박 사 학 위 논 문
Ph.D. Dissertation

대화형 관광 지도의 변형을 위한 적응형 그리드 기법

Adaptive Grids towards Interactive Tourist Map Deformation

2017

클라우디오 피오 (Claudio, Benjamin Pio)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

박사학위논문

대화형 관광 지도의 변형을 위한 적응형 그리드 기법

2017

클라우디오 피오

한국과학기술원

전산학부

대화형 관광 지도의 변형을 위한 적응형 그리드 기법

클라우디오 피오

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2016년 11월 18일

심사위원장 윤 성 의

심 사 위 원 김 민 혁

심 사 위 원 박 진 아

심 사 위 원 서 진 욱

심 사 위 원 우 운 택

Adaptive Grids towards Interactive Tourist Map Deformation

Benjamin Pio Claudio

Advisor: Sung-Eui Yoon

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science

Daejeon, Korea
November 18, 2016

Approved by

Sung-Eui Yoon
Associate Professor of Computer Science

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

DCS
20125376

클라우디오피오. 대화형 관광 지도의 변형을 위한 적응형 그리드 기법. 전산학부 . 2017년. 61+v 쪽. 지도교수: 윤성의. (영문 논문)
Benjamin Pio Claudio. Adaptive Grids towards Interactive Tourist Map Deformation. School of Computing . 2017. 61+v pages. Advisor: Sung-Eui Yoon. (Text in English)

초 록

관광학 연구에서 이르는 관광용 지도는 주요 관심지점 파악, 경로 계획, 지리적 정보의 제공 등 크게 세 가지 기능을 제공하는 것으로 여겨진다. 모바일 장치와 디지털 지도의 시대에, 상기 기능을 사용자에 맞추어 동적·대화형 시점으로 지도에 구현할 필요가 있다. 또한

본 연구에서는 도시 관광을 위한 주요 운송 수단으로 지하철을 고려한다. 기존의 툴은 노선도에 효과적으로 관심지점을 가시화하지 못한다. 이를 보완하기 위해 관심지점의 표현과 노선도의 표현을 결합하는 대화형 프레임워크를 제안한다. 본 프레임워크는 관심지점에 대한 가시적 중요도를 계산하고, 사용자가 지하철 교통망으로부터 접근 가능한 관심지점을 찾아 제시한다. 지하철 노선도의 표현에 8방향 레이아웃이 사용되고, 주요 관심지점의 이미지가 사용자의 시점에 맞추어 레이아웃 상에 가시화된다.

또한 사용자 특화 웹 서비스의 시대에 맞추어 온라인 맵 서비스 역시 사용자 특화를 지향하고 있다. 이러한 경향을 충족시키기 위해서는 특정 작업에 특화된 지도 표현을 실시간으로 처리할 수 있어야 한다. 지도 변형을 위한 선행 기술들은 대개 균일 격자에서 구현되어 실시간으로 처리하기 어려울 수 있다. 이를 극복하고 해상도를 유연하게 할당하기 위하여 컨텐츠 맞춤 비균일 격자를 제안한다. 제안된 구조는 안내 도로 패턴 등 주요 영역을 더 많이 분할시키고, 덜 중요한 영역에서는 영역을 덜 분할시켜 처리해야 할 모서리 정보의 수를 감소시킨다. 이러한 기법으로 내용에 기반한 적응형 해상도를 구현하여 지도를 변형하면 더 높은 변형 수준과 성능을 얻을 수 있다.

마지막으로 제안된 기법들을 여러 지도 애플리케이션에서 구현하여, 관광용 지도의 기능을 효과적으로 향상시킴을 보인다.

핵 심 낱 말 적응형, 격자, 대화형, 지도, 가시화

Abstract

In tourism studies, it is considered that a tourist map has three basic functions: discover potential points-of-interest (POI), plan routes and give spatial information. In the age of ubiquitous mobile devices and digital maps, it is essential that these functions be implemented in modern maps in addition to dynamic and interactive views customized to users. Additionally, increasing trends of online map personalization demands real-time task-dependent map representations enabled by map deformation. Prior map deformation techniques mainly based on uniform grids can be insufficient for such real-time demands. For higher flexibility of allocating resolutions, using an adaptive approach is needed. An initial investigation on adaptive approach with deforming models is performed in the context of view-dependent rendering integrated with occlusion culling for large-scale crowd scenes. To provide varying resolutions on each articulated model, it is proposed to use a cluster hierarchy in the view-dependent representation. The cluster hierarchy, constructed by adaptively clustering regions with similar significance values in terms of simplification quality, results in a high quality simplification that can achieve interactive performance. Lessons from deforming models are applied to map deformation. A proposed content-aware non-uniform grid leads to adaptive resolutions on the map deformation. In other words, finer subdivisions on more significant regions such as guide road patterns, and coarser subdivisions with less quad edges on less significant

regions are applied. This adaptive approach results in both higher road deformability and performance. Finally, a dynamic and interactive tourist map application is demonstrated. An interactive framework that holistically combines presentations of a POI map and a metro network, helping users identify popular POIs based on visual worth computation, and discover POIs within reach of a metro. Octilinear layouts are used to highlight the metro network, and representative POI images are shown in layout space visualized within a user-specified viewing window.

Keywords Adaptive, grids, interactive, map, visualization

Contents

List of Tables

List of Figures

Chapter 1. Tourist Map Visualization

1.1 Introduction

For travelers, tourist maps are essential guides in unraveling a city. It gives an overview of a city's unfamiliar features by showing travel destinations and major thoroughfare. Traditionally, tourist maps are in a booklet format that is conveniently provided at stations or lodgings for presenting travelers a pocket guide and introduction to a city.

In a recent tourism study [2], it is considered that a tourist map helps three things: discover potential attractions, plan routes, and give spatial information. A tourist map influences the activities that travelers could do by controlling aspects of its objectives. Published maps are based on various approaches whose effectiveness differs by catering to objectives. In this way, paper tourist maps have been a utility for travelers, but its static nature shows its limitations. With increase of ownership of mobile devices, travelers have learned to utilize general purpose digital maps to assist them in their travels, yet these digital maps have yet to fully address the tourist map objectives.

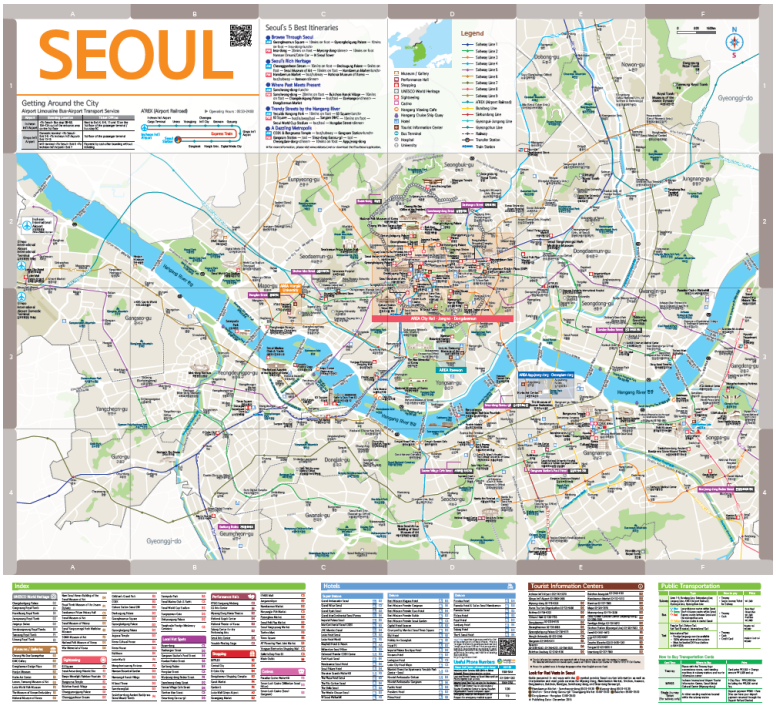


Figure 1.1: [Seoul paper tourist map (http://www.visitseoul.net/)]

The ubiquity of mobile devices supports the wide spread of web services or apps utilizing the geo-spatial and social networking information. The data gathered can result in real-time changes in trends that can only be shown in dynamic representations. In this way, digital tourist maps can reflect time and location based information that enables the user to get personalized recommendations to attractions, routes and information. This article explores approaches to enhance tourism experiences by dynamic updating and adding interactivity to maps. We also cover approaches that augment the experience by highlighting helpful information through personalization, enhancing readability of map annotations, or deforming geometries.

Before going in depth, we would like to recall the contrasting properties of paper and digital tourist maps. Paper tourist maps are usually expandable-as-unfolded brochures that have a large display area for an overview map, close-up of smaller important areas, and a list of points-of-interest. Digital tourist maps can display the said paper map features too, yet in a smaller display area. In contrast to a static paper tourist map, dynamic digital maps make up for a small display area by a combination of basic functionalities like interactive zooming and panning, and level-of-detail hierarchies for visualization.

1.2 Map Information

As more map data have become more available and portable screens get ubiquitous, displaying digestible map information has also become a challenge. There are ways to display useful relevant information without presenting an overwhelming amount of data. Personalization of data can prioritize which information is useful to a specific user. Decluttering can filter out unnecessary elements and annotations in a map. Focus+context, a technique where significant elements are highlighted against a background of low-significance elements, can help users focus on essential information.

User personalization has been applied in many internet services based on preferences and browsing history. With mobile services increasingly integrating more as part of human lifestyle, movement behaviors of users can also be analyzed and thus be utilized for customizing a unique map information. For instance, Google has employed personalization of its search results based on a user's previous search history and recently applied this approach in their maps [3]. Their maps show a personalized set of landmarks and points-of-interest based on the user's profile.

