Automatic Generation of 2.5D Terrain Models without Occluding Routes of Interest

Hao Deng, Liqiang Zhang, Jingtao Ma, Liang Zhang, and Dong Chen

Abstract
When a car drives in mountainous regions, the views based on conventional perspective projection often suffer from features of interest being occluded. We propose a method for generating disocclusion views in mountainous regions. The terrain is segmented to build a potential set of occluders; and then an optimized viewpoint is determined, and elevations are rearranged. To obtain a smooth deformed terrain, a smooth displacement function is introduced to deform the level-of-detail terrain models. Compared with previous methods, the merit of this study lies in automatically generating disocclusion views with temporal coherence, while keeping the details of the deformed terrain the same as the original terrain. Experiments performed on the 4098 pixel × 4098 pixel mountainous terrain landscape prove that the disocclusion views can achieve 42 to 58 frames/second. Moreover, the shapes of the features of interest on the driving route without occlusion and the spatial configuration of geographical landmarks in its neighborhood can be easily recognized.

Introduction
Researchers and map providers are endeavoring to construct the geographical environment in 3D to ease the stress associated with wayfinding because for human eyegight, it is much easier and faster to interpret the 3D environment than 2D maps by referencing the current locations, surroundings, and features of interest (FOIs). Currently, 3D geographical visualization has gained a wide application with the development of 3D reconstruction techniques, such as Bulatov and Lavery (2010), and 3D geographical platforms (Zhu et al., 2002) gaining a major advantage over those based on 2D maps. Prevalent application systems usually use perspective projection for rendering 3D spatial objects, which is based on the patterns on how human eyes or cameras observe the external world. In 3D environments however, perspective projection often suffers from occlusions by nearer objects (e.g., the images in the first row in Figure 1). This problem becomes more prominent in certain situations like rugged mountainous regions and cities with myriad buildings. One solution is to compensate the information of the occluded objects by embedding the corresponding 2D maps into the 3D systems, but this method requires users to frequently fall back on 2D map interpretation; this hybrid approach demands more screen space when this is constrained by the applications on mobile devices such as smart phones or portable navigation devices. For these systems, panorama views are preferred over standard projection views, as a panorama captures a visible view from a point in space in all directions. Panoramas can be comprehended as the composite projections of 3D objects from multiple viewpoints; it preserves the similarity between 3D objects before and after the deformations, while all features of interest remain visible. Generally, creation of a panorama is a complicated process and often requires skills of graphics artwork. A number of approaches have been proposed to automatically generate panoramic views (Takahashi et al., 2006; Takahashi et al., 2008; Möser et al., 2008; Degener and Klein, 2009); however, these approaches found their limits in interactive 3D visualization applications, or in some cases introduced exaggerated distortions of the spatial objects. In short, it has been considered a challenging problem to achieve an interactive and automatic generation of realistic views without the FOIs being occluded to ensure the view readability.

This paper focuses on real-time generation of 2.5D disocclusion views for terrain navigation across mountainous regions (the images of the second row in Figure 1). The generated views will ensure the resemblance to the FOIs and landscapes in appearance. As a result, these new views suffice the need for interactive occlusion-free navigation in 2.5D complex terrain environments. Practical applications can benefit directly from the research here, for example, any ski resort map of pixels that can show the landscape with the occlusions addressed here removed to create a clear mind map for winter sports fans.

Related Work
In recent years, many approaches can generate non-perspective views of 2.5D landscape for occlusion reduction. Existing methods in this field can be classified into single camera projections, composite projections and object-based deformations. Mei (2005) designed a camera which bends the line of sight in the vicinity of silhouette to reveal occlusion. Yang et al. (2005) developed magic 2.5D lenses. Related work is exemplified by observer dependent
deformations (Marttin et al., 2000) which automatically deform the model according to the viewpoint, and visual access distortion (Carpendale et al., 1997) which is also dependent on the view direction.

Mesh deformation can be implemented by space-based approaches (Sederberg et al., 1986; Lipman et al., 2008) or surface-based ones (Botsch and Kobbelt, 2004). The space-based approach deforms the mesh by changing the space to only embed the initial mesh, while the surface-based techniques work directly on the initial mesh. These techniques allow specifying the position constraints on meshes, which gives more freedom to control the mesh shapes more intuitively. The surface-based techniques can preserve such low level shape characteristics as the curvature, normal or local rigidity (Gal et al., 2009).

