ICAT 2005
December 5-8, Christchurch, New Zealand

A Constrained Road-Based VR Navigation Technique for Travelling in 3D City Models

Timo Ropinski, Frank Steinicke, Klaus Hinrichs
Institut für Informatik, WWU Münster, Einsteinstraße 62, 48149 Münster, Germany
{*ropinski,fsteini,khh*}*@math.uni-muenster.de*

Abstract

In this paper we propose a novel navigation metaphor for the exploration of 3D city models in virtual environments. The presented metaphor supports intuitive navigation without disorientation through 3D city models in a manner similar to travelling in the real world. Based on a graph representation of the road network of a 3D city model camera paths are calculated and used to enable smooth camera motion. We will explain how smooth camera motions are computed and describe a user interface usable in both desktop-based as well as projection-based environments to specify travel destinations.

Keywords: Virtual Reality, City Models, 3D Navigation, Travelling Metaphor

1. Introduction

Virtual 3D city models provide an important tool for visualizing and communicating spatial information associated with urban environments. They form the basis for many visualization applications in the areas of city-planning, scientific simulations as well as tourism. Although the graphical representation of today's 3D city models tends to be very realistic, especially when combined with stereoscopic display technologies in virtual reality (VR) environments, the navigation metaphors for exploring these city models need further improvements to allow an intuitive navigation without disorientation. In particular, current camera and viewpoint motion control techniques are reviewed in this paper and a new technique is introduced. This technique has been developed with the goal to allow natural navigation through 3D city models.

In VR the term *travelling* denotes an action performed to get from one location to another [3]. There are three important parameters associated with travelling, namely direction of motion, speed and acceleration. In reality we travel for instance by steering a car while using the accelerator and the brake pedal. As pedestrians we simply walk through the world and turn our head to explore the surroundings. In most VR environments we are not able to perform such a kind of travelling, because of a lack of one-to-one mapping between the user's and the camera's motion. Although a one-to-one mapping would result in a natural navigation, since it does not require any special navigation-related actions, it is not intended in applications where long distances have to be covered and the bounds of the virtual environment (VE) exceed the bounds of the tracked area. Besides these mapping problems, a user navigating through a VE needs to be aware of his current position, his destination as well as the *best* path to reach the destination; the quality of a path and hence the best path may vary with the application domain. Therefore, optimized navigation metaphors are needed which ease travelling through VEs and consider these aspects.

Bowman et al. state, that the most intuitive navigation metaphors adapt real world principles and thus improve usability [2]. For instance, relative navigation as well as tele-transportation are not very intuitive and both often lead to disorientation. A common navigation metaphor for desktop-based VR is the *fly metaphor* which enables the user to define position and orientation as well as speed and acceleration of the virtual camera by using the mouse and/or the keyboard. This multiple DoF navigation metaphor often involves a cognitive overload, which may result in disorientation if the user is not familiar with this kind of navigation, since in reality he is forced to move with 2 DoF on the ground plane and the head can be oriented to have a look-around [8].

To avoid disorientation during navigation the concept of constrained navigation has been proposed. A *constrained navigation is a method that appropriately restricts the user's degrees of freedom* [8]. Often the degrees of freedom are reduced to only 2 DoF, which corresponds to ground-based navigation, where the user can only move with respect to a plane, in most cases the ground plane of a VE. By using constrained camera navigation, the cognitive overload is reduced significantly, and therefore the risk of disorientation, which often results from awkward camera orientations, is minimized. However, solely reducing the DoF during navigation does not necessarily avoid disorientation. In addition it is important to support the user during the navigation task visually by displaying appropriate information, e.g., the current camera position in relation to the entire environment.

In this paper we introduce a constrained navigation metaphor, which permits a very natural exploration of 3D city models. This metaphor has been developed to provide an intuitive interface supporting VR-based exploration of 3D city models and has been preliminarily evaluated in the context of the city of Münster (see Figure 1). We constrain the users degrees of freedom by allowing road-based navigation only. Our road-based navigation metaphor prompts the user to specify a destination to which he wants to travel. After this specification the shortest road-based camera path from the current position to the destination is computed, and the camera is appropriately moved in a smooth manner, i.e., we include animated transitions between successive way points. The proposed navigation metaphor allows a natural navigation by specifying the destination with a single action and by travelling on the road network similar to the real world exploration of urban environments. Thus the spatial comprehension of the 3D city model is improved, which facilitates the generalization process needed to transfer the explored environment to the real world environment. Furthermore, the presented

technique provides a mechanism to give the user feedback about his current position as well as the precomputed camera path at any time during the exploration process.