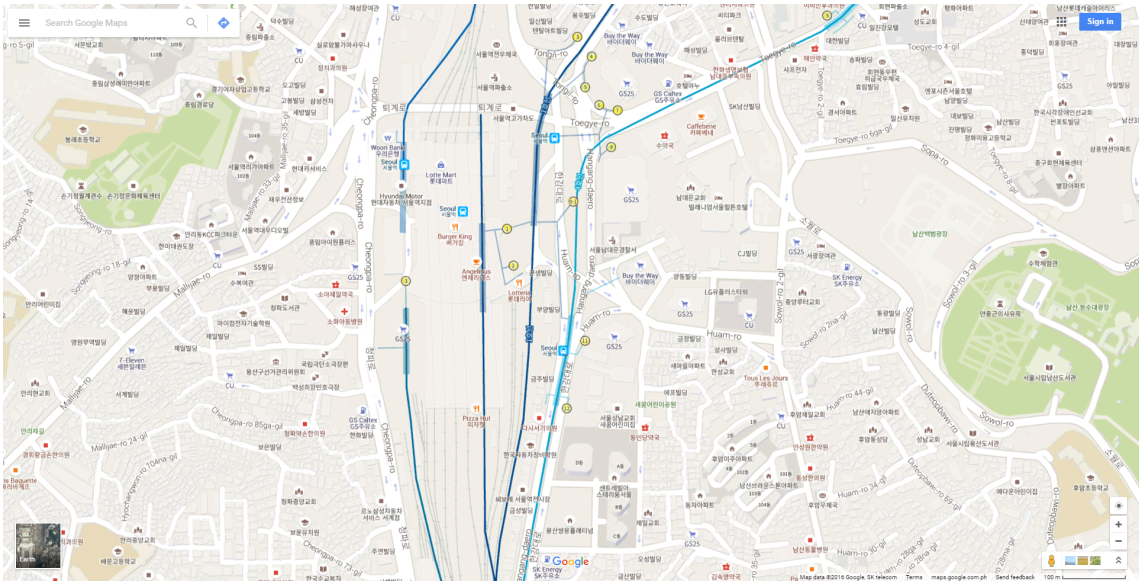


Figure 1.2: [Google Maps showing map information.]

Decluttering a map involves minimizing clutter that occurs when there is too much information in a map. Usually in this approach, elements with lower priority are removed or hidden from the view. In a work by Jaffe et al. [4], decluttering is applied to a map filled with geo-tagged images. They apply a significance function to determine priority and creates a clustering hierarchy to display only the top images in a specified area and view.

Focus+context has help distinguish important regions against a backdrop. A common implementation of this in maps is by applying a fisheye lens [5] to enlarge focus regions by distorting map images. A similar approach

of enlarging focus regions is shown to be applied to vector maps [6] without distorting road edges by defining an optimization framework with edge scaling objectives.

1.3 Discovering Points-of-Interest

Another essential function of tourist maps is showing potential points-of-interest (POIs) of a city. Correct, updated and verified information about POIs would help the tourist in planning a tour. A dynamic map can generate and update information about POIs by using multiple sources including social driven crowd-sources.

In representing POIs in a map, their annotations come in the form of representative text, image, geometry or symbol. The following examples describe approaches in helping users discover POIs in tourist maps.

Grabler et al. [7] describe an approach to automatically generate tourist maps with POIs that pop-out in 3D using POIs' pre-acquired building geometries. In addition, relevant POIs are selected by importance that can be set depending on information (user ratings, genre, etc.) gathered from internet sources or user preference. Following the style of some hand-drawn paper tourist maps, they render a 2D road map in an oblique / perspective projection that highlights the relevant 3D POI models.

A work by Birsak et al. [8] tries to create automatic generation of tourist brochures and put the annotations around map boundaries. Multiple frames of maps are shown wherein a central overview map of the city is with marked areas whose close-up views are shown in smaller frames. Each frame has a set of pins showing relevant POIs in that specific view. A similar method of fetching relevancy of POIs from Internet sources are also applied in this approach.

POI Annotation placement. POI annotations can be placed in a map externally or internally. External annotations are conveniently put around the map boundaries and may be connected by 1) lines to the actual location or 2) respective symbols (number or letter) placed on the location. Internal annotations place them in the vicinity of the sources that allow for instant association of POI to the actual location in the map.

1.4 Route Planning

There are two ways that tourist maps guide users in their route planning: by showing roads in the vicinity of POIs and by showing a transportation network (typically a metro). In dynamic maps, route planning is enhanced by highlighting the road or metro network through deformation of the map. It has been studied that deformation has been used to assist users in improving their cognitive tasks [9] in many different applications, e.g., road network and metro network map generations. Maps can be easily re-generated and customized as desired by using different parameters. In addition, performance improvement of map deformation has given a way to interactive explorations of such maps.

Road Map. In map deformation approaches, road networks are represented as graphs. In most applications, a set of design objectives is defined in order to control characteristics of layouts of road maps, and is modeled into a minimization problem.

In the work of Haunert et al. [6], regions of a road network are scaled to give focus to selected roads. Its road network graph is formulated as a quadratic programming problem. Its follow-up work [10] reformulates the problem as a linear programming problem to achieve interactive rates for small maps consisting of a few thousands of edges.

Agrawala et al. [11] and Kopf et al. [12] describe automatic approaches to generate route / destination maps, i.e., a map summary of directions to a particular point of interest. Given a set of selected roads, an optimized

layout that rescales and reorients road edges is computed in order to increase the readability of these road maps. Similarly, in the work of Lin et al. [13], the goal is to create mental maps, aiming to help easily recall a featured area. The road map is deformed such that certain formations can be recognized from shapes of the streets.

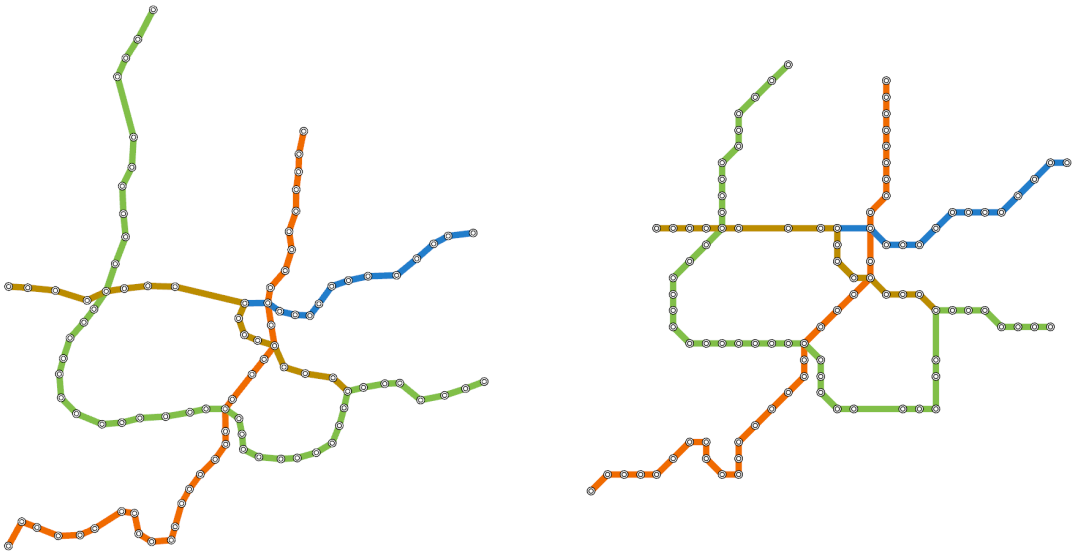


Figure 1.3: [Busan metro map deformation [1].]

Metro Map. Different approaches have been proposed for computing layouts for metro network [14]. Among them, octilinear layouts using diagonal edges as well as horizontal and vertical edges have been widely adopted for visualizing metro networks.

Automatic generation of octilinear metro map layouts has been studied in recent years. One of the initial works was of Hong et al. [15]. This approach utilizes mass spring techniques, where forces dictate relative positions to achieve octilinear layouts. Stott et al. [16] used a hill climbing algorithm to optimize the positions of the stations. Nollenburg et al. [1] viewed the layout problem as an optimization problem, and employed a mixed integer programming (MIP) approach to find a global solution maximally satisfying design constraints. Its resulting layouts show a strict compliance to octilinearity and feature almost uniform spacing. Similar MIP techniques have been used in related layout problems [17].

These introduced metro map layout generation techniques can then be utilized to generate a metro-centric tourist map. In this way, POIs near metro stations are connected by a line to annotations of POI which are placed externally[18] or internally[19, 20].

1.5 Motivation

As mentioned before, paper tourist maps have limitations in size, and being static. As more and more tourists travel along with mobile devices, it follows that more would utilize digital maps that are interactive. These digital maps provide more dynamic content with the rise of mobile services and social media apps. These can lead to updated information and personalization of maps. Yet currently general purpose digital maps like this do not consider about the functions of the tourist maps.

Along with providing specific functional views of maps comes providing transitions from one map to another. Several works [21, 22] use map morphing in order to assist the users in creating smooth transitions that help in mapping points from one map into another. However, these works operate in the image space that could result in

drastic deformations of map elements. Recently, vector-based maps undergo deformation in order to create map representations. Early works [6, 10] apply deformation on road edges. The challenge of deforming roads is scaling the performance as the number of road edges increases. An improvement on this approach[13] uses uniform grids as a medium to deform the road maps. Since the uniform grid requires less edges to deform, the performance of the uniform grid approach is faster than deforming on the road edge level directly.

Uniform grids has been studied a lot in GIS and graphics[23] [24] fields. While a uniform grid is simple to implement, determining the adequate resolution for balancing accuracy and speed is not straight-forward. A well-known approach to this is by using adaptive grids: more significant regions are given finer resolutions while less significant regions are given coarser resolutions. As a result of adaptive resolutions, an expected increase of performance can be achieved since less edges are used for similar accuracy levels.

Adaptive grids has been studied in other domains, however in map deformation there has been no known application of adaptive grids. A challenge of using this approach has been mentioned[25] in that shorter grid edges can cause non-optimal placement of neighbor points.

In this paper, the use of adaptive grids is explored in optimizing the performance of interactive applications. Given a scene, the deformation is optimized by applying adaptive grids that compute the appropriate level of detail for the given significance parameters. With less elements involved for processing the resulting effect is higher performance compared to a uniform grid.

1.6 Scope

The area of tourist map representation covers much ground. There are multiple ways to enhance the efficiency of interactive maps including line simplification, map generalization and adaptive deformation. This work focuses on the adaptive deformation.

Adaptive grids applied to map layouts is investigated in this dissertation. Specifically, by applying adaptive grids to efficiently deform maps to create tourist map layouts. It is not investigated the effective use of the resulting map layouts since those have been investigated by the related works. Achieving efficient deformation is the focus of this work.

The transportation network chosen in this work is limited to the metro network given that it is a prominent feature in city navigation. However this can be extended to include other public transportation networks. This could also be augmented by other factors such as traffic, taxi availability and transportation schedules.

This work employs the use of a quad grid mesh instead of a triangle mesh. The adaptive concepts can be also applied in a triangle mesh as demonstrated in image deformation[26]. However, as observed in vector deformation[13], inconsistent orientations of triangles results in inconsistent warping of road deformations, which would lead to higher residues in optimization.

1.7 Contributions

This dissertation proposes to improve tourist map functionalities in a digital map by adding dynamic and interactive features. This can be achieved through the use of adaptive grids that enable fast deformations.