Patterson (2000) summarized the panorama-generating techniques used by Heinrich Berann, the earliest pioneer of the modern panorama map. A taxonomy by Elmqvist and Tsigas (2008) provided an overview and classification of the existing methods. Degener et al. (2008) reduced the occlusion of short routes by a space deformation that aligns the route with the view direction. Qu et al. (2009) presented a focus plus context zooming technique to broaden the occluded roads with other context being maintained. Meanwhile, Takahashi et al. (2002) presented an approach for generating panorama guide-maps, which can calculate the landscape displacement and rotation semi-automatically. However, all these approaches are not capable of producing panoramic views automatically.

Moser et al. (2008) developed interactive context-aware visualization systems which reveal FOIs by distorting the scene dramatically, but the distortion causes the resulting views to be too far from the ordinary perspective. Takahashi et al. (2006) proposed an occlusion-free route animation for car navigation. The occlusion is eliminated by the arrangement of landmarks onscreen. This process may be tedious for generating occlusion-free views at the interactive frame rates if the number of landmarks is large. This approach also needs the geographical landmarks to be sampled from different viewpoints on the hemisphere that covers the terrain in order to consider all occlusion situations. Therefore, the number of landmarks is usually large, and hard to keep small without introducing artifacts. The work (Degener and Klein, 2009) used non-linear optimization to search for a terrain deformation to produce panoramic maps. The approach considers two factors, resemblance and visibility, to constrain the possible deformation. While the non-linear optimization process can find satisfactory deformation results, its computational cost is too high for interactive applications. Whereas, their approach obtains optimized rearrangement of local geographic features in real-time, and then searches for the deformed terrain by a linear global optimization, which allows deforming the terrain mesh interactively. Cui et al. (2010) introduced a multi-perspective visualization technique based on the curved ray camera to overcome occlusions in visualization while minimizing distortions. The camera supports both 2.5D surface and volume datasets. Rosen and Popescu (2011) used a multi-perspective image framework that allows trading off occlusions for image simplicity. The framework interactively

Figure 1. Visualization of terrain and roads (in black). Ordinary perspective projection (the top three images) and their corresponding temporal coherent disocclusion perspective views (the bottom three images).
renders multi-perspective images, which enables investigating navigation and dynamic 2.5D scenes. Deng et al. (2011) presented an automatic method for generating panoramic map-like views in mountainous areas. The generated views can keep the resemblance in appearance to the features of interest and landscapes. However, in these views the FOIs and landscape are moved down on the 2D computer screen. Especially when some FOIs and geographical landmarks in the foreground are out of screen, the sky occupies more screen space in the background. Moreover, for large-scale terrain landscapes, the final views are difficult to be put into interactive visualization mode. Furthering the work of Deng et al. (2011), this paper presents an approach for implementing interactive occlusion-free navigation in 2.5D complex and large-scale terrain environments. The FOIs and geographical landmarks in the view are arranged as close as possible to the screen positions in perspective views.

Methodology Overview

Given a terrain model \( T \), a viewpoint \( P \), and a set of the FOIs, \( F \) is composed of points, lines, or polygons on \( T \). Generally, the view \( V_p \) in the conventional projection observing from \( P \) along direction \( D \) can be obtained. This paper now focuses on the approach for generating a similar view \( V \) with \( V_p \) while making \( F \) visible on \( V \).

The camera is first moved up to get a good viewpoint for reducing occlusions of the FOIs, and then the shape of the terrain \( T \) is altered to create an occlusion-free view. A seven-step approach is developed as follows (see Figure 2):

1. Preprocessing: Partition the terrain according to the geomorphic features such as elevated mountains, and set reference points on the FOIs;
2. Viewpoint selection: Move the camera up so as to get an appropriate overview of the FOIs and landscape;
3. Rearrangement: Detect the occluders, and rearrange the elevations that occlude the FOIs on the 2D terrain field;
4. Deformation: Deform the terrain based on the rearrangement results;
5. Acceleration: Enhance the interaction of the deformed large-scale terrain visualization;
6. Postprocessing: In the screen space, adjust camera parameters to bring the FOIs in occlusion-free views visually closer to the same positions as their original perspective projection;
7. Landscape rendering: The occlusion-free terrain views are rendered.

At the Preprocessing Step, the terrain surface is segmented to build a potential set of occluders. Generally, occluders that block the lines of sight are convex parts of the terrain, such as peaks and hills. Identifying and extracting these occluders is naturally the first task, based on the geomorphic features of the terrain of interest. A method of finding the Morse-Smale complex in the Morse theory is applied to accomplish this task (Comic et al., 2005); except the preprocessing is irrelevant to the viewpoints. The other steps are performed in real time.

