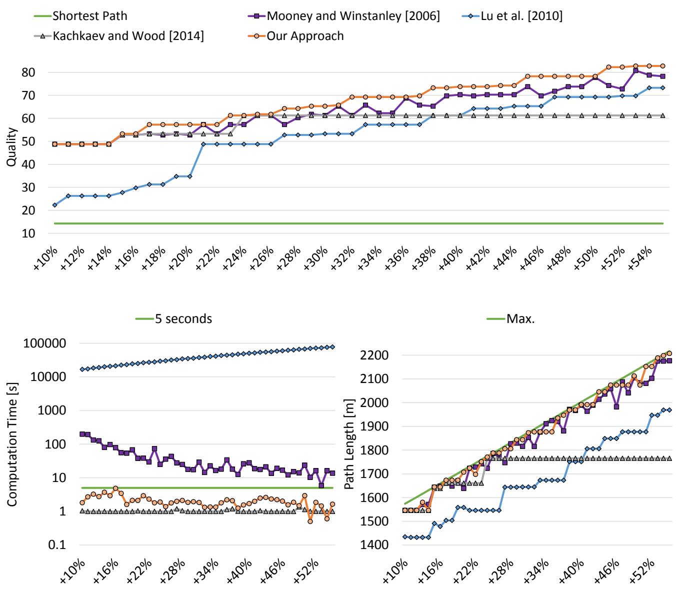


Fig. 9: Comparison of our path computation approach with several recently proposed methods. The horizontal axis corresponds to the maximum walking distance in relation to the shortest path.

The only approach that was faster than ours is the Dijkstra-based method from Kachkaev and Wood [6], which is however very limited in finding high-quality paths for large maximum distances. In summary, our BIP-approach performs superior in comparison to recently proposed methods used for computation of high-quality paths. We refer the interested reader to the supplementary materials for more details about the competing methods.

9.2 Detail Lens Placement

To evaluate our method for detail lens placement, we carried out a user study in which we tested the preference of users for different methods for presenting information about points of interest in visualizations similar to ours. To this end, we showed pairs of short videos, with each video in a pair showing a method different from the other, to 32 participants in an online test and asked them which method they would prefer if they were using it as a guide in an unfamiliar city. We used our framework to generate results for three different paths in three different cities and arbitrarily chose London, New York and Vienna for this purpose. For every path we queried 100 POIs in the environment of the path origin and used our methods for path computation and map cutout optimization. Since the goal of the user study was to test the preference for different methods for presenting information about POIs, we used five different methods for the detail lens placement:

1) Complete optimized placement (cf. Section 6).
2) Incremental placement with 4 parts (cf. Section 6.7).
3) Incremental placement with 8 parts (cf. Section 6.7).
4) Naïve placement per frame without temporal coherence. To this end a BIP is solved independently for each frame, without caring about the transition between the results of two consecutive BIPs.
5) Complete optimized placement in screen space.

We used modified versions of our approach to have a basis for comparison. For method (5), we used our approach as it was described in Section 6 but did not use a discrete grid in the map, but partitioned the screen into as-square-as-possible sized cells which best resembled the visible cell size of the other solutions. In contrast to the other solutions, the detail lenses do not move during the visualization when they are placed in screen space but stay on a fixed position on the screen. With “frames” for method (4) we refer to the finite number of map cutouts along $\mathbf{m}(t)$.

Altogether we generated 15 results (3 paths $\times$ 5 methods for detail lens placement) and extracted 3 10-second long video sequences corresponding to 3 non-overlapping path segments from each result, yielding 45 video sequences. We restricted the comparable pairs of video sequences to those which belong to the same path segment, resulting in $3 \times 3 \times \binom{5}{2} = 90$ comparable pairs. To keep the time span per user in an adequate range, we reduced the number of pairs which were shown to each participant to 50. These 50 samples were chosen randomly, but in order to avoid favoring any of the algorithms, we ensured that each algorithm was presented equally often to each participant.