In the next section we discuss some related work concerning navigation metaphors designed for the exploration of VEs. In Section 3 we explain the calculation of the camera path based on a graph representation of a road network. A user interface which provides an intuitive usage of the presented navigation technique in both desktop-based as well as projection-based VR environments is described in Section 4. In Section 5 we outline some approaches to obtain a graph representation of urban road networks for cases where no such data is available. The paper concludes in Section 6 by summarizing our contribution and discussing some thoughts concerning future work.

2. Related Work

Bowman states that the navigation process within VEs is split into two separate tasks [3]: *travelling* is the actual movement through VEs without considering decision making processes, and *wayfinding* is the cognitive process to define a path through a VE which may be assisted by the underlying system.

Several navigation techniques have been developed to aid both expert as well as novice users when navigating through VEs. In this section we briefly summarize these approaches. But, instead of providing a complete survey of current navigation metaphors, only the most relevant metaphors are reviewed briefly.

In 1990 Ware and Osborne [20] introduced and evaluated their *scene in hand*, *eyeball in hand* and *flying vehicle* navigation metaphors for exploring VEs. All these techniques exploit 6 DoF input devices to manipulate the camera position and orientation in a VE. Ware and Osborne have shown that the *flying vehicle metaphor* is particularly advantageous to explore complex VEs. In 1995 Mine has discussed the *point and fly metaphor* [10], a gesture-based metaphor for navigating in VR. In contrast to the *flying vehicle metaphor*, the user does not steer the flying vehicle itself; instead he points out where to go, and the computer moves the camera to the desired location appropriately. The *Go-Go technique* has been proposed in 1996 by Poupyrev et al. as an extension to the *point and fly metaphor* [12]. This gesture-based navigation metaphor uses a non-linear mapping between gesture and movement and therefore allows acceleration during navigation.

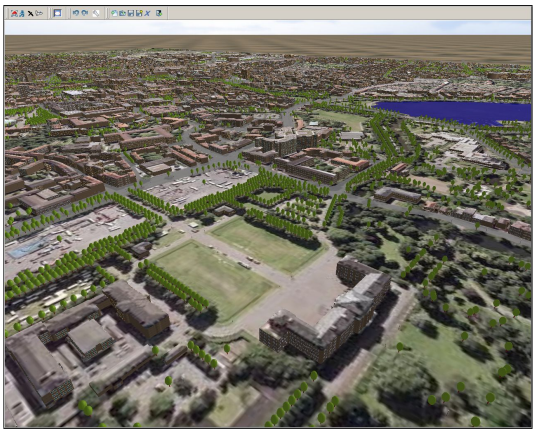


Figure 1: Screenshot of our city visualization application showing the 3D city model of Münster.

The *click and fly metaphor* relieves the user from planning the path to move along; instead it requires the user to specify certain destinations with the mouse and initiates navigation on a system-calculated camera path. This navigation metaphor can be also used in combination with the *world-in-miniature metaphor* (WIM), which offers a second dynamic viewport onto the VE [16]. The WIM metaphor has been originally developed to ease object manipulation in VEs. Pausch et al. have extended this metaphor for navigating through virtual worlds [11]. Their VR-based approach incorporates a technique similar to the *click and fly metaphor* for specifying the destination within a miniature model of the world, which is usually projected onto a small transparent screen held with the user's non-dominant hand. The user moves a miniature model of his view frustum by manipulating a camera icon within the WIM to specify the destination and to orient the camera. In their work Pausch et al. do not propose a solution for how to do the camera path planning to avoid collisions with obstacles during the travelling. Furthermore, they do not provide the user any visual feedback about the current camera path.

Tele-transportation represents an alternative approach for travelling through VEs. When using this metaphor, the user specifies a destination where the camera will be positioned immediately. This technique has the advantage, that no time is needed for travelling which may be favorable in large scale VEs. The obvious disadvantage is that no frame to frame coherence is maintained which makes it difficult to comprehend spatial structures.