It is worth introducing the ordinary ray-tracing process for better understanding the proposed approach. Ray tracing serves two purposes: detect the visibility of the reference points, and locate the occluders. Let us specify a reference point \( P(x_p, y_p, z_p) \) and a viewpoint \( V(x_v, y_v, z_v) \). Given a terrain height function \( z = f(x,y) \), if there exists any point \( Q(x_q, y_q, z_q) \) on the line of sight (LOS) that satisfies \( z_q < f(x_q, y_q) \), the reference point is occluded; Otherwise, the reference point is visible. Here the point \( Q \) on the LOS can be denoted as \( Q = (1-t)P + tv, 0 \leq t \leq 1 \). Starting from \( t = 0 \), cast the ray by letting \( t = t + \Delta t \) until \( t = 1 \), where \( 0 < \Delta t \leq 0.1 \). During the process, once the ray reaches the terrain, the reference point is detected as being occluded. Denote the LOS to \( P \) as \( \text{LOS}(P) \), and the line composed by the set of points \( (x_q, y_q, f(x_q, y_q)) \) the projection of \( \text{LOS}(P) \). This notation is also followed in the next sections.

Viewpoint Determination

As FOIs are more likely to be occluded by mountains in the foreground, a proper viewpoint can be determined by choosing a higher view position for a better overview of the FOIs and landmarks. In order to keep sky and as many FOIs at a far place as much as possible, the position of the camera should only be moved higher vertically while retaining the tilt angle unchanged. However, retaining the tilt angle may potentially move some front FOIs out of sight; this problem will be addressed in the Postprocessing Step (Step 6).

If the view position is low, the FOIs are more likely to be occluded by mountains in the foreground. In order to get a better overview of the FOIs and geographical landmarks, we determine an appropriate viewpoint by choosing a higher view position. Meanwhile, to keep sky and more FOIs visible at far places, our approach only moves the camera higher and still retain the tilt angle (unchanged) of the camera. However, preservation of the tilt angle can cause some front FOIs to be out of sight. This problem is addressed in the postprocessing step.

Optimizing Camera Height

Two criteria are applied to estimate the camera height: (a) the camera height minimizes the terrain distortion, and (b) the camera movement is kept as minimal as possible. The first criteria maintains the shape of the terrain while it reduces the computation cost to rearrange the occluders. The second criteria makes the final views close to the views observed from the original camera. An energy function will be designed to measure the camera height.

The occluder regions should be detected first, and then the downscaling amount \( d \) for the occluder regions is estimated: \( d \) is used to formulate the changes of the occluder regions in the quadratic form. The entire terrain satisfies Equation 1:

\[
E_f(h) = \sum_{h \in O} w_i d_i(h)^2
\]

where \( O \) is the set of occluder regions, and \( w_i \) is the weight set to be the normalized area of the region.

Ray tracing from each reference point is applied to detect occluder regions. To estimate the amount \( d_i \) for each
occluder region \( C_i \), calculate the maximum height \( h_i^{\text{max}} \) of the viewpoint at which \( C_i \) does not occlude any FOI. Obtain \( h_i^{\text{max}} \) by computing the minimum slope \( \kappa_i \) of the LOS originating from the reference points and passing through the occluder \( C_i \). Figure 3 illustrates a 2D scenario for a single reference point. During ray tracing, when the occluder region \( C_i \) is found, it satisfies Equation 2:

\[
\kappa = z'/l
\]  
where \( z' \) is the relative elevation to the reference point at the current step, and \( l \) is the length that the ray advances in the 2D terrain field. Here \( h_i^{\text{max}} \) is updated to be:

\[
h_i^{\text{max}} = \max \{ |h_i^{\text{max}}, z_p + \kappa L | \},
\]  
where \( z_p \) is the elevation of the current reference point, and \( L \) is the distance between the reference point and the view position in the 2D terrain field. After ray tracing from all reference points is finished, \( h_i^{\text{max}} \) is computed for all occluder regions. Given the camera height \( h \), the needed vertical camera movement \( D_i \) is computed in the way so as to clear the LOS blocked by \( C_i \):

\[
D_i = h_i^{\text{max}} - h.
\]

where \( D_i \) is related to the needed downscaling amount \( d_i \) for \( C_i \):

\[
d_i = k_i D_i.
\]

The ratio \( k_i = l/L \) can also be obtained during ray tracing; it is recorded for each occluder region \( C_i \) and updated each time. Update \( h_i^{\text{max}} \), according to Equation 3. Combining Equations 1, 4, and 5 can result in the final terrain changes term:

\[
E_i(h) = \sum_{i \in O} w_i k_i^2 (h_i^{\text{max}} - h)^2.
\]