1. Adaptive techniques from computer graphics are used to improve efficiency, as investigated by applying adaptive simplification in animated models in crowd rendering. A cluster hierarchy is proposed that acts as a multiresolution representation and a bounding volume hierarchy.

2. As applied to map deformations, proposing the use of adaptive grid subdivision improves performance from the decrease of the number of grid edges to deform. The challenge of placing of points is addressed by use of regular trees and triangulation of smaller quads resulting to the effect of preserving the shape of the adaptive grid comparable to the uniform grid.

3. An interactive application demonstrating a novel holistic approach of complying with tourist functions of showing tourism, map and route information is implemented and studied. Adaptive scaling of edges is applied to enables a focus+context effect displaying significance of regions.

The following chapters based off the author's previous work[27, 20, 28] has investigated the use of adaptive approaches that efficiently visualizes data. In Chapter 3, initial investigation on view-dependent adaptive rendering of 3D deforming models is done. In this proposed approach, applying simplification based on significance allows the rendering performance to increase. Lessons from this work are then applied to map deformation. In Chapter 4, to enable interactive transitions of changing map views and representations, a fast content-aware non-uniform grid is proposed that adapts resolutions by referring to the significance function as to minimize elements involved in map deformations. In Chapter 5, a demonstration of a metro-centric POI discovery application that automatically generates tourist maps is proposed where spatial information of tourist hot spots are determined by significance function based on factors (e.g. tourist density, popularity, metro proximity) and users are assisted in route planning by attaching POI annotations to metro stations. Finally, the limitations, summary and future directions are discussed in the concluding chapter.

Chapter 2. Related Work

In this section, prior methods directly related to our approaches are discussed.

2.1 Adaptive Rendering

Adaptive rendering techniques in graphics involve minimizing the number of elements rendered on the screen in order to gain performance.

2.1.1 Mesh Simplification

Mesh simplification has been extensively studied for polygonal meshes, and numerous techniques have been presented. Among them the quadric error metric (QEM) and edge collapse operations [29] have been most widely used for high-quality mesh simplifications. Many simplification techniques have been also proposed for dynamic models, but some of them [30, 31] are not directly applicable to articulated models that are animated with an underlying deformation model.

Simplification for articulated models. Mohr and Gleicher [32] applied the QEM method for simplifying articulated meshes. Their method takes a skinned mesh and a series of example poses. It computes the quadric error for each vertex by considering all the example poses. DeCoro and Rusinkiewicz [33] improved this method by considering the bone transformations and computing the quadric error in a base reference pose. Recently, Landreneau and Schaefer [34] perform the simplification by considering weights associated with vertices and thus achieve higher simplification quality over other techniques. Our simplification method is based on the work of DeCoro and Rusinkiewicz [33], but can be improved by adopting techniques proposed by Landreneau and Schaefer [34]. In addition, Pilgrim et al. [35] proposed a progressive skinning technique that progressively uses a subset of original bones of articulated models. This technique can be combined with our method for higher rendering performance.

Image-based simplification methods. As an alternative representation for polygonal representations, image-based representations [36, 37] like impostors have been proposed for articulated models and reported to achieve a high rendering performance, while maintaining a reasonable memory requirement [38]. As downsides, these techniques may provide low quality rendering results for certain views (e.g., overhead views) or require a high storage requirement. In addition, image-based representations may provide low-quality results for various geometric operations such as collision detection. Nonetheless, these image-based representations can be used together with polygonal representations including ours, to achieve a higher rendering performance and quality. For example, one can use polygonal representations in a near field and use imposters in a far field, as suggested by Kavan et al. [37]. As a result, our method is orthogonal to image-based techniques in the context of rendering.

2.1.2 View-Dependent Rendering and Culling

View-dependent rendering (VDR) uses lower resolutions for portions of the mesh that are located farther away from the viewer. This technique aims to reduce the number of rendered triangles of complex models depending on viewing configurations and has been extensively studied for polygonal meshes [39]. VDR originated as an extension of the progressive mesh (PM) that is a linear sequence of encoding finer meshes [40]. Hoppe [40]

improved this method by organizing the PM as a vertex hierarchy instead of a linear sequence. The vertex hierarchy is known to provide very smooth LOD transitions, while requiring high runtime computations. This technique has been extended to large-scale polygonal models that consist of hundreds of millions of triangles [41]. However, VDR has not been applied widely to articulated models for large-scale crowd rendering.

Visibility culling also has been well studied to reduce the number of rendered triangles in scenes that have a high depth complexity [42]. For general environments, image-based occlusion representations are widely used, and high-performance culling algorithms use GPUs to perform occlusion culling [43].

2.2 Adaptive Maps

Adaptive techniques in maps involve minimizing the number of elements rendered on the screen in order to gain performance and readability.

2.2.1 Map Deformation

Deformation has been used to assist users in improving their cognitive tasks [9] in many different applications, e.g., road network, cartograms, and mental map generations. Maps can be easily re-generated and customized as desired by using different parameters. In addition, performance improvement of map deformation has given a way to interactive explorations of such maps. We discuss techniques related to map deformations used for different, but related applications below.

In map deformation approaches, road networks are represented as graphs. In most applications, a set of design objectives is defined in order to control characteristics of layouts of road maps, and is modeled into a minimization problem.

In the work of Haunert et al. [6], regions of a road network are scaled to give focus to selected roads. Its road network graph is formulated as a quadratic programming problem, where its optimization is performed at *road edge-level*. Its follow-up work [10] reformulates the problem as a linear programming problem to achieve interactive rates for small maps consisting of a few thousands of edges.

Agrawala et al. [11] and Kopf et al. [12] describe automatic approaches to generate route/destination maps, i.e., a map summary of directions to a particular point of interest. Given a set of selected roads, an optimized layout that rescales and reorients road edges is computed in order to increase the readability of these road maps. Similarly, in the work of Lin et al. [13], the goal is to create mental maps, aiming to help easily recall a featured area. The road map is deformed such that certain formations can be recognized from shapes of the streets. Predefined control points are used to guide the final position of vertices. To achieve the goal, a uniform grid is overlaid on the map, which is used as a deformation medium – the linear programming optimization is done on *grid-level*, which enables higher performance than the edge-level approaches.

In Bouts et al. [25], contiguous area cartograms are generated from attributes of a geographic location. A uniform grid is used to overlay the map, and attributes are used to affect the deformation to show dissimilarity of attributes of a location. Similarly, Claudio et al. [20] used tourist destination data as attributes to deform and highlight popular regions.

As discussed above, uniform grids with grid-level optimization have been commonly adopted for fast map deformation. While this approach is simple, its time complexity goes higher as we have higher resolution for higher deformation quality. Our work extends uniform grid-level approaches by using a non-uniform grid that adaptively allocates edges. Because the optimization running time is commonly described by a function of those edges, our adaptive approach can improve the running time and even quality.

2.2.2 Adaptive Approaches

Adaptive Mesh in Graphics. In computer graphics, utilizing non-uniform grids and meshes is commonly adopted to achieve high quality simulation with a lower polygon count than regular structures (e.g., uniform grids). These adaptive approaches generate denser regions in interesting areas and coarser regions in uninteresting areas [44]. However, using irregular structures for optimization problems can pose stability issues and tend to require complex theory and implementations. In similar reasons, using irregular meshes has not been widely tested for our problem, map deformation. Specifically, Lin [13] found that a grid mesh produces better deformations than that of a triangle mesh thanks to the regular structure of the grid mesh for map deformations.

Adaptive grids in GIS. In the field of GIS (Geographic Information System), different variants of regular structures have been actively studied [23]. The quadtree, a type of non-uniform grids, has been well studied; a good survey by Pajarola et al. [23] details the developments. In particular, we mention a specific type called the restricted quadtree, wherein nodes are allowed to differ in size by one subdivision level [45]. A restricted quadtree adaptively samples the space and balances tree depths locally, leading to a smoother transition in terms of map deformation. However, to the best of our knowledge, only a recent map deformation work [25] considered non-uniform grids, but decided to use the uniform grids, since they found no benefit as they treat the grid and map as independent entities.

In our approach, we take advantage of the geometry extracted from the map in order to create an adaptive grid that increases the performance of deformation. In creating an adaptive grid, quads that have significant areas are subdivided in order to give more precision in deforming those areas. To preserve shapes of quads and avoid various deformation artifacts, we introduce support edges, quadtree-like triangulation, but only applied to coarse quads with adjacent smaller finer quads, which are included in the deformation optimization.

Adaptive Simulation. In cloth simulation, cloth is typically modeled as a triangle mesh and forces between vertices are computed to determine the positions. A good survey on the methods is described[46]. Adaptive techniques have been proposed to improve performance [47, 48]. A more recent method by Lee[49] describes an adaptive mesh approach that identifies smooth regions that can be represented with lower resolutions while dynamic regions are represented by higher resolution in order to preserve the details of cloth simulation.

Similarly, we identify less significant regions that can be represented with lower resolutions and more significant regions that can be represented with higher resolutions. We apply this in a road map network deformation in order to achieve the same effect of higher performance while retaining accuracy of the model.

2.3 Map Layout

The related map layout techniques described below help create readable informative maps by controlling the placement and number of information displayed.

2.3.1 Metro Network Layout

Different approaches have been proposed for computing layouts for metro network [14]. Among them, octilinear layouts using diagonal edges as well as horizontal and vertical edges have been widely adopted for visualizing metro networks.

Octilinear layouts. Automatic generation of octilinear metro map layouts has been studied in recent years. One of the initial works was of Hong et al. [15]. This approach utilizes mass spring techniques, where forces dictate

relative positions to achieve octilinear layouts. Stott et. al. [16] used a hill climbing algorithm to optimize the positions of the stations.

Nollenburg et al. [1] viewed the layout problem as an optimization problem, and employed a mixed integer programming (MIP) approach to find a global solution maximally satisfying design constraints. Its resulting layouts show a strict compliance to octilinearity and feature almost uniform spacing. Similar MIP techniques have been used in related layout problems [17]. Our work also utilizes MIP based optimization for computing a global solution given our design criteria.

Annotation placement. Map annotation has been well investigated, yet annotation placement in octilinear metro maps is another problem raised with automated octilinear layout generation. Wu et al. [18] used flow networks to put external annotations around the map boundaries connected by lines to the stations, while avoiding intersections with metro lines. More recently, Wu et al. [19] investigated internal annotations: placing them in the vicinity of the sources; an MIP framework proposed by Nollenburg et al. [1] is utilized to place annotations without overlaps, while drawing octilinear line connectors. Although the resulting internal annotations may not retain relative topology, the spacing is well distributed thanks to the additional aesthetic criteria such as alternating distribution and closed region avoidance. Nonetheless, these techniques are not designed for interactive tour planning and computing representative POIs.