Bowman et al. have conducted a user study to evaluate various travelling techniques for immersive VEs [2]. Their results indicate that *pointing metaphors* seem to be advantageous in comparison to *gaze-based metaphors*. Furthermore they have shown that *motion techniques which instantly teleport users to new locations are correlated with increased user disorientation*. Therefore, smooth camera motions between the source and the destination location are necessary to allow intuitive as well as efficient navigation.

For this reason we propose a navigation technique, which combines the WIM metaphor with the *click and fly metaphor* and can be applied in both desktop-based and projection-based VR. Alternatively, we can use a 2D map metaphor, to ensure that the user is aware of his current position. Within projection-based VR environments the miniature model of the virtual world can be controlled by using a *personal interaction panel* which originally has been proposed by Szalavári and Gervautz for the interaction in augmented reality environments [18]. Schmalstieg et al. [15] have presented an extension for VR environments, whereas Stoev et al. use a similar technique for navigation purposes [17]. As part of Section 4 we will briefly describe in Subsection *The Navigation Widget* how we exploit these concepts for our purpose.

In addition to the navigation metaphors reviewed above, the user has to be assisted during exploration of the VE. This is a wide area of research which cannot be covered entirely in this paper. Instead we will list examples of different research directions within this area.

One key feature of intuitive navigation is appropriate camera path planning. In 2003 Salomon et al. have proposed an algorithm which calculates navigation paths in large-scale VEs [14]. Another approach to deal with the complexity of controlling the virtual camera by Drucker and Zeltzer [5] introduces a high-level user interface to ease camera control and to reduce the cognitive overload during navigation. Galyean presents the *river analogy* [7] for a smooth motion along a calculated camera path. With this metaphor the camera moves along the defined path in a manner similar to a boat floating on a river, i.e., *with some latitude and control while also being pushed and pulled by the pre-defined current of the water* [7].

Several approaches constrain the user's freedom during navigation to reduce the cognitive overload, which is usually introduced when using 6 DoF navigation metaphors. Hanson and Wernert define *constrained navigation* as *the restriction of viewpoint generation and rendering parameters to goal-driven subclasses whose access is specified by the application designer* [8]. As mentioned above constraints serve as an important addition to intuitive exploration without overstraining the user who usually travels with only 2 DoF in the real world. Fuhrmann and MacEachren [6] have first postulated the necessity for constrained navigation techniques when exploring geo-virtual environments. Different constraint navigation techniques for geo-virtual environments have been presented by Döllner in 2005 [4]. He proposes restrictions for camera control to eliminate situations where the user gets disoriented because the camera is oriented at awkward directions, e.g. into the sky. In particular, he introduces techniques which force the camera to be positioned between a minimal and a maximal height above the ground, and he constrains the camera to those orientations for which a certain amount of the scene content is seen by the camera.

3. Camera Movement

This section deals with the wayfinding and the travelling behavior chosen for the road-based movement of the virtual camera through a 3D city model. We assume that the road network is given as a planar graph $G = (V, E)$, with the set of nodes V representing the terminal and intersection points of the streets and the set of edges E containing straight segments which represent streets and connect the nodes. One-way streets can be modelled by introducing directed edges into the graph G. For cases, in which a graph representation of the road network is not available, we discuss some strategies on how to extract such a connection graph in Section 5.

Before the calculation of the camera path can be initiated, the user has to specify the travelling destination the camera should move to (see Section 4). The current position of the camera is assumed as the default start position. Neither the current position of the camera nor the travelling destination of the camera has to lie on an edge or a node of the graph G. Based on the travelling destination *end* and the current position of the camera further denoted as *start*, there are different approaches for calculating the camera path. We have decided to use the shortest path from *start* to *end* with respect to the underlying graph G. To determine the shortest path the A* algorithm is a good choice; it exploits topological properties to calculate for an initial node $v_{start} \in V$ the distances to the destination node

