Modifying Route Map Visualizations For Walking Smartphone Use

Louise Oram

Computer Science, University of British Columbia

ABSTRACT

This project explores visualizations for route maps that are displayed on a smartphone. Route finding and following is a multitasking situation that would benefit from better readability of the map, as it would decrease the cognitive effort required for the user to synthesize the information they need to follow the route. This can be done through various information visualization techniques, such as focus + context and highlighting. An initial implementation of such visualizations is created for an Android smartphone, and its strengths and weaknesses are discussed. Finally, conclusions are drawn about how the route map visualization could be improved and what areas of future research would be beneficial.

KEYWORDS: Maps, route finding, visualization, smartphone.

1 INTRODUCTION

Finding your way from one place to another used to be accomplished by asking for directions or using a printed map. With the advent of the internet, maps became available online and algorithms were used to find routes, which then required printing or sketching out. Smartphones now allow one to carry these capabilities around in the pocket and therefore route-find on the fly.

For the purpose of this paper, readability will be defined as the ability for a map to be easily understood when looked at (for instance, being able to easily find where you are currently on the map). Maps are often used while walking, so the basic problem of readability has been around a long time. However, now that route maps can be created we have the potential to increase the readability in a way that is specific to the route being taken.

Smartphones add a layer of problems as well as introduces some advantages. A major drawback is the small screen, since when zoomed in to a readable level some or most of the route may be off the screen. This map must then be interacted with through touch gestures such as dragging to pan, and pinching to zoom. An advantage is that smartphones have position capabilities, allowing the map to indicate where the user currently is (with a varying level of error depending on the signal). Additionally, they can orient the map to the surroundings as the user changes the orientation of themselves and the phone.

This paper focuses on the problem of increasing the readability of route maps on smartphones. Data from Google maps and its route finding capabilities were used. The Android operating system was used as the mobile development platform, and a Nexus One smartphone borrowed from the Human Computer Interaction group at the University of British Columbia was used.

1.1 Related Work

As smartphones become more prevalent, more research is being conducted on interaction with these devices. One avenue of research is their use while walking, as would be seen when navigating a route. Schildbach and Rukzio [8] saw that walking slowed when reading text on a smartphone, regardless of the font size. Three font sizes were tested, and the increase in scrolling needed when larger font was used is thought to mediate the effect of the text being easier to read. There was also a trend towards users stopping more when reading the small font size. Additionally, a significant effect was found for text size on subjective workload [8]. This supports the idea that increasing the readability of maps will decrease workload.

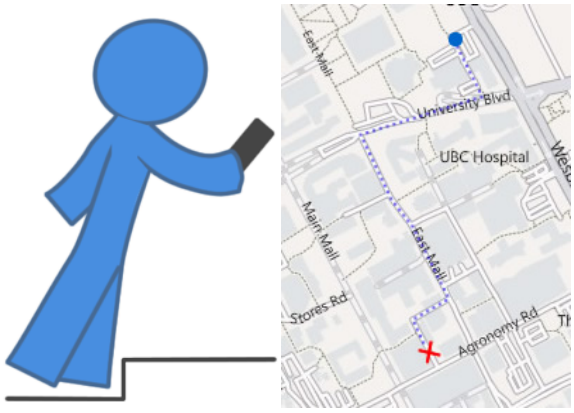

Figure 1. How can we increase the readability of route maps to be viewed on a smartphone while walking?

To mitigate problems with locations being off the screen, Baudisch and Rosenholtz developed a technique called Halo [4]. This technique virtually extends the off-screen locations by surrounding them with a circle large enough to show near the edge of the current frame of the screen. This technique was compared to arrows with the distance of the location displayed within them, and the Halo technique was found to be faster for completing tasks such as finding the closest restaurant among 5 restaurants [4]. This visualization technique would likely help users of route maps understand where the destination is in relation to where they currently are on the route.

Maps are easier to follow when oriented to the surroundings. Additionally, an egocentric frame of reference is generally assumed to be more natural, as people tend to describe a scene from their viewpoint [9]. Creating a route finding application that shows the user where they are on the map and lines the map up with the environment will reinforce this frame of reference as it enables them to imagine themselves as a location on the map.