Map warping. Jenny et al. [50] used map warping, Helmert transformation, to analyze geometric distortions of maps, and analyzed octilinear metro maps compared to its geographic counterpart. Bottger et al. [22] used distortion by moving least squares to fit a street map in an octilinear metro map. We also apply map warping techniques for aligning POIs in the computed octilinear layout. A simple map warping method like Jenny et al. suffices for our purposes of transforming point positions. Yet more sophisticated methods exist such as Bottger et al., which can be substituted in order to perform non-overlapping transforms.

2.3.2 Removing Clutters

Metro maps can be populated with many associated attributes such as labels, pictures, etc. Focus+Context and computing representative images have been studied to optimize the display of these annotations.

Focus+Context. Focus and context is one of the staple techniques in the visualization community. This technique magnifies graphs and grids to focus on certain areas, while retaining the normalcy of the remaining areas as a context [51]. In particular, Sarkar and Brown [52] defined a visual worth attribute for each vertex to determine its importance in visibility. In the work of Wang et al. [53], as applied in metro maps, visual worth are set to be higher for edges in the selected route.

Representative images. In a map featuring hundreds of photos, a representative image is usually used to represent an area, as commonly applied in geo-tagged image websites (e.g., *flickr.com* and *panoramio.com*). An approach by Jaffe et al. [4] is to display summaries of a large collection of images by traversing a hierarchy resulting from the Hungarian hierarchical clustering method[54]. The representative images of a cluster are defined by their scores coming from criteria such as relevance. We find the Hungarian method useful as a hierarchical clustering method, which creates a hierarchy handily used as an LOD tree. At the same time, this method does not need a defined number of clusters as input. These techniques have not been applied to city tour planning, the main application of our method.

Chapter 3. VDR-AM: View-Dependent Representation of Articulated Models

3.1 Introduction

Owing to advances of data capture and modeling technologies, detailed articulated models are easily generated and widely used in many different applications. Moreover, various crowd simulation techniques have been designed and a high number of articulated models are frequently used in large-scale crowd scenes [55, 56]. In typical crowd scenes with lots of articulated characters, it can require high computation costs of rendering and performing other operations (e.g., collision detection) for handling those articulated characters.

A significant amount of research has been put in order to improve the performance of rendering and conducting various operations for polygonal models. Some of them include designing multi-resolution representations of polygonal models [57, 39], performing visibility culling [42], collision detection [58], etc. Unfortunately, most of these prior techniques assume polygonal models and do not easily apply to articulated models. This is mainly because articulated characters are dynamically animated with an underlying deformation model (e.g., skeleton) that changes the shapes of characters.

In terms of rendering articulated models, many techniques have been proposed to improve the rendering performance. At a high level, these techniques can be classified as image-based [36, 37] and mesh simplification approaches [32, 33, 34] for articulated models. Image-based approaches have been reported to achieve a high rendering performance by rendering textures instead of triangles. However, these techniques may provide low quality rendering results for certain views (e.g., overhead or near views [36, 37]) or may not be used for other geometric applications such as collision detection.

Geometric simplification methods for articulated models, on the other hand, can provide high-quality polygonal meshes that can be used for various views. However, these simplification methods have focused on computing different versions of level-of-detail (LOD) meshes from an input articulated mesh, and have not been applied widely to view-dependent rendering that uses different LODs for portions of the mesh depending on viewing information. To the best of our knowledge, there have been no prior methods that adopt view-dependent rendering integrated with culling for large-scale scenes consisting of hundreds or thousands of articulated models [59].

Main contributions. In this paper we propose a novel, View-Dependent Representation of Articulated Models, VDR-AM. We show its benefits in the context of rendering, especially view-dependent rendering integrated with occlusion culling. VDR-AM consists of a cluster hierarchy that serves as both a multi-resolution representation and a bounding volume hierarchy. We use its multi-resolution representation for view-dependent rendering, and its bounding volume hierarchy for occlusion culling. We construct each cluster of the cluster hierarchy to contain a spatially-coherent small portion of the mesh that also has similar simplification errors. To construct such clusters, we propose an error-aware clustering method. Given the cluster hierarchy of VDR-AM, we can compute a LOD cut that satisfies a user-specified screen space error in terms of pixels-of-errors (PoE) at runtime. We also perform occlusion culling for clusters in the LOD cut in an efficient manner that utilizes the temporal coherence between consecutive frames.

We have implemented our method and applied it to three large-scale crowd scenes that consist of up to five thousand articulated models that have 242 M triangles in total. Even though our VDR-AM representation is not mainly designed for static polygonal models, it can be naturally applied to handling those models without major modifications. Therefore, we have also applied our representation even for static models that are parts of tested

Figure 3.1: [This figure shows two images of an exhibition crowd scene that has 1 K articulated characters. All the characters and the scene have 83 M triangles. Our method achieves 46 frames per second (fps) on average for this model with 0.5 pixel-of-error (PoE) in a 1280 by 720 HD screen resolution.]

crowed scenes. Our method shows 16 to 50 frames per second (fps) without visible artifacts in a 1280 by 720 HD screen resolution. Compared with a base rendering method that uses the original articulated models combined with view-frustum culling, our method achieves four to eight times improvement.

3.2 Related Work

In this section we give a background on animating articulated models based on skinning, and briefly review previous work related to our method. For detailed information about crowd rendering in general, refer to an excellent survey by Ryder et al. [59].

3.2.1 Background of Articulated Models

Articulated models are typically designed with skinning and skeleton animations. In this case an articulated model consists of a base mesh, a skeleton, and vertex weights. The base mesh, also known as skin, is a 3D polygonal mesh, and the skeleton is a hierarchical representation of bones. The vertex weights associated with vertices of the base mesh represent the skin-to-skeleton binding. One of the most popular techniques used to achieve interactive animation is linear blend skinning.

The animation of the base mesh of an articulated model is defined by a series of poses. Each pose is defined by a 4 by 4 transformation matrix representing positions and orientations of bones of the skeleton. A vertex position, v, in the base mesh is then moved to a new position, $\hat{v}$, by linear blend skinning with a pose, based on the following equation:

$$\hat{v} = \sum_i w_i M_i v, \tag{3.1}$$

where M_i is the transformation of the ith bone associated with the vertex with the weight w_i given the input pose. There are more advanced skin blending methods such as skinning using the dual quaternions, and they can be also used with our method.

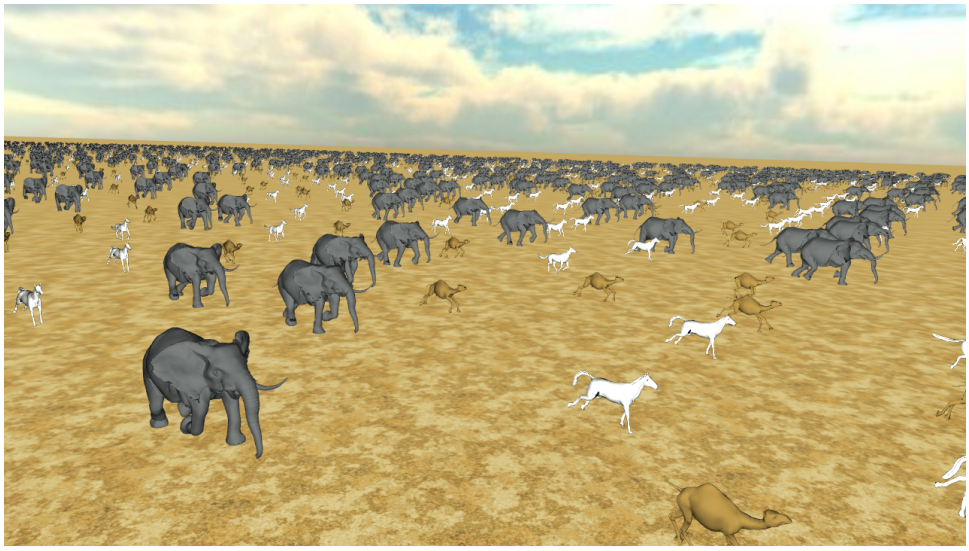


Figure 3.2: [This figure shows a stampede crowd scene that has 5 K articulated characters and 242 M triangles. Our method can achieve 16 frames per second (fps) on average for this large-scale crowd scene.]

3.2.2 Mesh Simplification

Mesh simplification has been extensively studied for polygonal meshes, and numerous techniques have been presented. Among them the quadric error metric (QEM) and edge collapse operations [29] have been most widely used for high-quality mesh simplifications.

Many simplification techniques have been also proposed for dynamic models, but some of them [30, 31] are not directly applicable to articulated models that are animated with an underlying deformation model. In this section we focus on simplification methods that can handle articulated models with skinning.

Simplification for articulated models. Mohr and Gleicher [32] applied the QEM method for simplifying articulated meshes. Their method takes a skinned mesh and a series of example poses. It computes the quadric error for each vertex by considering all the example poses. DeCoro and Rusinkiewicz [33] improved this method by considering the bone transformations and computing the quadric error in a base reference pose. Recently, Landreneau and Schaefer [34] perform the simplification by considering weights associated with vertices and thus achieve higher simplification quality over other techniques. Our simplification method is based on the work of DeCoro and Rusinkiewicz [33], but can be improved by adopting techniques proposed by Landreneau and Schaefer [34]. In addition, Pilgrim et al. [35] proposed a progressive skinning technique that progressively uses a subset of original bones of articulated models. This technique can be combined with our method for higher rendering performance.

Image-based simplification methods. As an alternative representation for polygonal representations, image-based representations [36, 37] like impostors have been proposed for articulated models and reported to achieve a high rendering performance, while maintaining a reasonable memory requirement [38]. As downsides, these techniques may provide low quality rendering results for certain views (e.g., overhead views) or require a high storage requirement. In addition, image-based representations may provide low-quality results for various geometric operations such as collision detection. Nonetheless, these image-based representations can be used together with polygonal representations including ours, to achieve a higher rendering performance and quality. For example, one can use polygonal representations in a near field and use imposters in a far field, as suggested by Kavan et al. [37]. As a result, our method is orthogonal to image-based techniques in the context of rendering.