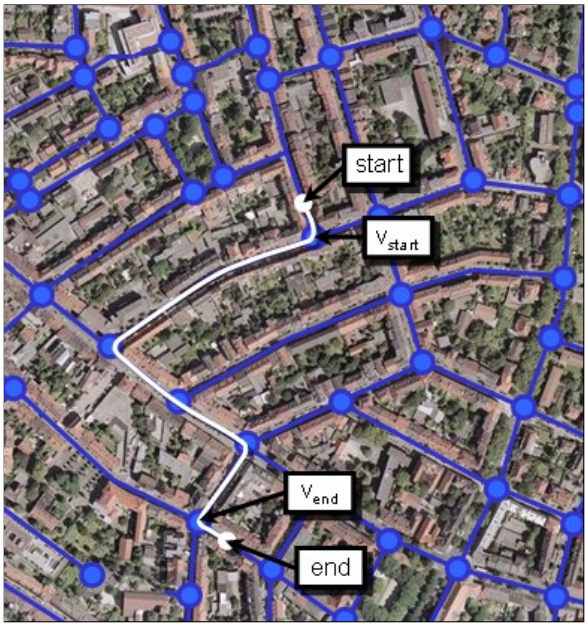


Figure 2: Schematic view of a camera path in the representing road network. The road network is colored blue, while the white shows the calculated camera path.

$v_{end} \in V$. Depending on the size of G it may be favorable to use a more complex algorithm, as the one proposed by Lauther in 2004 [9]. A good although not up-to-date evaluation of shortest path algorithms based on real road networks has been conducted by Zhan et al. in 1998 [21].

Furthermore, as an extension to calculating the shortest path in some cases it can be useful to incorporate additional information for wayfinding. For instance, one could exploit the location of certain landmarks within a tourist information system. Thus it would be possible to include locations of interest into the camera path if the shortest path would run nearby. Another extension could calculate the fastest path by incorporating speed limits or size of streets and first guiding the camera to the nearest main street before proceeding from there. However, in our implementation we use the shortest path form v_{start} to v_{end} as the camera path. To obtain v_{start} and v_{end} one can simply choose the nodes nearest to *start* and *end*. Then the calculated camera path runs from *start* to the location of v_{start} through the shortest path to the location of v_{end} and finally to *end*. This is illustrated in Figure 2 where the road network graph G is represented by the blue lines and dots, and the calculated camera path from *start* to *end* is indicated by the white curve.

Another strategy to get from *start* into the road network resp. from the road network to *end* is to drop a perpendicular from *start* to the nearest $e \in E$ resp. from the nearest $e \in E$ to *end*. This is particularly favorable in cases where for instance v_{start} is located in the opposite direction than the travelling will occur.

An exception, where both strategies are not appropriate are the cases where *start* is closer to *end* than to any node of V. In this case we directly calculate the camera path without considering the graph G.

Based on the calculated camera path, which is a projection to the ground plane, a smooth camera motion can

be performed. As mentioned above, we have chosen the river analogy introduced by Galyean [7]. Instead of giving the user free rein, the river analogy moves the camera along a predefined camera path, which in our case is defined by the road network. Additionally, the river analogy controls the speed of movement by adapting acceleration and deceleration when passing curves. In our application we always orient the camera in a way that the look-at-vector is collinear to the tangent vector of the road network at the current position heading towards the direction of movement. In analogy to driving with a car through the city this ensures that the camera is always aligned with the travelling direction. However, when applying this strategy for orienting the camera, the road network incorporates a problem when smooth camera motions are desired. Since all streets are represented as straight line segments, which meet in the intersections stored in the nodes of the graph G, a turn is needed at each intersection. When using an unmodified camera path derived from the graph G there are two options to perform this turn. The first one is to stop the camera movement at an encountered intersection and to perform a smooth rotation. The second option is to not use a continuous rotation but simply reorient the camera from one frame to the next. Since this technique incorporates a loss of frame to frame coherence it will likely lead to disorientation. Besides, it should be obvious that none of these techniques match the movement one would perform when travelling on roads through a real city. Therefore we slightly modify the path derived from the graph G to allow smooth motions and natural camera movement through the 3D city model. This modification is performed by introducing curves at the intersection points along the camera path. In our implementation we use beziér curves to improve the camera behavior at the intersection points. To include such a beziér curve we reduce the length of each road segment and introduce the control points of the beziér curve. Thus, the curve is given by three points: one endpoint from the incoming road segment, one point centered on the intersection and one startpoint of the outgoing road segment. These included beziér curves ensure that neither the camera stops at the intersection to turn into the direction of travelling nor an abrupt turn is noticed by the user. It is important that only short parts of the segments meeting at an intersection are replaced by a curve, because when longer parts would be replaced this could lead to camera collisions with buildings near the intersection. If the width of roads is available for a 3D city model, this information could be exploited to determine an appropriate degree of curvature and thus the length of the segment parts replaced by a curve.