The results of each participant k were translated into an off-diagonal 5×5 matrix M^k with entries M^k_{ij} corresponding to the number of times method j was preferred to method i. By summing up the columns, we identified the number of points which were distributed to the methods by the participants. The point distributions of all 32 participants were then merged and analyzed by performing an ANOVA whose outcome was used to perform Tukey’s honest significance test.

As shown in Fig. 10, our method is preferred significantly by the participants to a screen-space solution and a naïve per-frame solution. Although there is a noticeable trend that the quality of the result decreases proportionally with the number of parts in the incremental approach, we could not identify a significant acceptance drop for 4 and 8 parts. This in turn means that one can use the incremental approach with 8 parts to get a reasonable result in 10% of the time (cf. Fig. 13).

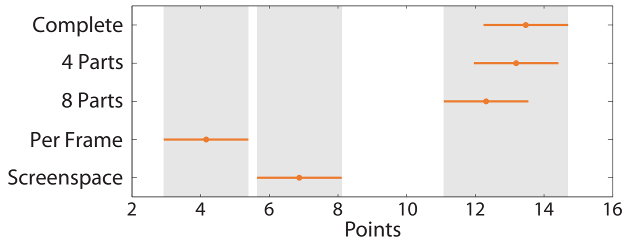


Fig. 10: Results of our user study with 32 participants. Our method is preferred significantly by the users to a screen space solution and a naïve solution in which the detail lenses are positioned without temporal coherence per frame. The orange dots with the connected lines show mean and standard errors of points the users assigned to the methods. The gray bars show non-overlapping point intervals and therefore significant difference.

9.3 Usability Test and Comparison

In order to evaluate the usability of our system, we performed another user study whose aim was

1) to see if our method appeals to users and
2) to compare it to an existing method that is widely used to perform similar exploration tasks.

To this end we asked the participants to use both our and the existing method to perform several predefined exploration tasks.

Due to its success with over 10 million downloads, very positive reviews (4.5 of 5 stars at Google Play) and a growing community we chose the functionalities of the Yelp app to represent the current state of the art. In order to allow an unbiased comparison we did not use the Yelp app directly, but re-implemented its functionality into our framework. This allowed us to present the same set of POIs and computed path of our system but using the visualization techniques of Yelp. In detail, we implemented the map view of the Yelp app that shows the queried POIs on the map using standard map pins. In contrast to our system that shows a precomputed animation, using Yelp the user has to click on a pin to get information about the corresponding POI. Moreover, we did not try to recreate the shape and appearance of the information blocks that are displayed in the Yelp app after clicking on a pin but used our detail lenses instead to avoid any bias caused by a possible preference for either of the shapes (cf. Figure 7). For easier notation, in the following we refer to the Yelp functionalities in our framework just by using the term Yelp.

City	#POIs	Exploration Task
London	89	Find an Algerian café
Manhattan	74	Find a café close to the path serving Belgian beer
Paris	32	Find a restaurant that serves Ramen
San Francisco	54	Find a restaurant with a rating of at least 4 stars that serves Pizza

TABLE 1: Cities we chose as examples in the second user study with corresponding number of POIs and exploration tasks containing descriptions of the requested POIs that were handed to the participants. We measured the time the participants needed to find the corresponding POIs for a quantitative comparison of the methods. Figure 11 shows the results of these measurements.

We arbitrarily chose four cities as examples and computed one path for each city. For each path we used a different number of POIs to demonstrate the behavior of the methods for differently complex scenarios. Furthermore, we randomly picked a POI representing a restaurant from the environment of each path and formulated an exploration task containing a short description of the POI that was handed to the participants during the study (cf. Table 1). We only used restaurants to avoid any bias caused by any knowledge of the users about the prominent sights in the environment of the paths. Each participant was asked to perform the corresponding exploration task for each path using either our system or Yelp. We measured the performance of either method by measuring the time the participants needed to find the requested POI and to click on the corresponding detail lens. We avoided a preference by choosing both the order of the paths as well as the method randomly and referred to them in an anonymous way by calling them "Method 1" and "Method 2". We assured that each participant had to use each method equally often and that after n participants $n/2$ would have used Yelp for one particular path and $n/2$ our method. Furthermore, for both methods we initialized the view to show the origin of the path in the middle of the screen.