Since the camera movement only causes height changes, given a camera height \( h \), measure the height changes from \( h \) to the user specified height \( h_0 \) in the quadratic form:

\[
E_i(h) = (h - h_0)^2.
\]

Overall, the camera height \( h' \) can be solved by minimizing the energy function:

\[
E_i(h) - E_i(h') + \lambda E_e(h)
\]

where \( \lambda \) is set to be 0.1 in practice. Note that the set of the occluder regions \( O \) may be changed when a new \( h' \) is solved, so the energy \( E_i(h') \) may not be minimized. For this reason, the variants in the energy term \( E_i \) are iteratively updated to minimize the overall energy \( E_{h_i} \). Let \( h' \) be the camera height at time \( t \). To obtain the camera optimized height, iteratively solve \( h' \) until a terminal condition is satisfied:

\[
h^{t+1} = \arg \min_h E_i(h').
\]

The iteration is terminated until the changes of the camera height between two successive time-steps are less than a given threshold value. To accelerate the viewpoint selection process, only a representative subset of the reference points is used. The subset selection is carried out by a uniform sampling from the reference points on the FOIs.

**Approach for Preserving Temporal Coherence**

The camera height is only optimized in an individual frame. Naively moving the camera to an optimized height often lead to temporal incoherence, flickering, or waving in the view. To address this problem, this paper takes the sequence of the camera heights as a spatiotemporal curve. The camera height is constrained to be coherent by smoothing the curve. The discrete Kalman Filter algorithm is applied to filter the curve. It performs as a sequence of predict-and-correct cycles. In each cycle, it estimates a process by a form of feedback control: the filter estimates the process state and then obtains feedback in the form of measurements, i.e., predicting the camera height in the current frame to be the same as the one in the last frame. The measurement is considered as the optimized camera height involving a kind of unexpected changes causing flickering or waving effects. It is incorporated to correct the prediction of the camera height during which the noise is filtered. When users control the camera with views being animated, the discrete Kalman filter cycles update the camera height. Figure 4 shows the spatiotemporal curves of the camera height in 721 frames before and after filtering. The comparison indicates that the Kalman filter-based method is effective for smoothing camera heights and keeping temporal coherences.
Occlusion-free Rearrangement

There are two situations where a reference point is invisible: (a) the reference point is located on the back-face meshes far from the silhouette, and (b) the reference point is occluded by the terrain. Use the following method to process these two situations.

Handling the Reference Points

First, the entire region is scaled down if it is determined that both the reference point and the hit are in the same region. It could also be the case that the region is back-facing; then we anisotropically scale down the terrain by altering the part which is nearer than the reference point, since the rest of the region do not cause occlusions. Otherwise, the scaling will be applied to the whole region.

A piecewise scaling is used in the anisotropic implementation. Each occluded reference point in the region will lead to the scaling-down of that sub-region. Given a point \( Q(x_p, y_p, z_q) \) at the LOS with \( z_q < f(x_p, y_p) \), Equation 10 is obtained:

\[
k = (z_q - z_c)/(f(x_q, y_q) - z_c).
\]

where \( \epsilon \) is an infinitesimal, ensuring terrain surface is below the current LOS.

Despite the fact that the obtained parameters can eliminate occlusions, the parameters of different regions are discontinuous, which may cause the view discontinuity and gap artifacts on the deformed terrain (see Figure 5). So the parameters are not directly applied to the terrain.

Displacement Approximation

To obtain a smooth deformed terrain, a displacement smoothing function is introduced to avoid the discontinuities and gap artifacts illustrated above. Meanwhile, the details of the terrain should be kept before and after the deformation. Several methods can be used to achieve this goal, such as Bary-centric coordinates (Floater et al., 2005), B-spline interpolation (Lee et al., 1997), and radical basis function interpolation (Micchelli, 1986). These methods, however, either need the interpolation which is not smooth enough, or would prohibitively expensive in computing. Here we apply an elevation displacement smoothing function based on the moving least-square method (Levin, 1998), for it is considered highly flexible and demonstrates good approximation properties.