When the camera moves along this modified camera path the speed can be adapted in curves as described in [7]. Furthermore, we have applied an acceleration phase at the beginning and a deceleration phase at the end of the camera movement. Because we always want the camera to be oriented in a way that the look-at-vector is directed in the direction of movement we have introduced a continuous rotation at the very beginning of the movement process.

So far we have only considered 2 DoF movement of the camera with respect to the ground plane, i.e., the altitude of the camera is not altered. Although in some cases it does not relate to real world travelling with a car it may be desirable to change the altitude of the camera as well. Therefore similar to the navigation metaphor proposed in [19] we change the altitude of the camera in relation to the total length of the camera path to travel, i.e., longer distances result in higher camera positions (see Figure 3). It shall be pointed out, that only the altitude of the camera is changed, but the projection of the path onto the ground plane remains unchanged. Thus, the user gets a better overall impression when travelling long distances. As shown in Figure 3 it is helpful to introduce a maximum travel height when using this strategy.

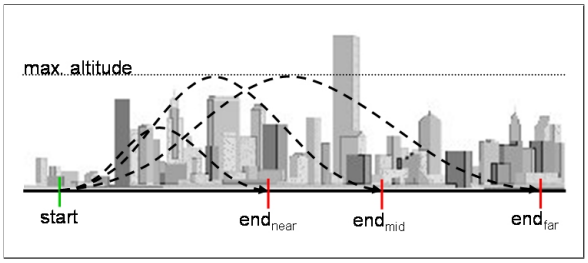
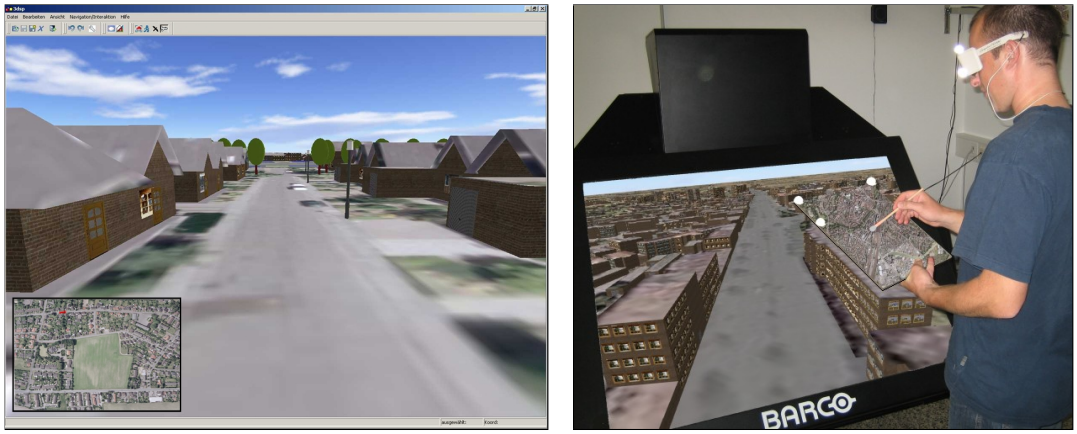


Figure 3: Adaption of the camera altitude based on the length of the computed camera path.

4. The User Interface

Before the system can compute a camera path the user has to specify the travelling destination he wants to navigate to. As mentioned in Section 2 we have combined the WIM metaphor resp. the 2D map metaphor with the *click and fly metaphor* to provide an intuitive interface for the user. Although the same functionality is required for both desktop-based and projection-based VR environments, a different interaction technique is needed since the used input devices have different properties. For desktop-based VR environments we have to provide an interface which is accessible by using the mouse, while in a projection-based environment the user usually interacts with a 6 DoF input device, e.g., glove or wand. For desktop-based VR environments we display the WIM model as a screen aligned head-up display (HuD) which is always visualized on top of the scene content (see Figure 4(a)). The user can specify the destination he wants to navigate to by simply clicking at an arbitrary point on the HuD using the mouse. When the mouse button is released, the current mouse coordinates are used to calculate the corresponding location within the 3D city model, and the camera path calculation is initiated.