Altogether we could acquire twelve participants for our second user study, ten male and two female, all between 20 and 40 years old. Nine of the participants were graduate students, two were university staff and one female was a technical professional. After the measurements we removed all outliers using Grubbs' test under the assumption of normally distributed data. For all remaining

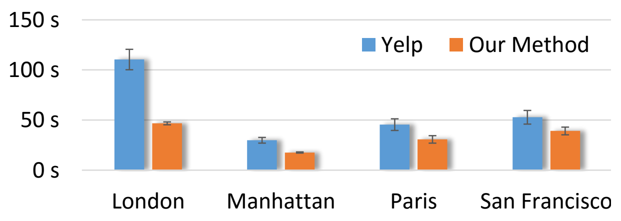


Fig. 11: Average times and standard errors of the measurements from our user study in which we asked the participants to find specific POIs given an exploration task (cf. Table 1). Our method allows the user to find a suitable POI in significantly less time than with Yelp. Especially in environments with many POIs like in our London example we observed a time reduction of almost 60%.

samples we computed the average values and standard errors. As can be seen in Figure 11, using our method the participants needed significantly less time to find a POI that matched the description in the exploration task. Especially for complex environments with many POIs we observed a time reduction of almost 60%. The long average time in the London example can be explained by the choice of the requested POI, which was harder to find than the POIs in the other examples due to the total number of POIs in the environment and its location in a rather late section of the path.

Questionnaire. In addition to the exploration tasks, the users were asked to assign levels of agreements to several statements about the used methods. The questionnaire consisted of 13 statements and 6 open questions. In Figure 12, some of the statements with corresponding answers are shown. Please refer to the supplementary materials for the full evaluation of the questionnaire. As can be clearly seen in Figure 12, 83% of the participants agree or strongly agree that our method is easier to use than Yelp and that it would be their preferred method if they would have to explore a path in an unfamiliar region. Moreover, 92% of the participants agree that our method allows to gain quick insight into the POIs in the environment, while only 17% had this opinion about Yelp.

The open questions were used to clarify what the users most liked or disliked in both presented methods. Many users stated that they liked the freedom provided by Yelp but that they found it very tedious to use it for exploration due to the intense level of interaction. In contrast, our method was perceived as very user-friendly and people liked that they could quickly get a feeling about the POIs in the environment. Two aspects they did not like about our method is the fixed zoom level and the inability to freely move the viewport for investigation of regions which are more distant to the path. This leads to exciting possibilities of future work to increase the adaptiveness of our system.

10 Results and Discussion

Figure 16 shows examples of our approach for several cities using different mobile output devices. Table 2 shows the impact of the complexity of some examples (number of variables and constraints) on the run-time of the BIP.

In order to evaluate the impact of the number of parts in the incremental approach, we measured the runtime of the BIPs for detail lens placement for one particular example (San Francisco, iPad Air 2) in which the number of parts was gradually increased. (cf. Figure 13). With 4 parts we could decrease the overall runtime by 76%, with 8 parts by almost 90%, and we could show in a user study (see Section 9) that our method then still provides acceptable

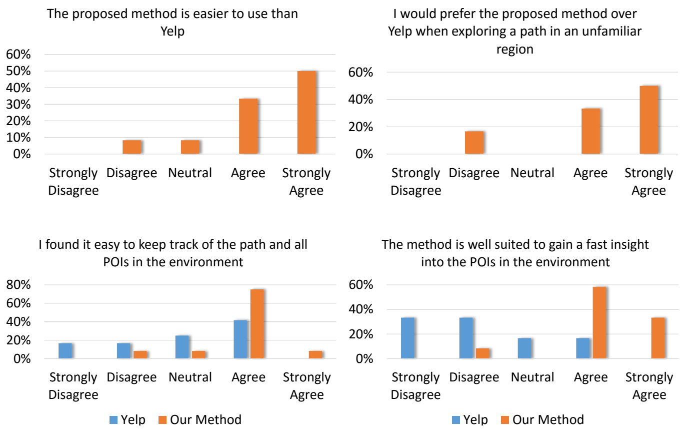


Fig. 12: The participants were asked to assign levels of agreement to several statements. Above some of these statements plus answers are shown. The answers show the significant acceptance rate of our method: 83% agree or strongly agree both that our method is easier to use than Yelp and that it is the preferred method for exploring a path in an unfamiliar region.

solutions. Figure 14 shows the impact of the number of POIs, Figure 15 the impact of the path length on the runtime.