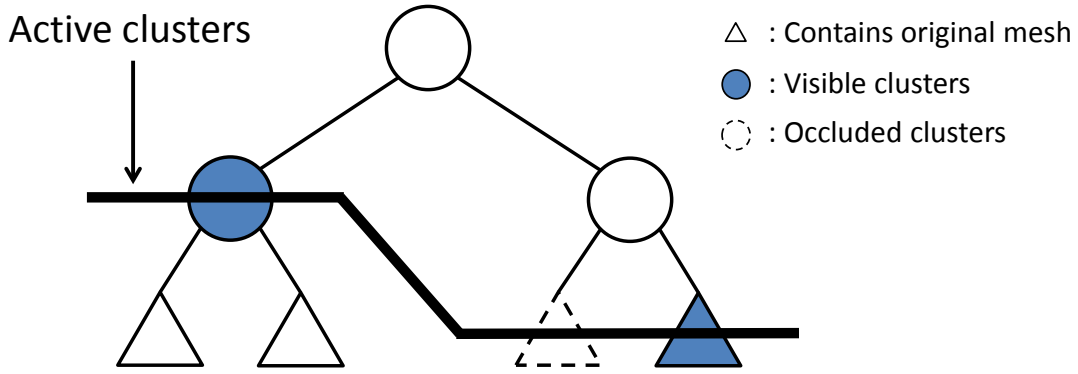


Figure 3.3: [This figure shows our cluster hierarchy of our VDR-AM representation with active clusters (i.e. LOD cut) including visible and occluded clusters. Note that leaf clusters contain the original mesh.]

3.2.3 View-Dependent Rendering and Culling

View-dependent rendering (VDR) uses lower resolutions for portions of the mesh that are located farther away from the viewer. This technique aims to reduce the number of rendered triangles of complex models depending on viewing configurations and has been extensively studied for polygonal meshes [39]. VDR originated as an extension of the progressive mesh (PM) that is a linear sequence of encoding finer meshes [40]. Hoppe [40] improved this method by organizing the PM as a vertex hierarchy instead of a linear sequence. The vertex hierarchy is known to provide very smooth LOD transitions, while requiring high runtime computations. This technique has been extended to large-scale polygonal models that consist of hundreds of millions of triangles [41]. However, VDR has not been applied widely to articulated models for large-scale crowd rendering.

Visibility culling also has been well studied to reduce the number of rendered triangles in scenes that have a high depth complexity [42]. For general environments, image-based occlusion representations are widely used, and high-performance culling algorithms use GPUs to perform occlusion culling [43].

Hybrid algorithms for rendering acceleration. Many hybrid algorithms [60, 41, 61] that combine simplification with visibility culling for static polygonal models have been proposed. These techniques have integrated various visibility techniques into VDR frameworks of static polygonal meshes. Our VDR-AM representation can enable hybrid rendering methods for articulated models.

3.3 Overview of Our Method

In this section we explain our representation, followed by an overview of its construction and our runtime rendering algorithm.

3.3.1 Dual Representation

We use a *cluster hierarchy* (Fig. 3.3) for an articulated model, as a dual representation. The cluster hierarchy serves both as a multi-resolution hierarchy for View-Dependent Rendering (VDR), and as a bounding volume hierarchy for occlusion culling, view-frustum culling, and other geometric operations.

We call each node of the hierarchy a *cluster*. Each cluster serves as a main processing unit for VDR and occlusion culling. Each leaf cluster of the hierarchy contains a portion of the base mesh of the articulated model. For an intermediate cluster of the hierarchy, we merge two sub-meshes contained in its two child clusters, simplify them, and store them in the intermediate cluster. Therefore, each leaf cluster can provide the original resolution of

a portion of the mesh, while an intermediate cluster can provide a low resolution for a portion of the mesh. Also, each cluster is associated with a bounding volume that contains all the geometry throughout the animation.

Each cluster records its maximum geometric simplification error that is caused by simplifying the sub-mesh of the cluster, while considering poses of an animation for the articulated model. In order to allow drastic simplifications on the articulated model given an error bound, it is critical to cluster vertices that have similar simplification errors, thus leading to smaller geometric simplification errors for each cluster. In order to provide a high culling ratio, each cluster should be constructed such that it contains a spatially coherent portion of the base mesh of the articulated model.

3.3.2 Construction

In order to construct the cluster hierarchy of an articulated model, we first decompose the base mesh of the model into clusters, which become the leaf clusters of the hierarchy. We perform our error-aware clustering method to construct each cluster to have a spatially-coherent portion of the mesh whose vertices have similar simplification errors (Sec. 3.4.2). For the simplification of sub-meshes contained in clusters, we use the well-known edge-collapse and quadric-based simplification methods, which also consider poses of the animation of the model. Since the simplification process can take a large amount of time for the animation that consists of many poses, we propose a pose selection method that chooses representative poses for the animation and simplify the mesh by considering only those representative poses (Sec. 3.4.1).

3.3.3 Rendering Algorithm

To show benefits of our VDR-AM representation, we apply it to view-dependent rendering integrated with occlusion culling for articulated models. At runtime, we compute a new position and orientation (e.g., animation pose) of each articulated model in the scene. To perform VDR, we maintain *active clusters* (i.e. a LOD cut) that represent the articulated model with the lowest number of triangles, given a user-specified error bound (Sec. 3.5.1). To determine active clusters, we compute a screen-space projected simplification error for each cluster and compare it with the user-specified error bound. To perform occlusion culling we compute a conservative set of visible clusters based on the bounding volume information encoded in the cluster hierarchy and render them for the final images (Sec. 3.5.2).

3.4 Cluster Hierarchy Construction

In this section we explain our cluster hierarchy construction method. We also compute clusters from the static polygonal models of scenes, as we compute clusters from the animated, articulated models.

Pose space reduction. Since an articulated model can have many bones (e.g., 30 to 60 bones for human-like characters), each pose can be considered as a point in a high dimensional pose space. Unfortunately, any operations on these high dimensional points can be very difficult and expensive because of the well-known curse of dimensionality. To ameliorate this issue, we reduce the dimensionality of the pose space. In particular, we use the multi-dimensional scaling, a statistical tool for reducing dimensions of data [62]. This method maps data into a reduced dimensional space, while preserving the distance between data in the original dimensional space. For all the tested models in the paper, we reduce the pose space of such models to 9 dimensional space, as suggested by Assa et al. [63]. Though Assa et al. suggested mainly for human-like models, we found that 9 dimensional space works well to other tested animal models.

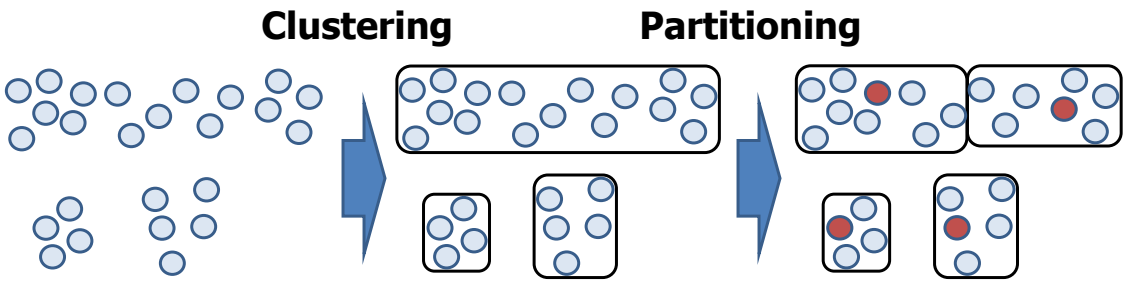


Figure 3.4: [This figure shows the process of our pose selection method that performs the clustering and partitioning. The red points are the selected final poses that represent the distribution of the original poses.]

3.4.1 Representative Pose Selection

In order to reduce the time taken on simplifying models, we propose to consider only *representative poses* instead of considering all the original poses during the simplification process. As the number of representation poses decreases, we can improve the performance of the simplification, but can underestimate the simplification error more, leading to a low-quality simplification.

To minimize this negative effect, we define *pose groups*, each of which contains a coherent set of poses. We then choose a representative pose from each pose group. After reducing the dimension of the pose space, we partition poses into pose groups based on our *clustering and partitioning* framework (Fig. 3.4).

We start with creating a pose group for each pose point. Our pose selection method consists of two stages: clustering and partitioning stages. In the clustering stage, we recursively merge two groups into a new pose group, if any two points chosen from those two pose groups are within the Euclidean distance of d. In order to efficiently perform this operation, we can construct a kd-tree from poses in the reduced pose space and find nearest neighboring pose points [64].

The pose groups computed from the clustering stage can be quite big (e.g., the top group computed from the clustering step shown in Fig. 3.4), since we compute pose groups based only on the local distance information between poses. We adopt a second, partitioning stage that splits big groups into smaller groups. More specifically, we check whether the diagonal size of the bounding box computed from a pose group, c, is bigger than a threshold (e.g., $1.5 \times d$). If so, we recursively split the group c into two pose groups by using a median spatial partitioning, which divides the longest width of the bounding box of the group c into half. For each final pose group, we choose a representative pose (e.g., red circles shown in Fig. 3.4) that is closest to the center of the group.

We tried a well-known clustering method, K-means, for computing representative poses. We chose our clustering and partitioning framework instead of K-means, mainly because it is hard to set the target number of pose groups that well represents the distribution of the original poses for different animation patterns (e.g., walking, running, etc.).

3.4.2 Error-Aware Clustering Method

To construct the cluster hierarchy of an articulated model, we first decompose the base mesh of the model into a set of clusters. These computed clusters become leaf clusters of the cluster hierarchy.

We identify two different criteria and consider them for the cluster construction. First, each cluster should be a spatially coherent portion of the mesh, in order to keep the bounding box of the cluster small and thus achieve a high culling ratio. Second, each cluster should contain vertices that have similar simplification errors. For

Figure 3.5: [This figure shows eight poses chosen out of 35 poses for the walking animation based on our pose selection method.]

Figure 3.6: [This figure shows two images of an office evacuation crowd scene with two hundred articulated characters. This scene consists of 16.4 M triangles. Our method can achieve 49 frames per second (fps) on average for this scene.]

example, if a cluster contains two different mesh regions such as a joint and upper arm, highly deforming and less deforming regions respectively in the animation, then the cluster can get a high simplification error caused by the joint region, even though some portions (e.g., the upper arm) contained in the cluster may have a much lower simplification error. In this case, we may have to render a higher number of triangles even though some of those triangles can be simplified further given the user-specified error bound.

We, therefore, propose an error-aware clustering method that considers the simplification error as well as the spatial coherence for the sub-mesh contained in each cluster. To consider the simplification error for the geometry contained in each cluster during clustering, we have to simplify the mesh and compute simplification errors. However, this causes a chicken-and-egg problem, since we have to construct clusters first before simplifying clusters.