In projection-based VR environments where 6 DoF input devices are used it is not appropriate to display the WIM model screen aligned, which is only optimal for 2D interactions. Instead we project the WIM model onto a transparent prop, which is tracked by our optical tracking system. This prop is controlled by the user's non-dominant hand, while the tracked 6 DoF input device can be controlled with the dominant hand to specify the travelling destination by pointing to the prop. In Figure 4(b) the prop is shown, when using with a 2D map metaphor. Since the input device has no special trigger that could be used to specify the destination we utilize topological information to initiate the navigation. Therefore we compare the position of the prop and the position of the input device; in cases where the tip of the input device is near enough to the plane given by the prop we assume that the user wants to start navigating. Besides the intuitive usage the prop also serves tactile feedback, which is an important cue for VR interaction techniques. In both environments, desktop-based as well



(a) User interface for desktop-based VR environments. The HuD can be accessed by using the mouse to change the camera's position and orientation.

(b) User interface for projection-based VR environments. The input device can be used to specify the destination while the view direction of the tracked glasses is mapped to the look-at-vector.

Figure 4: Two different user interfaces for accessing the presented navigation technique.

as projection-based, the current position and orientation of the camera is indicated by visualizing an arrow on top of the WIM view.

Although the proposed navigation metaphor has been designed to support road-based navigation, we still want to support examination of the local area surrounding the user's current location, which can be done by changing the orientation of the camera. Thus, the user is able to view his surroundings from different camera angles. While in desktop-based VR environments the superimposed arrow visualizing the camera can be turned to allow this orientation, we need a different concept for the projection-based VR environments. As it can be seen in Figure 4(b) the user wears a pair of tracked shutter glasses, which allows us to calculate his current heads position. We simply map this position to the virtual camera to support orientation of the camera and thus local scene examination.

With the exception of the previously discussed issues the described navigation widget is the same for desktop-based and projection-based VR environments. In the following subsection we will explain how the visualization of this widget further supports the user during navigation.

The Navigation Widget

The navigation widget visualizes the WIM model of the 3D city model to support the user when specifying the destination he wants to navigate to. To reduce the amount of geometry needed to be sent down the rendering pipeline, instead of visualizing the down-scaled 3D city model we use a resized version of the aerial photograph to serve as a generalized WIM model (see Figure 5). Alternatively one could use a street-map of the city for visualizing the WIM model.

Besides the aerial photograph we visualize the current camera position and orientation as well as the camera path within the navigation widget. As mentioned above and shown in Figure 5, the camera position and its orientation is indicated by an arrow superimposed onto the aerial photograph. The dotted polyline visualizes the current camera path to give the user a better orientation by recognizing the path.

Figure 5: Navigation widget showing the current camera position and orientation as well as the camera path.

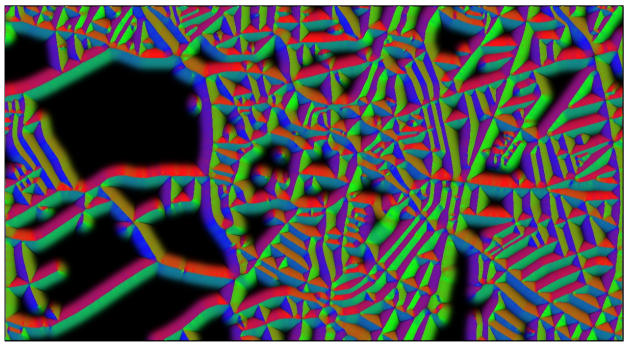
5. Extraction of Road Networks from Cadastral Data

For many 3D city models the road network is already digitally encoded, e.g., as part of a GPS navigation software. In cases where the road network for a given 3D city model is not accessible, there are different options to automatically extract the needed information. There are mainly two strategies, one could either use computational geometry algorithms or image processing algorithms. The former strategy can be applied only if appropriate geometric information for the streets is present, e.g., cadastral data.

If no geometric specification of the streets is available, one can try to extract the road network information by applying image processing algorithms. In the following an image processing approach is outlined which uses the midline extraction algorithm proposed by Rothaus and Jiang [13]. This algorithm calculates gradient images to extract con-

(a) Aerial photograph of the city center of Münster.