Agrawala and Stolte [1] created an application that used cartographic generalization techniques to render route maps. They found that almost all users would use their application again, although some wanted to use it in conjunction with standard driving directions. In a later study Kopf et al. [7] created software that runs with Bing maps, and creates a destination map. To facilitate finding the destination, the area surrounding it is enlarged and the detail is kept, whereas the areas leading to it are simplified and shrunk. Both these examples use non-uniform scale to ensure that the information needed for each stage of the route is

olouise@cs.ubc.ca

available. However, these examples applications are more suited for driving, or generally traveling longer distances that are easier to simplify. When walking, the simplification of the map requires different details, as there are more visual cues to use and less reliance on roads and road names.

1.2 Motivation

Walking while looking at information on a smartphone increases the overall cognitive load, as there are many distractions and obstacles (figure 1). Therefore a route map that can be designed in a way that decreases mental effort on the part of the user will be beneficial. As seen in the above-related work, this is often accomplished with generalization. A simplification of the map in figure 1 is seen in figure 2, as an example of generalization.

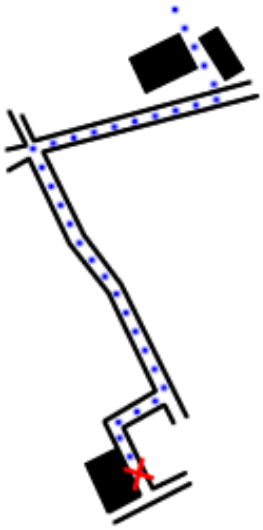

Figure 2. Example of route map generalization showing the suggested route (blue dots), the destination (red X), and relevant details such as roads and buildings.

In order to reduce cognitive load when looking at the route map, we need to make it easier to get salient information to users' eyes when they glance at the screen. Frequent users of maps, such as orienteers, develop heuristics for route planning and seeing the information needed for the current navigation task [5]. How can route maps automate this and create a visualization for the user?

The rest of the paper is organized as follows: Section 2 describes the proposed Android-based application, its implementation, and scenarios of use. It also contains screenshots of the application, and explains its functions. Section 3 discusses what was learned from the process as well as the strengths and weaknesses of the project. Section 4 concludes by outlining ideas for future work.

2 SOLUTION

Maps generally employ visual layering, colour and shapes in their legends. However, when adding a route to a map, there are additional ways to aid visualization. Routes involve one or more decision points along the way where the user must use the map to interpret where to go. As seen in the related work section, generalization and distortion visualizations are often used to increase the readability of route maps. This implementation chose to use focus + context to get more salient information about decision points along the route across to the user. It was decided that integrating the focus + context into one view was a better use of screen real estate than dividing the screen into dedicated focus and context sections. The areas of focus used higher scale maps to offer more detail and were highlighted by fading the background map. The halo off-screen visualization [4] was used to help the user understand the direction and distance of the destination from their current location. In this application, a circle centred at the destination location with a radius such that the edge of the circle touched the current location was drawn. Since the view when following the route is egocentric, the arc will always go through the middle of the screen. This is different than the implementation in [4], which had several arcs placed very close to edge of the screen. Since only one arc was used in our implementation, it is better to have the arc go through the centre of the screen to maximize the intuition that the arc provides about direction and distance to the destination.

2.1 Implementation

The outcome of this project is an Android application that can be installed on any Android phone. However, to develop or compile the code on a computer, one must install the android software development kit (SDK) and download the essential components for the development environment [3]. Eclipse is the recommended integrated development environment (IDE), as there is a plugin available to integrate with the SDK.

Initially Bing maps was considered for the application, since it appeared to have better data for smaller roads useful for walking routes. However, upon exploration very little documentation or examples were available for Bing maps, as compared to Google maps. For this reason I chose to use Google maps as I could still work on the visualizations, but have an easier time finding documentation to understand how play with the map.

To create a basic Android application, a main 'Activity' must be defined (figure 3). This class sets up any 'Views' and has access to the phone's location information. The map view class has functionality to display the Google map as well as detecting touch events on the map. The map view uses a number of overlay classes to draw on top of the map.

To load maps in the Android application, a map view is created which uses an application programming interface (API) key (stored in the main.xml file) for Google maps access. The map view must be programmed to handle touch events within it, as desired (see code in MyMapView).