Given \( n \) control points located at the positions \( x_i \) with the elevation displacement \( u_i \) in the 2D terrain field, where \( i \in [1 \ldots N] \), we first obtain a global smoothing function \( u(x) \) approximating the given elevation displacement at points \( x_i \). For a position \( x \) in the 2D terrain field, the best affine transformation \( u_x(x) \) is searched that minimizes:

\[
\sum \| \theta(d_i) \| u_x(x_i) - u_i \|^2 \tag{11}
\]

where the displacement \( u_x(x) \) can be denoted as:

\[
u_x(x) = b(x)^T c_x, \tag{12}\]

where \( c_x = [c_1, c_2, c_3]^T \) is the vector of the coefficients to be solved and \( b \) can be \([1, x, y]^T\). The weighting factor \( \theta(d) \) has the form \( \theta(d) = 1/\| x - x_i \|^2 \). The final global displacement function \( u(x) \) is obtained from a set of local functions:

\[
u(x) = u_x(x) \tag{13}\]

The global function is continuously differentiable only if the weighting factor is continuously differentiable (Levin, 1998). By solving Equation 11 in the least-square sense, a \( 3 \times 3 \) linear system is obtained, which yields the coefficient \( c_x \) to interpolate the displacement \( u_x(x) \) at the position \( x \) through Equation 12, so,

\[
c_x = \left( \sum \theta(d_i)b(x_i)b(x_i)^T \right)^{-1} \sum \theta(d_i)b(x_i)u_i \tag{14}\]

where \( p_x \) denotes the vector of the weight coefficients, and \( u \) stacks the elevation change of the control points. The computation of the coefficients \( c_x \) for each position in the 2D terrain field is too time-consuming; fortunately, the coefficients \( c_x \) are a linear combination of \( u_i \). Therefore, combine Equations 11, 12, and 13 to get Equation 15:

\[
u(x) = b^T p_x^T u = w^T u \tag{15}\]

where the vector \( w \) only relates to the position \( x \). So the vector \( w \) can be pre-computed. In vector \( w \), there is only a small set of elements \( w_i \) corresponding to the positions neighboring \( x \) that have non-zero values. In the deformation, take the neighbor set \( N(x) \) into account, and let \( u(x) \approx \sum_{i \in N(x)} \hat{w}_i u_i \), where \( \hat{w}_i \) is the normalized weight yielded by

\[
\hat{w}_i = \frac{w_i}{\sum_{i \in N(x)} w_i}. \quad \text{The computational load is greatly reduced in this way; for example, it only took three minutes to create 19,642 triangles for the deformation. This example further reflects the efficiency of the proposed MSL-based method.}

Extraction of Terrain Control Points

Note from Equation 14 that the elevation displacement function \( u(x) \) is highly related to the positions of the control points; to find the representative control points on the terrain field for the displacement function approximation, we first reiterate the following two simple facts: (a) the higher elevations are more likely to be displaced, and (b) in each region where FOIs are located, the deformation is more complex. Thus, to better approximate the displacement function, it is an intuitive step to set control points at positions along with higher elevations, while for the second
case, those regions with residing FOIs will naturally have more control points.

From the above reasoning, we propose a graph-based technique to select control points. The graph is constructed in the 2D terrain field; this algorithm consists of four steps, which are performed at pre-processing:

- Step 1. Construct an initial graph for the terrain, whose vertices are potential control points and edges corresponding to the edges of Delaunay triangulation of the vertices (Figure 6a);
- Step 2. Reshape the graph by deleting the shorter edges using the edge collapse (Figure 6b);
- Step 3. Move the vertices to the positions with higher elevations (Figure 6c); and
- Step 4. Simplify the graph again similar to the procedure in the above Step 2. The positions of the vertices compose the final set of representative points (Figure 6d).

Initially, the terrain in a 2D square lattice is embedded. For the regions where the FOIs are located, the vertices are denser than those of the regions with no FOIs. Next, more vertices are set at the peaks of each region. To construct the edges, use these vertices again to create Delaunay triangulation. The generated triangulation forms the initial graph. During the graph simplification, traverse each vertex and find its shortest incident edge. If this edge’s length is less than a threshold, it is collapsed to the nearest incident vertex. In the implementation, the threshold is the half of the average edge length of the graph. After all vertices are traversed, the control points that are too close with each other are deleted.

**Acceleration of Detection and Deformation**

Detection of occluder regions can consume a great amount of computer RAM. We have developed a multi-thread technique to speed up the detection and deformations (see Figure 7). The first few threads are devoted to data operation and refinement that run in the back end of the application communicating with the external processes and fetch data from the disk. The data operation thread is invoked to compute the visible region once the viewpoint changes, and pre-fetch the current level of the terrain model. The refinement thread is responsible for updating the terrain vertex array and computing the terrain visibility in real-time. Since the disk is the slowest component of the system and new geometries need to be loaded from it, a pre-fetching algorithm that runs parallel to the detection thread is used to reduce the latency. The eye point $E_n$, target point $R_n$, and view direction $L_n$ of the next frame can be predicted based on the parameters of the previous and current frame (see Equation 16):

\[
E_n = E_c + \mu(E_c - E_p) \\
R_n = R_c + \rho(R_c - R_p) \\
L_n = R_n - E_n
\]

where $E_c$ and $E_p$ are respectively the eye points of the current frame and previous frame; $R_c$ and $R_p$ are respectively the target points of the current frame and previous frame; $\rho$ and $\mu$ are variables. Generally, $\rho = \mu = 1$.