(b) Black and white rendering of the cadastral road layers of the same area.



(c) Gradient image created by the midline extraction algorithm.

(d) The extracted midlines superimposed on the road layer rendering with color-encoded tangents.

Figure 6: Extraction of road network information for the city of Münster.

tinuous midlines by still preserving thin structures such as small streets. Although it may be possible to extract the desired information from an aerial photograph, to obtain optimal results, this algorithm needs input images having a reasonable contrast between the background and the structures for which the midline should be extracted. For the city of Münster we were able to extract only those vector layers from the cadastral data which represent traffic areas. The extraction process is illustrated in Figure 6, where Figure 6(a) shows the aerial photograph, Figure 6(b) shows the representation of the road network. In Figure 6(c) the gradient, which has been constructed by the midline algorithm is shown and Figure 6(d) shows the extracted midlines superimposed on the road network, with the tangent direction encoded by color. As shown in Figure 6(b) we have rendered the cadastral layers containing the traffic areas black on a white background to get an optimal result. After applying the midline extraction algorithm to this image, we have obtained a sufficient midline representation of the road network, which is shown in Figure 6(d) where the midlines have been superimposed on the rendering of the cadastral layers. Although for some street intersections the midlines are not continuous this algorithm provides a good result for subsequent graph extraction, because the color encoded tangents simplify the search for connecting street segments.

Alternatively one could use the technique proposed by Ali et al. [1], which originally has been invented to calculate anchor points for label layout. They have developed a two-pass algorithm which computes for every pixel the distance to the closest segment boundary and stores these values in a distance image. Based on this information the skeleton can be extracted.

6. Concluding Remarks

In this paper we have presented a constrained navigation metaphor, which supports improved navigation for exploring 3D city models in virtual environments. The proposed metaphor is applicable in desktop-based as well as projection-based VR environments and can be accessed through an intuitive and easy to use interface, which allows *simple point and click* navigation. When the user has specified the travelling destination the shortest path from the current position to the destination is computed. Since this computation involves the road network of the underlying 3D city model, a very natural navigation is ensured, since the camera moves along roads, and the user experiences similar views as when exploring the city by car or by foot. By adapting the topographic map concept to the user interface, even users with only a little or even no experience with VR environments are able to navigate easily through 3D city models by using the proposed metaphor.

Although our application in both desktop-based and projection-based VR environments has shown a high potential of the proposed navigation metaphor, we will have to conduct a user study to perform a detailed evaluation. Furthermore, different algorithms can be developed for road network extraction. Besides the image processing approach some geometric algorithms may be sufficient for our purpose. Another alternative to provide an appropriate road network is the reuse of geo-data found in navigation sys-

tems. This data is available for almost every city and can be easily processed to be usable with the presented navigation metaphor.

In addition, it may be reasonable to adapt the used path-finding algorithm. In contrast to the shortest path, one could compute a path which involves certain landmarks. Thus, it would be possible to compute camera paths, which would include these landmarks in cases where the shortest path would lead near by. This may be particularly expedient for tourist information systems, where the landmarks are of main interest. Furthermore, the presented concepts can be easily extended to support one-way streets or no-go areas by simply adapting the underlying graph structure which represents the road network.

In addition to the intuitive as well as natural usage of the proposed navigation metaphor, it has a major advantage when exploring large scale urban environments. Because of the high polygon count of these environments special techniques are required to allow visualization at interactive frame rates. When using our navigation metaphor the calculated camera path can be exploited to perform a caching of those graphics objects, which lie along this camera path to improve rendering performance. Since in general the user does not stop the camera during movement or specifies a new travelling destination, the rendering performance benefits from this approach. In the future it has to be figured out, how additional information can be used to enhance this caching, e.g., if it is helpful to remain the buildings along certain streets inside the graphics cache.

7. Acknowledgments

We thank Kai Rothaus for providing us an implementation of the midline extraction algorithm. Furthermore, we would like to acknowledge the city planing department as well as the cadastral department of the city of Münster for providing the datasets of the city of Münster. We also would like to thank the students, who have implemented many parts of the visualization software shown in Figure 4.