Limitations. Currently, our approach is directed to one specific zoom level, which we consider as the main limitation. Users can still zoom in and out, but this leads to a growing/shrinking of detail lenses and possible visible artifacts like POI pins which get occluded by detail lenses.

Another limitation of our approach lies in the complexity of the binary integer program for the path computation (Section 4) for very big problem instances. Although we can find an optimal path for problems with a similar range to the presented results in a few seconds, we observed a significant performance drop for considerably larger distances (e.g. over 10 km). However, we do not expect any usability issues since our framework is specifically designed for pedestrians and reasonable walking distances.

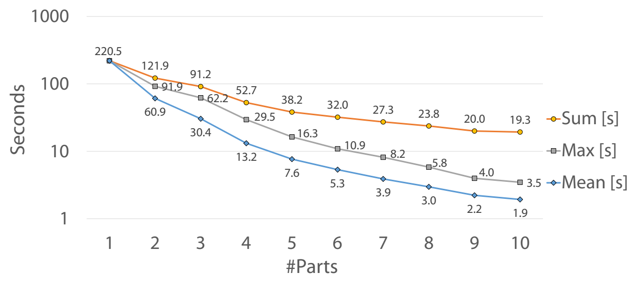


Fig. 13: Runtime to solve the BIPs using the incremental approach (cf. Section 6.7) depending on the number of parts the path is segmented into. The shown numbers refer to an example with 62 POIs and 80 frames and an iPad Air 2 as the chosen output device.

11 Conclusion

We have presented a system for dynamic route exploration on mobile devices. Our results give immediate guidance to a user by presenting both routing information for an attractive path whose environment contains many high-quality points of interest (POIs), as well as information about those POIs by the use of detail lenses.

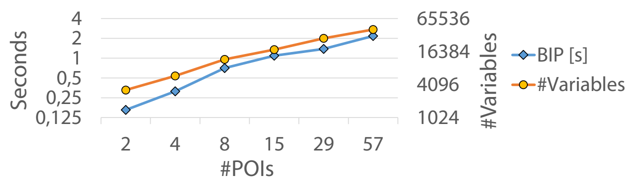


Fig. 14: Impact of the number of POIs on the resulting run-time and the number of variables of the BIP. The numbers refer to the San Francisco example (80 frames) by taking only every 2^n-th queried POI of P for $5 \geq n \geq 0$ along the longitudinal direction.

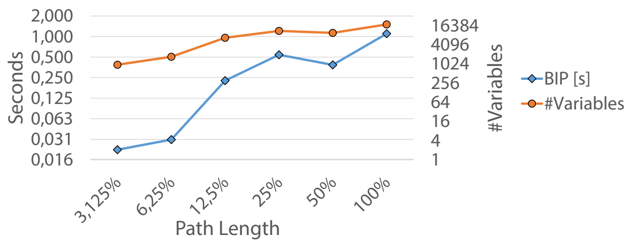


Fig. 15: Impact of the path length on the resulting run-time and the number of variables of the BIP. The numbers refer to the San Francisco example with 15 equally distributed POIs along the middle $100/2^n$% of the original path for $5 \geq n \geq 0$.

In addition, we contribute an algorithm for optimizing the map cutouts which correspond to the viewports in the final visualization that is presented to the user. Moreover, we introduce a novel layout algorithm that computes a dynamic layout of rectangular entities in a discrete grid under specific constraints and present the results of a study that shows the significant acceptance rate of our method.