The main development hurdle in this project is the lack of an API for function calls to access Google route information. Route data was downloaded from a Google website by specifying locations in URL parameters. The start location coordinates was acquired from the phone's current location and the destination was set when the user double tapped the screen. The pixel coordinate of the tap location was translated to the latitude and longitude coordinates, based on the current state of the map view. These coordinates were then encoded into a URL string, which was used to get a JSON file off Google's route finding server. This JSON file was then parsed. For this project I extracted both the full route, as well as the coordinates of each 'leg' of the route. The starting points of these legs seem to line up with decision points, such as turning points or major intersections. When two or more decision points were located close to one another, they were merged into a single point.

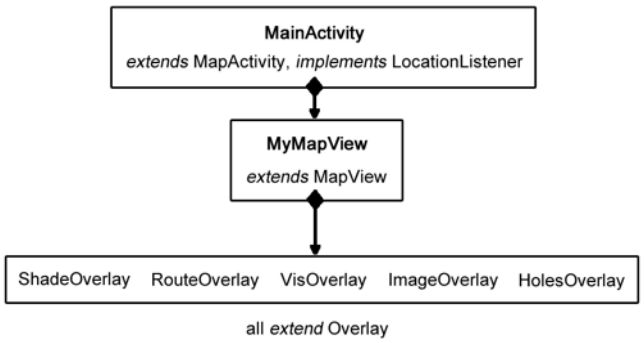

Figure 3. Outline of the code.

The basic map cannot be modified; the only way to modify the map for android is by using overlays, which are drawn on top of the map. Each of these overlays needs its own class, since for different types of overlays the draw function must be overridden to create the desired graphics. When an overlay is instantiated, the draw function is called and updates as the map is zoomed in and out. The types of overlays used in the application range from a simple semi-transparent shading used to fade the map (ShadeOverlay), to an more complicated overlay that draws static map images on top of each of the decision points (ImageOverlay).

The route is passed into the route overlay to draw the route as a dark grey line (RouteOverlay). The coordinates of the decision points are passed into ImageOverlay as well as HolesOverlay. The image overlay uses static map images, which are procured by creating a series of URL's and saving the returned bitmaps. The class VisOverlay draws a red X at the destination and creates the red halo circle that will always extend to the current location.

HolesOverlay is an attempt at an alternate visualization, with holes cut in a semi-transparent overlay around the decision points. While it did achieve the goal of highlighting the decision points, it did not achieve the desired distortion, as the whole map was still at one scale.

The calls to the overlays are found in the function onInterceptTouchEvent, in the section that detects double taps. Therefore, to see what happens with different overlays or different orderings of the overlays you must comment in/out or re-order the calls to the overlays there.

Lastly a button to start or stop the application following the current location was added. The code for this is found in the main activity, as the button is actually a separate view. It did not need a separate class since it is not as complicated as MapView, and I did not need to extend its functionality.

Many standard java libraries were used, as can be seen by the lengthy import lists at the top of some of the files. Overall, some of the functions needed already existed, but were overridden in order to do as was desired for this application. The main design hurdles for this application was determining what overlays would work well, which are discussed in the next section.

2.2 Application

Potential use case:

A user has just gotten off the bus and wishes to find the quickest route to the community centre from there. She takes out her smartphone and opens up the application. Her current location is displayed, and she pinches to zoom in to the surrounding area. She pans around looking for the community centre and double taps it. Once the route is drawn, she clicks the 'walk' button and the application centres around her current location at an appropriate zoom level. She then walks to the community centre, and glances down at the application intermittently to see when her location dot approaches a turning point.