In order to avoid this problem, we measure how much a vertex deforms during the animation as a *deformation level* for the vertex, and use it as a rough approximation of the simplification error for the vertex. This is because as a vertex deforms more, it tends to generate a higher simplification error computed by considering different poses of animations.

To measure the deformation level of a vertex, we identify a bone, b_a, that is a least common ancestor for bones that affect the vertex and then compute a tightest bounding box that contains the trajectory of the vertex during the animation in the reference frame of the bone b_a. The deformation level of the vertex, then, is computed as the

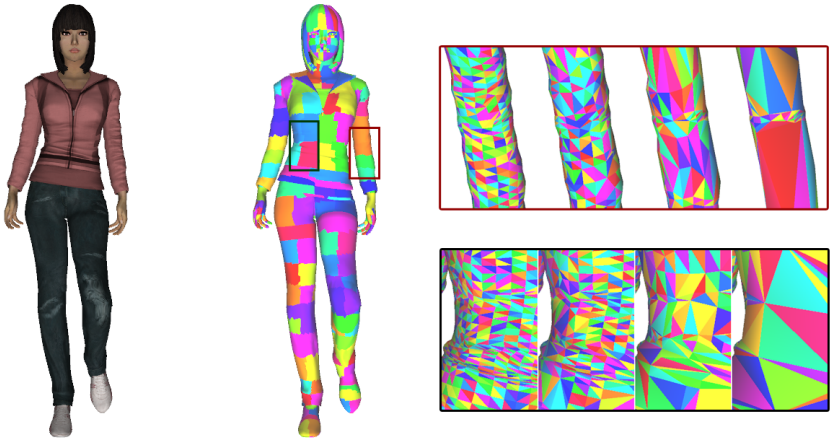


Figure 3.7: [This figure shows an original model (a) and its leaf clusters generated by our error-aware clustering method (b). (c) show zoomed regions of boxed parts in (b). The top of (c) shows a deforming elbow region with increasing simplification levels. The bottom shows a non-deforming torso region, resulting in more aggressive simplification.]

diagonal length of the bounding box. The computed deformation level does not fully capture the simplification error, which is also affected by the neighboring triangles of the vertex. Nonetheless, we have found that it works well for our tested benchmarks and is very easy to compute the deformation level during the clustering stage.

For computing clusters of the base mesh, we adopt a clustering and partitioning framework that is similar to the one we used for computing pose groups. In the first clustering stage, we group adjacent vertices (i.e. spatially coherent vertices) with similar deformation levels into a cluster. We define that two vertices have the similar deformation levels, when their deformation levels differ within the range of 10%. We continue this process until we cannot merge vertices any more.

Clusters computed from the clustering stage can have a very high number of triangles. Therefore, we apply the partitioning stage, which recursively splits clusters that have more than s (e.g., 100) triangles based on the median-based spatial partitioning. Fig. 3.7-(b) shows computed leaf clusters based on our method for an input model.

3.4.3 Hierarchy Construction

We construct the cluster hierarchy in a bottom-up manner, starting from leaf clusters computed in Sec. 3.4.2. We construct the hierarchy by merging two adjacent clusters that have similar deformation levels. We define two clusters to be adjacent, if they have adjacent triangles that share the same vertices. We also define the deformation level of a cluster to be the maximum of the deformation levels of vertices contained in the cluster.

Particularly, among adjacent clusters for a cluster, c, we identify a cluster, c_m, that has the minimum difference of the deformation level to that of the cluster c, and merge c_m and c into their parent cluster. We repeat this procedure until we construct the root node of the cluster hierarchy.

Simplification. Once we construct the hierarchy, we then perform the simplification for each cluster by traversing clusters in a bottom-up manner. For each leaf cluster, we simplify the sub-mesh contained in the cluster such that the number of the triangles in the cluster is reduced into half. For the simplification, we apply the pose-

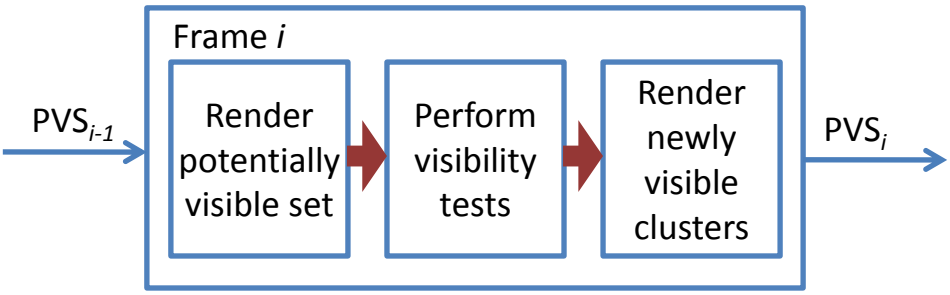


Figure 3.8: [This figure shows our runtime rendering architecture.]

independent simplification method [33] that uses the well-known quadric simplification error metric. During the simplification, we consider only representative poses computed by our pose selection method (Sec. 3.4.1).

3.5 GPU-based Rendering

In this section we explain how we can design an interactive GPU-based VDR method integrated with occlusion culling based on our VDR-AM representations for articulated models.

3.5.1 LOD Selection

In order to perform VDR and culling, we first compute a LOD cut in the cluster hierarchy that represents the articulated model given the error bound. To compute the LOD cut, we measure the geometric simplification error of each cluster in the screen space. To do that, we pre-compute a sphere whose diameter corresponds to the maximum Hausdorff distance between the original mesh and its corresponding simplified mesh contained in each cluster. For efficient computation, the maximum Hausdorff distance is approximated as the square root of the maximum quadric error associated with each cluster. We project the sphere associated with each cluster to the screen space. If the diameter of the projected sphere is equal to or smaller than the user specified pixels-of-error (PoE) value, we treat the cluster to have an enough resolution that can represent the articulated model given the PoE value. This LOD cut selection method takes only a minor CPU time (e.g., less than 1 ms) even for the exhibition crowd scene (Fig. 3.1) that consists of 1 K articulated models and 83 M triangles.

3.5.2 Rendering Algorithm

We use an image-based conservative culling algorithm [43] that is based on the frame-to-frame coherence and hardware-accelerated occlusion queries, in order to achieve an efficient performance of culling. We briefly explain how we enable such image-based culling based on our VDR-AM representation for articulated models.

Before we perform the culling, we decompose active clusters of the hierarchy into two disjoint sets: *potentially visible set* (PVS) and *potential occludee set*. Clusters in the PVS are treated to be visible and are used to create an occlusion map for visibility tests of clusters. We use PVS_i to denote the PVS at frame i. The overall architecture of our rendering algorithm is shown in Fig. 3.8.

Occlusion map generation. At frame i, we start with PVS_{i-1}, the PVS at the previous frame $i-1$. As Step 1 of our algorithm, we first update, i.e. simplify or refine, clusters of PVS_{i-1} to meet the error bound with our LOD selection method (Sec. 3.5.1) that considers the current view information. We set those clusters as PVS_i. We then

Figure 3.9: [The left figure shows the image created at a first person view. The right image shows a third person view with occlusion results. The black lines show the view frustum of the first person view. The pink, blue, and yellow boxes are visible, occlusion culled, and view-frustum culled clusters.]

render clusters of PVS_i into color and depth buffers. The depth buffer information computed with PVS_i serves as an occlusion map for visibility tests in the next rendering step.

Visibility tests. All the clusters of the LOD cut that are not in the PVS_i are used for the potential occludee set. We also update all the clusters in the occludee set. In Step 2 of our rendering algorithm, we check the visibility of clusters of the occludee set by using hardware accelerated occlusion queries. For clusters that passed the standard view-frustum culling, we use the bounding volumes of those clusters with the occlusion queries against the occlusion map. These bounding volumes serve as a conservative proxy to the geometry contained in corresponding clusters. We call those clusters of the occludee set that are identified as visible clusters with occlusion queries *newly visible* clusters. Fig. 3.9 shows an example of culling results in one of our benchmark scenes.

Rendering newly visible clusters. As the final step of our algorithm, we render newly visible clusters, to create the final image. Also, these newly visible clusters are added to the PVS_i. We use the PVS_i for the next frame. Note that by taking advantage of temporal coherence, we only update and reset the PVS every n frames.

Instancing and static models. It is common to use instancing to create large-scale crowd scenes. We also store various data of cluster hierarchies such that we can efficiently utilize the GPU-based instancing. More specifically, we adopt the pseudo-instancing method [65] for our VDR representation. In addition, our VDR-AM representation can be easily applied to handling static models. In this case our method considers only base meshes of static models. As a result, our rendering method can be applied to handling both static and articulated models.

3.6 Results

To show benefits of our representation, we have implemented our construction and runtime rendering algorithms on a 3.00 GHz Intel quad-core PC with a GeForce 8800 GTX GPU that has 768 MB. We store all the transformation matrices of bones of various articulated models with all the poses in a 1 D texture buffer, which is easily accessible in the GLSL vertex shader that implements our runtime skinning method. For all the performance tests, we use the HD image resolution of 1280 by 720. In this image resolution, we use the PoE value of 0.5, in order to avoid visual artifacts to viewers. Since the used PoE causes only sub-pixel errors in the screen space, we

do not perform any expensive geomorphing [40].

Benchmark scenes. We have tested our method with three different crowd scenes that consist of human and animal articulated characters. Our first benchmark represents an office evacuation scenario (Fig. 3.6). This office scene consists of 200 instanced virtual human characters and 16.4 M triangles. Our second benchmark represents an exhibition scenario (Fig. 3.1). This exhibition scene consists of 1 K instanced characters and 83 M triangles. The third scene is a stampede scenario (Fig. 3.2) consisting of 5 K instanced animal characters and 242 M triangles. In the first and second crowd scenes, we use 20 different virtual human characters that have 35 bones and 73 K to 110 K triangles for their base meshes. Also, they are animated by using a walking animation pattern that consists of 35 different poses. In the third crowd scene, we use horse, elephant and camel models that consist of 17 K, 85 K, and 44 K triangles respectively. They have 30 to 40 bones and are animated in a running pattern with 15 to 80 poses.

Pose selection parameter setting. In our pose selection method, the parameter d trades-off between the simplification quality and performance of our overall construction method. If we set d to be too high, we would choose too small number of representative poses, causing faster construction, but leading to a worse simplification quality. On the other hand, if we set d to be too low, we would get the reverse effects: slower construction, but higher simplification quality. Given this trade-off space, we found that the range of 10% to 30% of the diagonal size of the bounding box of the model for d strikes a good balance in our tested benchmarks.