For future work, we want to extend the functionality of or framework. We think of a multi-layer approach in order to allow a visualization on different zoom levels and smooth transitions between the layers.

Acknowledgments

We thank the reviewers for their helpful suggestions, Michael Hecher for valuable comments on the user study, Christian Hafner for discussions and proofreading, and all participants of the user studies for their time and patience. This work was funded by the FWF, project numbers P23237-N23 and P27972-N31.

References

[1] J. R. Akerman, *Cartographies of travel and navigation*. University of Chicago Press, 2006.
[2] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.
[3] X. Lu, C. Wang, J.-M. Yang, Y. Pang, and L. Zhang, “Photo2Trip: generating travel routes from geo-tagged photos for trip planning,” *Mm*, pp. 143–152, 2010.
[4] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura, “Travel route recommendation using geotags in photo sharing sites,” *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 579–588, 2010.
[5] ——, “Travel route recommendation using geotagged photos,” *Knowledge and Information Systems*, vol. 37, no. 1, pp. 37–60, 2013.
[6] A. Kachkaev and J. Wood, “Automated planning of leisure walks based on crowd-sourced photographic content.” *Paper presented at the 46th Annual Universities’ Transport Study Group Conference, 06-01-2014 - 08-01-2014, Newcastle, UK.*, no. January, pp. 227–246, 2014.
[7] F. Grabler, M. Agrawala, R. W. Sumner, and M. Pauly, “Automatic generation of tourist maps,” *ACM Transactions on Graphics*, vol. 27, no. 3, p. 1, aug 2008.

	iPhone 6 Plus					iPad Air 2				
Example	**#POIs**	**#Frames**	**#Variables**	**#Constr.**	**Time [s]**	**#POIs**	**#Frames**	**#Variables**	**#Constr.**	**Time [s]**
London	40	56	20442	1311	0.582	64	55	302499	8492	84.923
New York	43	67	34544	2511	1.776	74	79	360866	12229	170.401
Paris	20	98	24797	3663	0.909	41	160	297518	24087	107.351
San Francisco	58	78	32740	2878	0.843	62	80	362088	14102	220.539
Vienna	59	62	36145	1946	0.948	83	61	395288	8882	152.553

TABLE 2: Quantitative results for five different cities and two differently sized mobile devices. All numbers refer to a complete (non-incremental) solution. Note that the number of variables involved in the BIP for optimized detail-lens placement is an order of magnitude bigger for the iPad than for the iPhone, while the corresponding time to solve the BIP differs in 2 orders of magnitude.