Detection of occluder regions by ray tracing needs CPU resources, while loading new data from the storage unit to the main memory and then to the graphics board is often the bottleneck in the overall deformation process. Use of the programmable graphics processors to process them, can improve the detection and deformation speed and save CPU consumption.

High throughput from graphics cards can be further enhanced by storing the vertices of terrain on the GPU at render time, thereby reducing the data transferred from the CPU to the GPU during each frame. A second caching mechanism loads the visible parts into the video RAM of the graphics card. As a result, only the terrain vertices are transmitted to the GPU and the other vertices are cached in the GPU memory. Through these processes, the detection and deformation procedures are accelerated.

**Postprocessing**

After the camera is moved up and the terrain is deformed, naïve disocclusion views are obtained. However, in these
views the FOIs and landscape are moved down into the 2D screen. Especially some FOIs and geographical landmarks in the foreground are out of screen (see Figure 8), while sky occupies more screen space in the background. For the viewers’ perception benefit, however, the foreground FOIs and geographical landmarks are more important and should be preserved well in terrain navigation, because viewers can more easily recognize these features within the surroundings. Moreover, the FOIs and geographical landmarks in the view should be arranged as close as possible to the screen positions in perspective views.

This step is achieved by adjusting the camera parameters. Since the main purpose of the rearrangement is to move up the FOIs and geographical landmarks in naïve views, we then adjust the vertical coordinate of the projection center (COP), and move the camera along the view direction. These two parameters are selected for three reasons: (a) both of these parameters can help move up the FOIs and geographical landmarks, (b) the adjustment to these parameters do not lead to new occlusions, and (c) the relation between the parameters and projection is easy to be identified, which make the following optimization simple for implementation.

The parameters are adjusted by solving an optimization problem, which brings the resulting projection positions as close as possible to those for all reference points and geographical landmarks in the perspective views. Let $s$ denote the camera movement along the view direction $I$, and $v$ denote the vertical coordinate of COP. The projection $P_{s,v}$ of a position $p$ in the screen coordinate system can be computed as:

$$P_{s,v}(p) = \left( \frac{(SMp)_x}{(SMp)_z + s}, \frac{(SMp)_y}{(SMp)_z + s} - v \right)^T$$  \hspace{1cm} (17)

where the matrix $M$ is the model view matrix and the matrix $S$ is the scaling part in the projection matrix, and $(.)_{x,y,z}$ indicates the values of individual components. Let $P_0$ indicate the ordinary perspective projection of the original camera. For the reference points $p$ and geographical landmarks $q$, minimize the following energy function:

$$E_{\rho}(s,v) = \sum_{i \in R} W_R ||P_{s,v}(p_i) - P_0(p_i)||^2 + \eta \sum_{i \in L} W_L ||P_{s,v}(q_i) - P_0(q_i)||^2$$  \hspace{1cm} (18)

where $R$ and $L$ are the set of reference points and set of geographical landmarks, respectively; $w_R$ and $w_L$ are the weights corresponding to these two sets respectively, and $\eta$ is set to be 0.1 in the implementation. Here the set $L$ of geographical landmarks is set to be the peak of the regions in the view frustum, while $w_R$ and $w_L$ are in proportion to the reciprocal of the squared distance to the viewpoint. In practice, users do not wish the camera to move forward such that some surrounding FOIs and geographical landmarks in the foreground are still on the screen. So constraint $s \geq 0$ in Equation 18.

Note that naively animating the views with the optimized camera parameters also leads to temporal incoherence. Adding temporal coherence constraint is accomplished by minimizing the difference of the parameters between successive frames:

$$E_T(s,v) = (s^t - s^{t-1})^2 + (v^t - v^{t-1})^2$$  \hspace{1cm} (19)

where the superscripts of $s^t$ and $v^t$ indicates the parameters $s$ and $v$ are at time $t$. By combining Equations 21 and 22, the optimization can be formulated as:

$$\arg \min_{s,v} \left( E_{\rho}(s,v) + \gamma E_T(S,v) \right)$$  \hspace{1cm} (20)

subject to $s \geq 0$

which is a non-linear least-square optimization. We take an iterative approach based on Gauss-Newton method to solve $s$ and $v$, which possesses good convergence performance. In most cases, Equation 20 can be solved in five time-steps. To accelerate each time step for solving Equation 20, only use a representative subset of the reference points, as this paper deals with the optimization in the viewpoint selection (see the Viewpoint Determination Section).