In this setting, for a walking animation that consists of 35 poses, our method selects 8 different poses. For a snake crawling animation with 81 poses, our method selects 13 poses. As a result, our method achieves 3 to 5 times performance improvement for our construction method. Also, in terms of the simplification quality, we found that the RMS distance between the original mesh and the simplified mesh computed only from considering computed representative poses is within 1% to 2% difference to the RMS distance between the original mesh and the simplified mesh computed from considering all the poses; we use the metro tool [66] to measure the RMS distance. Fig. 3.5 shows eight chosen poses out of 35 poses of the walking animation. Note that more poses are chosen during a period that the human character switches his pivoting leg.

Hierarchy construction comparisons. In order to show the benefits of our error-aware clustering method, we additionally implemented a naive clustering method that uses an octree. More specifically, we construct clusters by recursively partitioning the base mesh until the sub-mesh contained in each node of the octree has s triangles. This naive method considers only the base mesh computed from a pose and does not consider any simplification errors. Once we compute clusters from the octree, the hierarchy construction and simplification that we have applied to to our error-aware clustering method are performed in the same manner to the naive, octree-based clustering method.

Compared to this naive method, our error-aware clustering method shows only 4% slower clustering performance at preprocessing, but shows 50% higher runtime rendering performances, by rendering 50% fewer triangles given the same error bound in our tested benchmarks. Since we cluster vertices that have similar simplification errors, we can allow more drastic simplification if possible, over the naive octree-based clustering method. Fig. 3.7-(c) shows our simplification results on different portions of a walking character.

Construction time and memory requirement. We set each cluster to have less than 100 triangles. In this setting, our method creates 1.8 K leaf clusters and takes 19 min in CPU to compute our representation for the biggest virtual character that has 110 K triangles. Also, our representation requires 72 bytes per each triangle for an articulated model. Since the original mesh requires 30 bytes per each triangle, our representation requires 140% more space over the original mesh. Since our representation stores simplified triangles, whose number is similar to the number of original unsimplified triangles, our representation requires at least 100% more space over the original mesh. Therefore, the memory overhead 140% of our representation is not significantly high.

Comparison configurations. We measure the performance of three different methods: 1) a base rendering system,

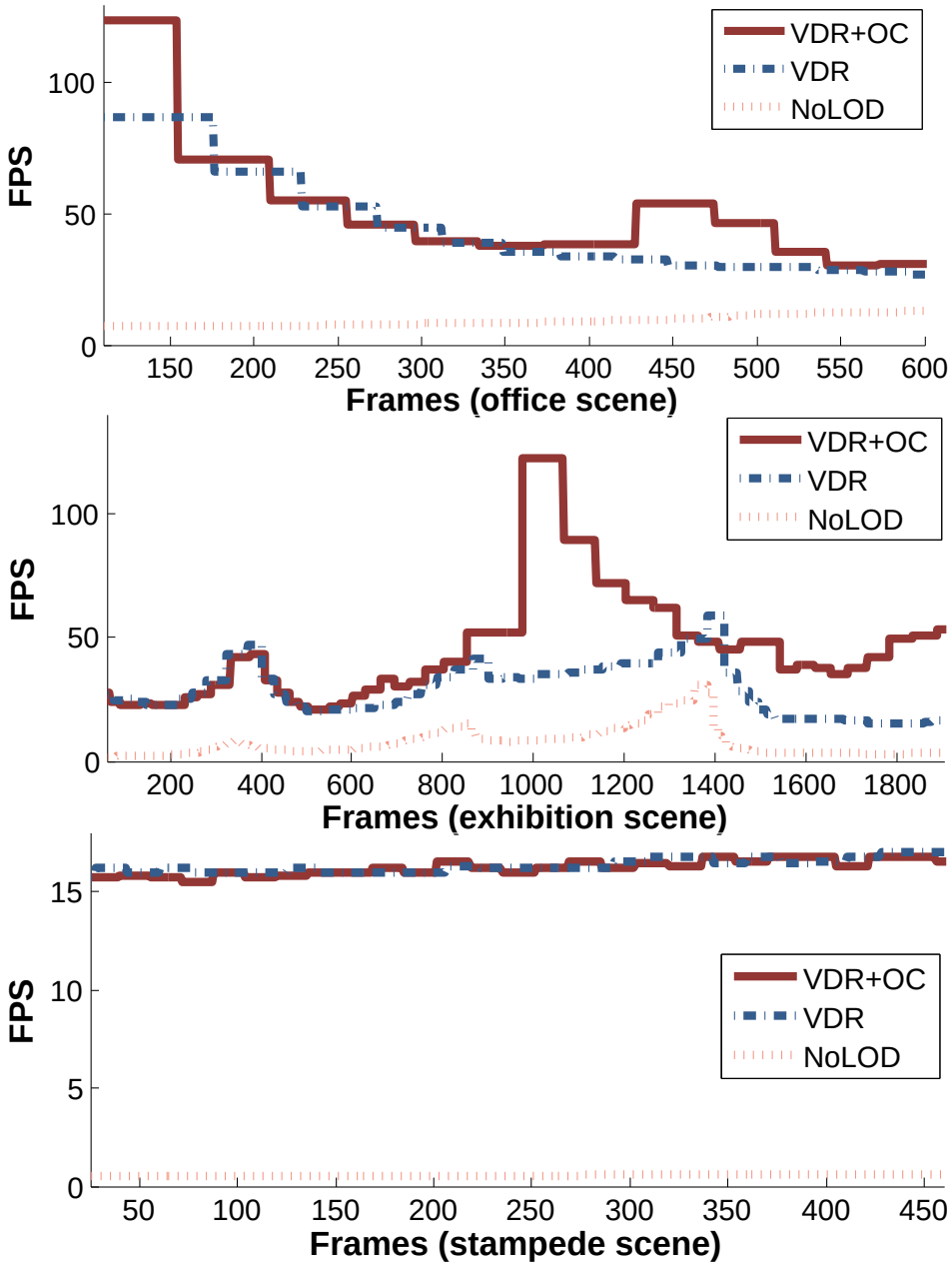


Figure 3.10: [These figures show fps graphs of different methods: our VDR method, **VDR**, our VDR method integrated with occlusion culling, **VDR+OC**, and **NoLOD** that uses the original resolutions and view-frustum culling.]

Benchmark	FPS			#. Rendered Tri. (M)		
	NoLOD	VDR	VDR+OC	NoLOD	VDR	VDR+OC
Office Fig. 3.6	10.27	43.29	49.21	12.46	2.29	2.06
Exhibition Fig. 3.1	7.73	29.15	46.08	24.66	3.14	1.94
Stampede Fig. 3.2	0.55	16.38	16.23	148.09	4.77	4.70

Table 3.1: [This table shows the frames per second (fps) and the number of rendered triangles on average while rendering scenes. **NoLOD**, **VDR**, and **OC** represent rendering with the original resolutions, our view-dependent rendering method, and our occlusion culling technique respectively.]

NoLOD, that uses the original resolutions and performs only view-frustum culling, 2) our VDR rendering system, **VDR**, that performs the VDR on the base system, and 3) our VDR system integrated with occlusion culling, **VDR+OC**. We measure the frames per second (fps) of these methods with pre-defined paths, which are shown in the accompanying video. The fps graphs of these methods with our benchmark scenes are shown in Fig. 3.10.

3.6.1 Rendering Performance

On average, **VDR+OC** achieves interactive performances, 49 fps and 46 fps in the office and exhibition crowd scenes respectively (Table 3.1). In the exhibition scene, compared to the base rendering system, we achieve 4 times performance improvement by enabling the VDR, and achieve 6 times performance improvement by enabling VDR and occlusion culling together. We also observe a similar performance gain with the office scene. These performance improvements are mainly caused by reducing the number of triangles that we have to process for the skinning and rendering operations. More specifically, the base rendering system renders 458 K clusters that contain 24.66 M triangles for the exhibition scene. On the other hand, **VDR** renders 12 K clusters with 3.14 M triangles and **VDR+OC** renders 8 K clusters with 1.94 M triangles.

In the stampede scene our method achieves interactive performance, 16 fps, even though the original model consists of more than 200 M triangles. Also, it demonstrates high performance improvement (up to 30 times) by enabling VDR over the base rendering method. Nonetheless we show a minor, but lower performance by enabling occlusion culling over VDR. This is mainly because we do not have much depth complexity in the tested view point.

3.6.2 Discussions

To highlight the benefit of our method, we show varying resolutions of our view-dependent representation for a snake articulated model that consists of 28 K triangles and 81 poses for its crawling animation (Fig. 3.11). We use the heat color map to show how the resolution of the model varies given the first person view. As a portion of the model is farther away from the viewer, it gets lower resolution as indicated by showing colors close to the blue one. At the given view our method requires only 18 K triangles for rendering the snake model.

Relationships with LODs. Many view-dependent representations have been proposed, as discussed in Sec. 3.2. These techniques have not been widely applied to articulated models. This may be mainly because that many prior view-dependent techniques have high computational overheads. Nonetheless, our technique reduces the computational overhead by providing view-dependent resolution at a granularity of clusters consisting of around

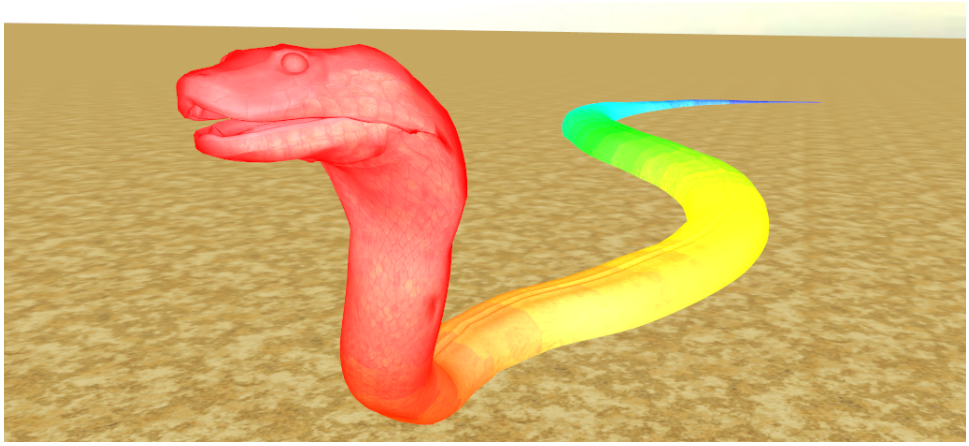


Figure 3.11: [This figure shows adaptively varying resolutions computed from our VDR-AM representation. We render each vertex of the model with colors computed from the well-known heat color map; the red color indicates the highest resolution, while the blue color refers to the lowest resolution.]