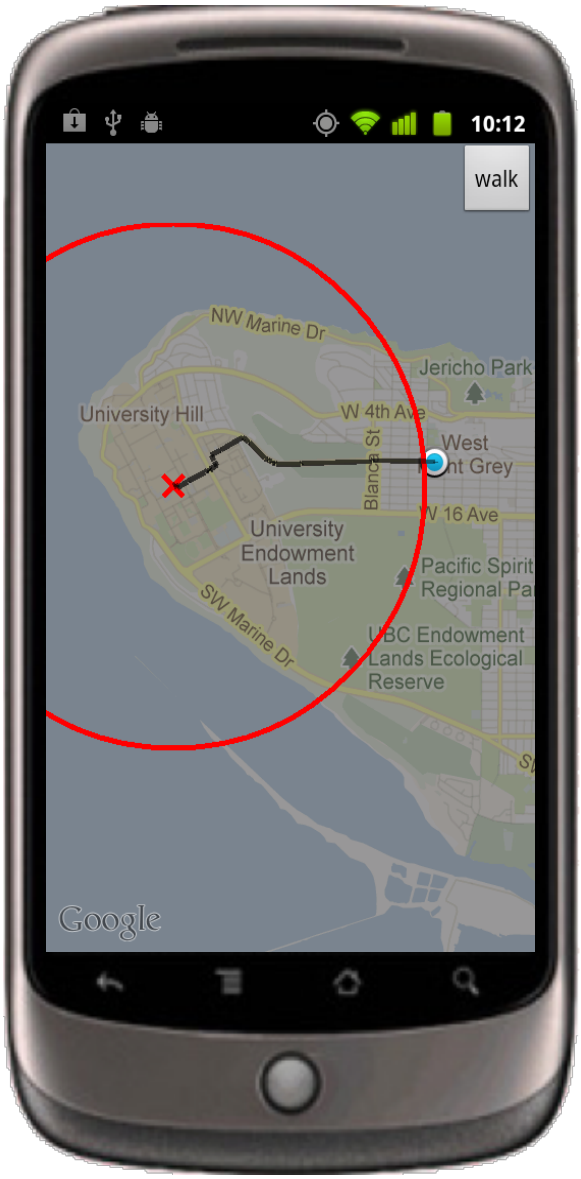

Figure 4. A zoomed out view of the route.

The application starts as a basic map with the current location displayed. The user can then drag to pan around on the map, and pinch to zoom in and out of the map. Once the destination has been decided, the user double taps it and the device retrieves the information for the route. If the user is zoomed out she will see the plain route (figure 4), but when zoomed in further, overlays of high zoom maps or air-photos will be displayed under the route at the decision points (figure 5). The button in the upper right corner allows the user to enter walk mode, where the map continually centres on the current location as the device receives location updates (figure 6). Clicking that button again (it will change from 'walk' to 'stop' after being clicked the first time) will exit that mode so the user can pan around the map if she wishes.

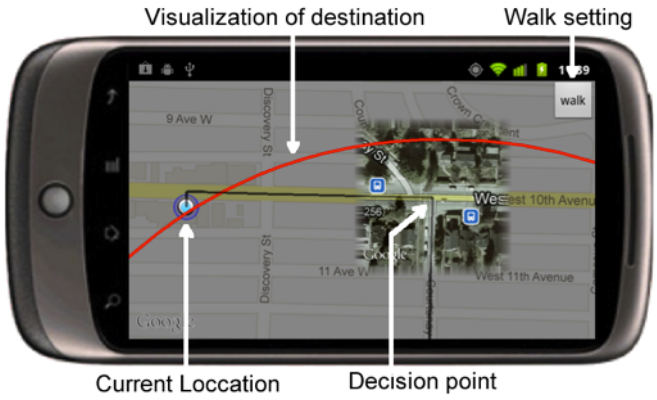

Figure 5. Many of the functions of the application are seen in this image, such as an air-photo at the decision point, and the red arc indicating the direction of the destination.

Figure 6. In this example the user is looking at the detail at the destination, but if they then click the walk button the application will pan to the current location.

3 Discussion

Many different types of overlays were experimented with for this project, and a lot of tweaking (such as setting the size and scale of decision point images) was necessary to find a good balance.

I tried using different static maps for each zoom level, but retrieving so many maps significantly decreased the overall speed of the application, so this was deemed unfeasible. Instead, a single set of static images were used and a compromise for the scale and size of the single set of bitmaps had to be decided on. Unfortunately these maps work best for certain scales, so a default zoom level was set for the walking mode.

A limitation of the smartphone used is that it would not accept my SIM card, so I could only access the location capabilities if connected to the internet. This would have been a significant limitation if I were to do further testing, particularly since the Wifi at UBC does not seem to offer very good triangulations for the location updates of the device.

3.1 Strengths and weaknesses

An overall strength of creating an Android application is that it works on a widespread mobile platform. Additionally it integrates the location data with the map, and can follow the location when requested. This allows the user to easily find where they are, and the map can orient to the surroundings making it easier to understand the map in relation to the environment.

The information visualization aspects are outlined below:

Strengths:

- *Readability:* draws the route in a highly visible manner.
- *Focus + Context:* is a first attempt at creating an effective focus + context visualization for route maps.
- *Small screen:* effective use of off-screen visualization.

Weaknesses:

- *Distortion:* too much distortion due to using overlays of static maps (figure 7).
- *Accuracy:* unsure of what information is used to navigate with, and may vary from user to user.