References

[1] Kamran Ali, Knut Hartmann, and Thomas Strothotte. Label Layout for Interactive 3D Illustrations. In *Journal of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '05)*, pages 1–8. Union Agency, 2005.

[2] Doug A. Bowman, David Koller, and Larry F. Hodges. Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control Techniques. In *Proceedings of the Virtual Reality Annual International Symposium (VRAIS '97)*, pages 45–52, Washington, DC, USA, 1997. IEEE Computer Society.

[3] Doug A. Bowman, David Koller, and Larry F. Hodges. 3D User Interface Design. In *Course Notes of the 27nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press, 2000.

[4] Jürgen Döllner. Constraints as Means of Controlling Usage of Geovirtual Environments. In *Cartography and Geographic Information Science*, pages 69–80. Cartography and Geographic Information Society, 2005.

[5] Steven M. Drucker and David Zeltzer. CamDroid: A System for Implementing Intelligent Camera Control. In *Proceedings of the Symposium on Interactive 3D Graphics (SI3D '95)*, pages 139–144. ACM Press, 1995.

[6] Sven Fuhrmann and Alan M. MacEachren. Navigation in Desktop Geovirtual Environments: Usability Assessment. In *Proceedings 20th ICA/ACI International Cartographic Conference*, pages 2444–2453, 2001.

[7] Tinsley A. Galyean. Guided Navigation of Virtual Environments. In *Proceedings of the Symposium on Interactive 3D Graphics (SI3D '95)*, pages 103–105. ACM Press, 1995.

[8] Andrew J. Hanson, Eric A. Wernert, and Stephen B. Hughes. Constrained Navigation Environments. In *Dagstuhl '97, Scientific Visualization*, pages 95–104. IEEE Computer Society, 1999.

[9] Ulrich Lauther. An Extremely Fast, Exact Algorithm for Finding Shortest Paths in Static Networks with Geographical Background. In *GI-Days 2004*, pages 219–230. IfGI prints, 2004.

[10] Mark R. Mine. Virtual Environment Interaction Techniques. Technical Report TR95-018, University of North Carolina at Chapel Hill, April 1995.

[11] Randy Pausch, Tommy Burnette, Dan Brockway, and Michael E. Weiblen. Navigation and locomotion in virtual worlds via flight into hand-held miniatures. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*, pages 399–400. ACM Press, 1995.

[12] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 79–80, 1996.

[13] Kai Rothaus and Xiaoyi Jiang. Multi-Scale Midline Extraction Using Creaseness. In *Proceedings of the 3rd International Conference on Advances in Pattern Recognition (ICAPR05)*, 2005.

[14] Brian Salomon, Maxim Garber, Ming C. Lin, and Dinesh Manocha. Interactive Navigation in Complex Environments Using Path Planning. In *Proceedings of the Symposium on Interactive 3D Graphics (SI3D '03)*, pages 41–50. ACM Press, 2003.

[15] Dieter Schmalstieg, L. Miguel Encarnacao, and Zsolt Szalavári. Using Transparent Props for Interaction with the Virtual Table. In *Proceedings of the Symposium on Interactive 3D Graphics (SI3D '99)*, pages 147–153. ACM Press, 1999.

[16] Richard Stoakley, Matthew J. Conway, and Randy Pausch. Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '95)*, pages 265–272. ACM Press/Addison-Wesley Publishing Co., 1995.

[17] Stanislav L. Stoev and Dieter Schmalstieg. Application and Taxonomy of Through-The-Lens Techniques. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '02)*, pages 57–64. ACM Press, 2002.

[18] Zsolt Szalavári and Michael Gervautz. The Personal Interaction Panel — A Two-Handed Interface for Augmented Reality. *Computer Graphics Forum*, 16(3):335–346, 1997.

[19] Desney S. Tan, George G. Robertson, and Mary Czerwinski. Exploring 3D Navigation: Combining Speed-Coupled Flying with Orbiting. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '01)*, pages 418–425. ACM Press, 2001.

[20] Colin Ware and Steven Osborne. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. In *Proceedings of the Symposium on Interactive 3D Graphics (SI3D '90)*, pages 175–183. ACM Press, 1990.

[21] F. Benjamin Zhan and Charles E. Noon. Shortest Path Algorithms: An Evaluation Using Real Road Networks. *Transportation Science*, 32(1):65–73, 1998.