After the postprocessing, the FOIs and geographical landmarks in the view are arranged as close as possible to the screen positions in perspective views (Figure 9).

**Terrain Rendering**

Currently, users usually adopt level-of-detail (LOD) algorithms (e.g., Yang et al., 2005) to render large-scale terrain landscapes. In LOD terrain models, the deformation does not introduce new vertices, making it a good candidate method for terrain deformation in this. To accelerate the deformation, the computation of vertices deformation is shifted to the vertex program of GPU. In the vertex program, if the segment where a vertex belongs needs to be deformed, the CPU transmits the corresponding deformation parameters to the vertex shader on GPU.
While the segment-based terrain deformation facilitates eliminating occlusions, it brings the discontinuities of the terrain surface, for the deformation parameters cannot preserve the continuity among segments. In order to achieve the continuity, we use the generic mesh refinement technique to deal with the triangular mesh in the LOD terrain models across the boundaries of the related segments, so that these meshes near the boundaries could be smoothed. In this approach, the procedures for additional vertex generation and vertex displacement are accomplished on GPU. First, a refinement scheme need to be set up and transmitted into a vertex buffer beforehand to avoid occupying the bandwidth of graphic bus at render time. For each triangular mesh to be refined, the rendering thread sends the coordinates of its three vertices to the vertex shader. According to the input vertices coordinates, the vertex shader calculates the position of the refined triangular patches. Moreover, Bézier polynomial interpolation is used in the vertex shaders to displace the vertices. This procedure provides visual smoothness to the terrain mesh which alleviates the discontinuities. The coefficients of Bézier polynomial could be computed on the CPU according to the coordinates and normal vectors of the vertices in the triangular meshes to be refined, and then sent to the vertex shader.

However, such schemes aim at global refinement and smoothness of original triangular meshes, while in this study it just needs to refine and smooth the triangular meshes that pass across different segments. In order to seamlessly render the refined triangular meshes and leave the other triangular meshes untouched, the LOD model is rendered without the refinement first. During the rendering process, the triangular meshes crossing different segments are identified and stored in the LOD terrain model traversingly. Next, we render the stored triangular meshes to be refined by the above approach. Here cracks may occur between those triangular meshes without refinement and smoothness, because vertices of the refined triangular meshes are displaced. To address this issue, the displacement of the vertices on the boundaries of the refinement pattern is constrained. Only execute the displacement to the vertices within the refinement pattern. Figure 10 illustrates

![Figure 9. Views before deformation and after the post-processing: (a) view before deformation, and (b) view after the post-processing.](image)

![Figure 10. Rendering results (a) without, and (b) with the triangular patches crossing blocks being refined and smoothed.](image)
the triangular patches crossing different blocks with and without being refined and smoothed, respectively.

**Experiments**

To validate the performance of the proposed method and compare with the approach of Deng et al. (2011), this paper also performs the tests on the terrain located in northern Shanxi, China on a PC with an Intel Core2 Quad Q6600 (2.4 GHz), 4 GB main memory, and an ATI Radeon HD 4650 graphics card with 512 MB video memory; but the terrain’s size becomes 4096 × 4096 pixels.

This paper tests the performance of the proposed approach in two conditions:

**Condition 1**

With no acceleration of detection and deformation (see the *Acceleration of Detection and Deformation Section*), we compare the frame rates with/without occlusions along the same camera path.

The companied video compared the standard perspective views with the final views. The tracks occluded by a hill and those occluded by the mountains in the distance are visible in the views. At the same time, most of the non-occlusion terrain is kept the same as the original shapes. The scenario in *Navigation in the West of Hunan Province: Route II* of the video is viewed from the north. Here the viewpoint is located at the low land to the north of the junction where two tracks meet. In the view, both of the tracks are again successfully displayed. Viewed from the hillside in the southeast of the terrain, the western tracks are invisible because the occlusion is caused by the mountains. With the proposed method, the occluded tracks in the western and northern valleys are revealed. In *Roaming Along the Hiking Tracks in the North of Shanxi Province: Route I* of the video, the route is blocked because it lies on back-face surface in the perspective views. The proposed method can still handle this kind of occlusions by scaling down the terrain anisotropically. The approach is further used to polygon features, and the views are illustrated by the scenes of the road and reservoir in the *North of Shanxi Province: Navigation Along the Road* and *User Interaction* of the video where a lake is set to be the FOI. Most parts of the reservoir are out of sight. The approach succeeds in revealing the full view of the polygon feature, while disclosing the road bypassing the reservoir. Integrating the deformation along with moving camera up, satisfactory views are obtained in which the distortions are further alleviated and the occlusions are avoided at the same time.