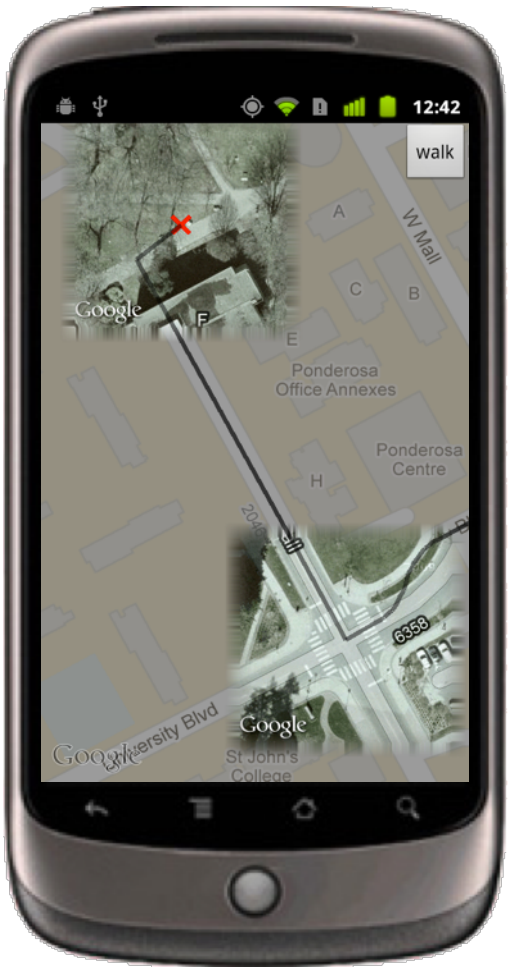

Figure 7. Problems arise when overlaying static map images on the background Google map, as the edges do not line up well for all zoom levels.

Figure 8. The previously mentioned holes overlay does not have any distortion, but does not offer any additional information about the decision points.

The outlined strengths are small, but are an exploration of how this type of application would be set up. The weaknesses are large and problematic, and the distortion problem would be hard to fix without the ability to distort the base map, which is not possible with only overlays. Accuracy could likely be fixed with empirical studies to understand what people look for in the environment when they are using a route map. This would create weightings for each map feature, however it is possible these weightings would not work for all users. For a route map to be created optimally for a specific user you would need to know their personal weightings, so as to display the features most useful to them. To determine which objects (such as buildings, streets, or landmarks) should be shown on the route map, an Importance metric (I) could be defined. This value could be computed as a weighted sum of various factors (such as building or street size) that determine the importance of the object, as seen in Equation 1:

$$I = \frac{\sum_{j=1}^{N} w_j F_j}{d} \qquad (1)$$

where the variables are defined as:

I = Importance of a certain object
w = Weighting of a factor
F = Factor associated with the object
d = Distance of object from the decision point
N = Number of factors considered

Factors such as the size of the object, distance from other objects, or the generic importance of the object could be considered. One could then either choose a threshold for what to include based on the value of I, or start by adding the object of highest importance until a certain accumulated value of I has been reached. Acquiring and displaying these objects would, however, require access to the vector elements of the map.