[8] P. Mooney and A. Winstanley, "An evolutionary algorithm for multicriteria path optimization problems," *International Journal of Geographical Information Science*, vol. 20, no. 4, pp. 401–423, 2006.
[9] E. Imhof, "Positioning names on maps," *The American Cartographer*, vol. 2, no. 2, pp. 128–144, 1975.
[10] S. A. Hirsch, "An algorithm for automatic name placement around point data," *The American Cartographer*, vol. 9, no. 1, pp. 5–17, 1982.
[11] S. Edmondson, J. Christensen, J. Marks, and S. Shieber, "A General Cartographic Labelling Algorithm," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 33, no. 4, pp. 13–24, dec 1996.
[12] J. Christensen, J. Marks, and S. Shieber, "An empirical study of algorithms for point-feature label placement," *ACM Transactions on Graphics*, vol. 14, no. 3, pp. 203–232, jul 1995.
[13] S. Zoraster, "Integer programming applied to the map label placement problem," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 23, no. 3, pp. 16–27, 1986.
[14] ——, "The solution of large 0–1 integer programming problems encountered in automated cartography," *Operations Research*, vol. 38, no. 5, pp. 752–759, 1990.
[15] M. Birsak, P. Musialski, P. Wonka, and M. Wimmer, "Automatic generation of tourist brochures," *Computer Graphics Forum*, vol. 33, no. 2, pp. 449–458, 2014.
[16] M. Agrawala and C. Stolte, "Rendering effective route maps," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*. New York, USA: ACM Press, aug 2001, pp. 241–249.
[17] J. Kopf, M. Agrawala, D. Bargeron, D. Salesin, and M. Cohen, "Automatic generation of destination maps," *ACM Transactions on Graphics*, vol. 29, no. 6, p. 1, 2010.
[18] P. Karnick, D. Cline, S. Jeschke, A. Razdan, and P. Wonka, "Route visualization using detail lenses." *IEEE transactions on visualization and computer graphics*, vol. 16, no. 2, pp. 235–47, jan 2010.
[19] Y.-T. Zheng, S. Yan, Z.-J. Zha, Y. Li, X. Zhou, T.-S. Chua, and R. Jain, "GPSView," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1, pp. 1–18, 2013.
[20] K. Been, E. Daiches, and C. Yap, "Dynamic map labeling." *IEEE transactions on visualization and computer graphics*, vol. 12, no. 5, pp. 773–80, jan 2006.
[21] T. Götzelmann, K. Hartmann, and T. Strothotte, "Agent-Based Annotation of Interactive 3D Visualizations," in *6th International Symposium on Smart Graphics*. Springer Verlag, 2006, pp. 24–35.
[22] J.-D. Fekete and C. Plaisant, "Excentric labeling," in *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*. New York, USA: ACM Press, 1999, pp. 512–519.
[23] L. Chittaro, "Visualizing information on mobile devices," *Computer*, vol. 39, no. 3, pp. 40–45, 2006.
[24] M. A. Rylov and A. W. Reimer, "Improving label placement quality by considering basemap detail with a raster-based approach," *GeoInformatica*, vol. 19, no. 3, pp. 463–486, jul 2014.
[25] F. Wang, Y. Li, D. Sakamoto, and T. Igarashi, "Hierarchical route maps for efficient navigation," in *Proceedings of the 19th international conference on Intelligent User Interfaces - IUI '14*. New York, New York, USA: ACM Press, feb 2014, pp. 169–178.
[26] A. J. Orman and H. P. Williams, *Optimisation, Econometric and Financial Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ch. A Survey o, pp. 91–104.
[27] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff, "Boundary labeling: Models and efficient algorithms for rectangular maps," *Computational Geometry*, vol. 36, no. 3, pp. 215–236, apr 2007.

Michael Birsak is a doctoral researcher at the Institute of Computer Graphics and Algorithms of the Vienna University of Technology. He received an MSc degree in computer science from the Vienna University of Technology in 2012. His current research interests include Geospatial Visualization, Discrete Optimization and Fabrication.

Przemyslaw Musialski is with TU Wien where he is senior research associate at the Institute of Discrete Mathematics and Geometry, and member of the Center for Geometry and Computational Design (GCD) where he heads the Computational Fabrication group. He obtained the MSc degree in 2007 from the Bauhaus University Weimar, Germany, and the PhD degree in 2010 from the TU Wien, Austria. From 2011 till 2012 he was postdoc at the Arizona State University, AZ, USA.

Peter Wonka received an MS degree in urban planning and the doctorate in computer science from the Vienna University of Technology. He is currently with King Abdullah University of Science and Technology (KAUST). Prior to coming to KAUST, he was an Assoc.Prof. at the Arizona State University. His research interests include various topics in Computer Graphics, Visualization, and Image Processing. He is a member of the IEEE.

Michael Wimmer is an associate professor at the Institute of Computer Graphics and Algorithms of the Vienna University of Technology, where he received an M.Sc. in 1997 and a Ph.D. in 2001. His current research interests are real-time rendering, point-based rendering and procedural modeling. He has coauthored many papers in these fields, and was papers co-chair of EGSR 2008 and Pacific Graphics 2012, and is associate editor of Computers & Graphics.