In the disocclusion view mode with the deformation, the rendering efficiency is much dependent on the number of ray tracing. In the mode of panoramic views with the deformation, the rendering efficiency is much dependent on the number of the ray tracing. Figure 12 shows the relation between the frame rate and the count of ray tracing when navigating along the path same as Figure 11, and only the reference points within view frustum are traced. When the count of ray tracing varies from 200 to 1,181, it basically keeps a linear relation with the frame rate. The turbulence appears when the count of the ray tracing falls between 0 and 200. The reason is that the viewpoint is on the path segment corresponding to the interval from 25s to 30s shown in Figure 11, in which the frame rates decline. At this time, the frame rate of panoramic views is already quite close to that of perspective views. To resolve the bottleneck caused by the ray tracing, we reduce the samples of the reference points and enlarge the step length of the ray tracing. However, this may potentially lead to occluders not being found, and thus a decline in the panoramic view quality. Generally, to serve the purpose of interactive visualization in PDAs without high computation capabilities, the real-time generation of panoramic views proves effective.

**Condition 2**

The second condition is to adopt the acceleration of detection and deformation. The frame rate of the approach is also tested in the experiment. Keeping the same the count of ray tracing with (Condition 1), the approach with acceleration of detection and deformation can obtain 42 fps to 58 fps. Therefore, in this condition (i.e., Condition 2), the frame rate of the animation is much higher than that of (Condition 1).

Compared with the method of Deng et al. (2011), the proposed approach makes the FOIs and geographical landmarks in the view arranged as close as possible to the screen positions in perspective views. Moreover, this speeds up occlude region detections and terrain deformation.

![Figure 11. Frame rates in perspective views and panoramic views, respectively.](image)

![Figure 12. The relation between frame rates and count of ray tracing.](image)
The performance the proposed method only depends on the graphics hardware moderately, which facilitates transferring the method to mobile platforms. Note that only by using an ordinary CPU-based terrain rendering algorithm, the method is able to run on PCs at satisfying interactive frame rates with an acceptable memory consumption. Currently, mobile phone CPUs with a main frequency higher than 1 GHz have emerged; high-end mobile platforms such as the iPad™ and PSP have exhibited similar powerful capabilities. High performance can also be potentially gained on mobile platforms by only disocclusion of a part of the front features, and by terrain deformation at a further reduced hierarchy.

**Shortcomings of the Approach**

It was discovered that the proposed method may cause an excessive vertical scale-down of some mountains when eliminating occlusions under some extreme situations. Sometimes it is not conductive for users to interpret the disocclusion views. An example of the problem is given in Figure 13.

In addition, the method only deals with the occlusion in mountainous areas. For urban scenes, occlusions can not be simply avoided by scaling down the occluders, the other techniques such as moving up the view position and displacing the occluders may also be employed. The authors are actively working on these two issues as an immediate next step.

**Conclusions**

This paper has presented a method to automatically generate disocclusion views for terrain navigation in mountainous regions. The created disocclusion views are effective in eliminating occlusions of the FOIs caused by rugged terrains. It takes a viewpoint, a view direction, DEM, and FOIs as input data, and generates disocclusion views directly by choosing a higher viewpoint and deforming the terrain at the same time. The approach is to segment the terrain to build a potential set of occluders in the preprocessing, and choose an optimized viewpoint and rearrange elevations with the support of ray tracing. Compared with previous methods, the study merit lies in automatic generation of disocclusion views with interactive frame rates where the FOIs are free from occlusions.

When encountering the terrain occlusions in some extreme cases, the disocclusion views from the proposed method can also deal with the occlusions excessively, and such views are not conductive for users to interpret in the case of some extreme terrain occlusions. It is also a challenge to implement interactive visualization for city navigation in the panorama views. In future work, to further enhance the validity and readability of disocclusion views, some constraint mechanisms will be introduced to avoid excessive deformations.

**Acknowledgments**

The authors would like to thank the editor and the anonymous reviewers for their valuable comments and suggestions. The study is supported by the National Natural Science Foundation of China (No. 60502008) and the National 863 High-Tech Program of China (No. 2011AA120302).

**References**


(Rceived 05 September 2011; accepted 07 December 2011; final version 14 March 2012)