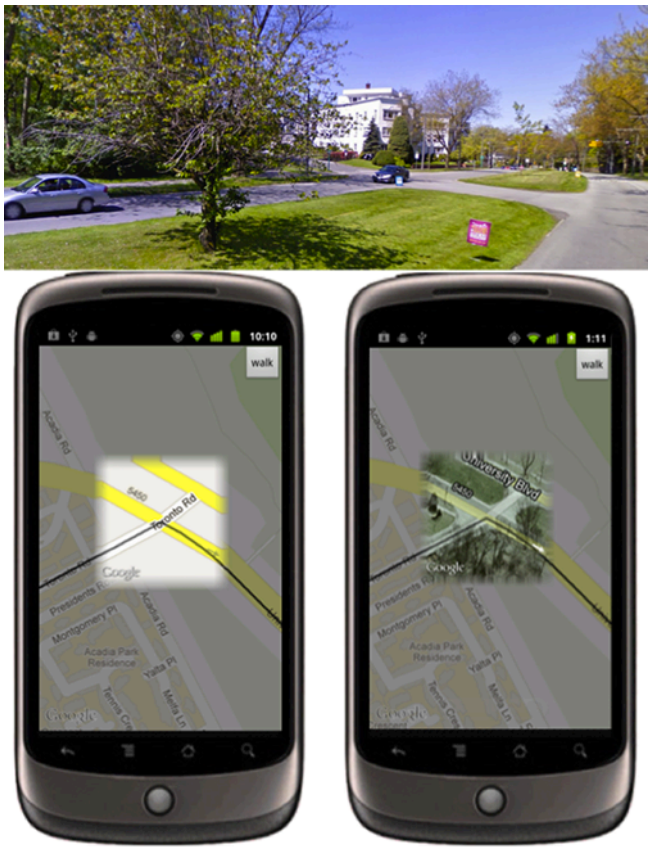

Figure 9. For this intersection what type of information would be most useful, a simple map, air-photo, or something else?

If there were more time I would have liked to get more feedback about the visualizations, and not just informal feedback from a few friends, to make sure my personal ideas about maps work well for other people. Designer bias is obviously a major concern for this project, particularly because of my experience with using and creating maps. For instance there are several types of static maps available for overlays, and feedback on what is intuitive to users would be beneficial. It is possible that they are unfamiliar with using air-photos for navigation, even if it hypothetically offers more information (such as the location of forested areas in figure 9).

The usefulness of different map features for route finding would need to be tested in order to determine the weighting of each feature (Equation 1, w). For feasibility this could be done with a logging study, where users record what they use to make decisions when navigating with a smartphone (or even speak aloud and have the phone record it while they are performing the task).

Once these visualizations are more refined I would need to test if they are more readable than a traditional route map. This could be done by monitoring the time the user spends looking at the screen of the smartphone, as well as the time taken to

complete the task. Less time spent looking at the screen would be a reasonably direct measure that they are able to get the visual information from the map more easily. If they only took less time to complete the task using the visualization it would suggest indirectly that they were able to relate the map to the surroundings faster, likely because it is more readable while walking.

4 FURTHER WORK AND CONCLUSIONS

In retrospect, it is possible that a website tailored for loading on a smartphone would have been a better choice for implementing this project. It appears that websites can now ask the user if they will allow access to the location information from the device, and most importantly there is an API for the vector layers for JavaScript [6]. It is easier to modify the map when vector layers are available since the data in the map can be altered. Also, this might have been faster to develop initially, and left more time for exploring visualization techniques and ideas.

The basic idea behind this visualization project is that it will be easy for the user to see when their location dot enters the area of a highlighted decision point, and that this will cue them to pay more attention to the map and the surroundings so as to follow the route accurately. Smartphones usually have the ability to produce vibrotactile feedback. An interesting idea for future work would be to give some a tactile signal when the location dot approaches a decision point, so that the user knows when to glance at the screen.

Overall this is a first pass at implementing an information visualization solution for route maps on a smartphone. I went from no familiarity with the Android development platform, to being relatively comfortable, and interacted with a smartphone more than I ever have before. Despite the restrictions the development environment created for the visualization options, I successfully explored some of the design space for the route map visualizations.

REFERENCES

[1] Agrawala, M., & Stolte, C. (2001). Rendering Effective Route Maps: Improving Usability Through Generalization. In Proc. Siggraph.

[2] Android. (2011, December 9). Package index. Retrieved from http://developer.android.com/reference/packages.html

[3] Android. (n.d.). Installing the SDK. Retrieved from http://developer.android.com/sdk/installing.html

[4] Baudisch, P., Rosenholtz, R. (2003). *Halo: A Technique for Visualizing Off-Screen Locations*. In Proceedings of CHI 2003, 481-488.

[5] Eccles, Walsh & Ingledew. (2002). The user of heuristics during route planning by expert and novice orienteers. Journal of Sport Sciences, vol 20: issue 4, p327-337.

[6] Google Code (n.d.). Google Maps JavaScript API V3. Retrieved from http://code.google.com/apis/maps/documentation/javascript

[7] Kopf, J., Agrawala, M., Bargeron, D., Salesin, D., Cohen, M. (2010). *Automatic Generation of Destination Maps*. SIGGRAPH Asia, 158:1-158:12.

[8] Schildbach, Rukzio. (2010). Investigating selection and reading performance on a mobile phone while walking. In Proc. MobileHCI.

[9] Tversky, B., Hard, B. (2009). Embodied and disembodied cognition: Spatial perspective-taking. Cognition, 110:1, 124-129.