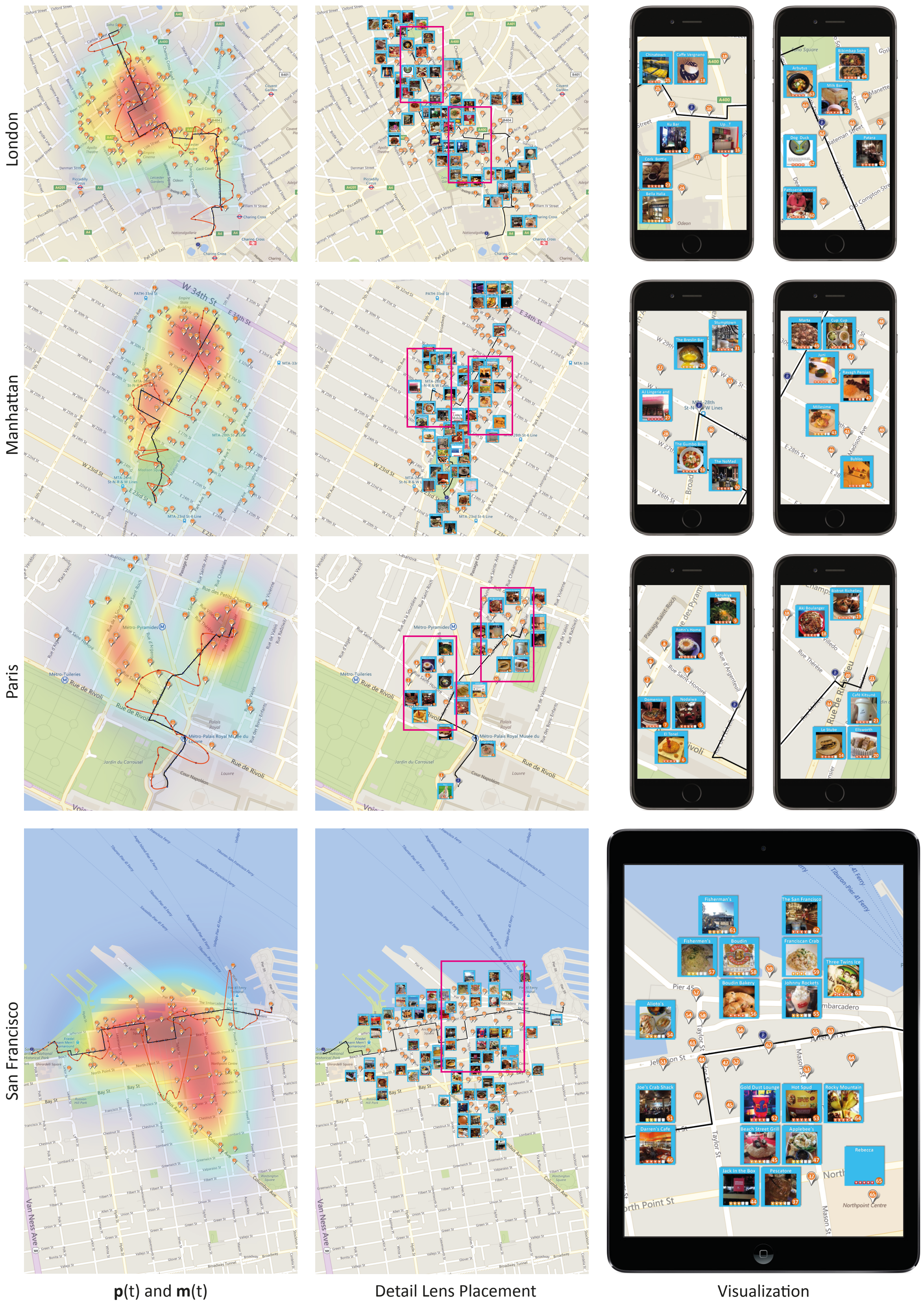


Fig. 16: Results of our framework. (left) Overview of the example areas with the queried set of POIs P and the computed paths $\mathbf{p}(t)$ (black) and $\mathbf{m}(t)$ (dark orange). The sum of the mixture of POI Gaussians, indicating the POI distribution, is outlined by a heat-map. (middle) The computed positions of the detail lenses. Note that the overlaps do not occur in the final visualization due to placement in different time spans. (right) Example views of the final visualization for different mobile devices (iPhone 6 Plus and iPad Air 2).