100 triangles, not each triangle of the mesh. Note that this kind of approach is inspired by efficient LOD rendering techniques such as HLODs (Hierarchical levels of detail) [67] designed for large-scale static models.

Breakdown of each rendering component. We also measure how much percentage each component of our method takes over the total rendering time. The CPU-based selection of the LOD cut given a viewing configuration takes less than 1.5 ms. The GPU-based skinning, rendering, and occlusion culling components take 14%, 82%, and 4% over the total rendering time on average across all the tested benchmarks.

Limitations. Even though our method shows performance improvements over the base rendering method, there is no guarantee that our VDR integrated with occlusion culling always improves the performance of various crowd scenes. This is mainly because performing VDR and occlusion culling has overheads. Also, our method may show visual artifacts with PoE values bigger than 1, while we were able to achieve interactive performance without noticeable artifacts by using 0.5 PoE for our tested models. Geomorphing can ameliorate *popping* artifacts by providing smooth transitions between different LODs.

3.7 Conclusion

We have proposed view-dependent representation for articulated models, VDR-AM, and presented how we can use it for an interactive view-dependent rendering method integrated with occlusion culling in large-scale crowd scenes. VDR-AM consists of a cluster hierarchy that serves both as a multi-resolution representation for VDR and a bounding volume hierarchy for occlusion culling. We also presented an error-aware cluster construction method to allow drastic simplifications on portions of meshes of articulated models. We were able to achieve 16 to 49 fps on average for large-scale crowd scenes that consist of thousands of articulated models and hundreds of millions of triangles without noticeable visual artifacts.

There are many avenues for future work. In addition to addressing current limitations of our method, we would like to first handle larger crowd scenes by designing a more drastic simplification method (e.g., volumetric simplification methods) as well as to simplify skeletal and behavioral models of characters [68]. One can combine our method with image-based representations [36, 37]; use our method in a near field and use impostors allowing more drastic simplifications in a far field. Also, we used deformation levels to estimate simplification errors. We would like to extend this concept to identify vertices that have similar rotational sequences [69], to more

accurately cluster vertices that have similar simplification errors. In addition, it would be interesting to conduct a user study measuring perceptual errors of our method. Finally, we would like to apply our VDR-AM to other geometric applications such as collision detection. Since our representation is based on a polygonal representation, we believe that it can be easily applied to collision detection in a similar spirit to the collision detection method designed for dynamic simplification [70].

Chapter 4. A Content-Aware Non-Uniform Grid for Fast Map Deformation

4.1 Introduction

As various map data have become available and portable screens get ubiquitous, displaying digestible map information has also become a challenge, especially in applications generating route and tourist maps. With mobile services increasingly integrating more as part of human lifestyle, our movement behavior can be analyzed and thus be utilized for customizing a unique map information. This trend is just beginning to be utilized in map representations [71], as more freely accessible and updated map data arises.

Map users can have various tasks at hand at a given time, and maps can be tailored to specific tasks. Deformation has been a common approach used by recent map makers in creating different views. To help in navigating tasks, deformation is applied by focus regions, which can be constructed by creating variably scaled regions according to importance [6]. To help in recall and memory retention, deformation is applied to roads that form shape patterns that are associated with regions [13]. To assist in analyzing and exploring geographical data, deformation is used to create cartograms [25].

Providing feedback in bi-directions between a user and device is critical to create such a unique and useful map [71]. Within a feedback loop, a user supplies the information and depending on that, a resulting personalized map is generated to address the current user need. It is, therefore, essential to formulate fast map transformations that bring real-time interaction in switching views with transitions [22]. Especially in cases where users have mobile devices on the move, map parameters have to be adjusted to display the current time/location-specific data.

Clarity, efficiency, reducing spatial information overload, and interactive experience are common goals of these deformation maps. While attainable at the moment for small maps, there is much to be improved, i.e., the number of roads under deformation, while maintaining high quality map representations (e.g., vector instead of image). As more roads are included, the time to process increases too. By deforming a uniform grid mesh of the map space instead of individual roads, the processing time can be lessened. However, choosing the resolution of the grid affects the quality of the deformation. In particular, optimization of a coarser mesh converges faster, but roads inside a quad may not be deformed well. On the other hand, optimization of a finer mesh deforms most roads well, but converges slower. In current approaches, this resolution parameter tuning is a trial-and-error process that could require repeated attempts.

Contributions. In this paper we offer the following contributions: 1) We extend the uniform grid method and propose an adaptive approach: an easily compatible non-uniform grid for deforming road networks. Subdivision of a non-uniform grid is determined by a significance function. In more significant regions, finer divisions are created, where roads can be deformed more carefully. In less significant regions, the divisions are coarser, thus less quad segments to optimize. Overall, this adaptive approach results in increase in road deformability and performance. 2) We also introduce support edges to preserve larger leaf quad shapes. We generate our non-uniform grid as a balanced grid, where each quad's neighbors have a single level size difference. 3) We implement our optimization method in CUDA to achieve a faster computation. Our goal is to scale up while maintaining efficiency, and to create the interactive experience, which is essential in many applications of using deformation maps.

We demonstrate how our approach can be easily integrated to existing approaches and improve their performance. We apply our approach to variable scaling in creating focus + context regions, transforming mental maps,

and destination maps. In these applications, our method using adaptive grids shows more robust scaling performance than those using uniform grids, showing slower relative growth rates in computational time as a function of residue. This robust improvement is mainly thanks to our content-aware non-uniform grids.

4.2 Related Work

In this section, we discuss prior methods that are directly related to our method.

4.2.1 Map Deformation

Deformation has been used to assist users in improving their cognitive tasks [9] in many different applications, e.g., road network, cartograms, and mental map generations. Maps can be easily re-generated and customized as desired by using different parameters. In addition, performance improvement of map deformation has given a way to interactive explorations of such maps. We discuss techniques related to map deformations used for different, but related applications below.

In map deformation approaches, road networks are represented as graphs. In most applications, a set of design objectives is defined in order to control characteristics of layouts of road maps, and is modeled into a minimization problem. In the work of Haunert et al. [6], regions of a road network are scaled to give focus to selected roads. Its road network graph is formulated as a quadratic programming problem, where its optimization is performed at *road edge-level*. Its follow-up work [10] reformulates the problem as a linear programming problem to achieve interactive rates for small maps consisting of a few thousands of edges.

Agrawala et al. [11] and Kopf et al. [12] describe automatic approaches to generate route/destination maps, i.e., a map summary of directions to a particular point of interest. Given a set of selected roads, an optimized layout that rescales and reorients road edges is computed in order to increase the readability of these road maps. Similarly, in the work of Lin et al. [13], the goal is to create mental maps, aiming to help easily recall a featured area. The road map is deformed such that certain formations can be recognized from shapes of the streets. Predefined control points are used to guide the final position of vertices. To achieve the goal, a uniform grid is overlaid on the map, which is used as a deformation medium – the linear programming optimization is done on *grid-level*, which enables higher performance than the edge-level approaches.

In Bouts et al. [25], contiguous area cartograms are generated from attributes of a geographic location. A uniform grid is used to overlay the map, and attributes are used to affect the deformation to show dissimilarity of attributes of a location. Similarly, Claudio et al. [20] used tourist destination data as attributes to deform and highlight popular regions.

As discussed above, uniform grids with grid-level optimization have been commonly adopted for fast map deformation. While this approach is simple, its time complexity goes higher as we have higher resolution for higher deformation quality. Our work extends uniform grid-level approaches by using a non-uniform grid that adaptively allocates edges. Because the optimization running time is commonly described by a function of those edges, our adaptive approach can improve the running time and even quality.

4.2.2 Adaptive Approaches

Adaptive Mesh in Graphics. In computer graphics, utilizing non-uniform grids and meshes is commonly adopted to achieve high quality simulation with a lower polygon count than regular structures (e.g., uniform grids). These adaptive approaches generate denser regions in interesting areas and coarser regions in uninteresting

areas [44]. However, using irregular structures for optimization problems can pose stability issues and tend to require complex theory and implementations. In similar reasons, using irregular meshes has not been widely tested for our problem, map deformation. Specifically, Lin [13] found that a grid mesh produces better deformations than that of a triangle mesh thanks to the regular structure of the grid mesh for map deformations.

Adaptive grids in GIS. In the field of GIS (Geographic Information System), different variants of regular structures have been actively studied [23]. The quadtree, a type of non-uniform grids, has been well studied; a good survey by Pajarola et al. [23] details the developments. In particular, we mention a specific type called the restricted quadtree, wherein nodes are allowed to differ in size by one subdivision level [45]. A restricted quadtree adaptively samples the space and balances tree depths locally, leading to a smoother transition in terms of map deformation. However, to the best of our knowledge, only a recent map deformation work [25] considered non-uniform grids, but decided to use the uniform grids, since they found no benefit as they treat the grid and map as independent entities.

In our approach, we take advantage of the geometry extracted from the map in order to create an adaptive grid that increases the performance of deformation. In creating an adaptive grid, quads that have significant areas are subdivided in order to give more precision in deforming those areas. To preserve shapes of quads and avoid various deformation artifacts, we introduce support edges, quadtree-like triangulation, but only applied to coarse quads with adjacent smaller finer quads, which are included in the deformation optimization.

Adaptive Simulation. In cloth simulation, cloth is typically modeled as a triangle mesh and forces between vertices are computed to determine the positions. A good survey on the methods is described[46]. Adaptive techniques have been proposed to improve performance [47, 48]. A more recent method by Lee[49] describes an adaptive mesh approach that identifies smooth regions that can be represented with lower resolutions while dynamic regions are represented by higher resolution in order to preserve the details of cloth simulation.

Similarly, we identify less significant regions that can be represented with lower resolutions and more significant regions that can be represented with higher resolutions. We apply this in a road map network deformation in order to achieve the same effect of higher performance while retaining accuracy of the model.

4.3 Background

In a simple road edge deformation, a road edge E undergoes a transformation $\mathbf{T}$, resulting to a deformed edge $\mathbf{T}E$. The goal of a basic edge deformation is to find a configuration of target edges such that the residual between the target, $\tilde{E}$, and the deformed edge $\mathbf{T}E$ are minimized. Specifically, the residual, D_{RE}, is defined as the following:

$$D_{RE} = \sum_{E \in \mathbf{E}} \left\| \tilde{E} - \mathbf{T}E \right\|^2, \tag{4.1}$$

where $\mathbf{E}$ is a set containing all those edges. In this simple edge-based deformation, its computational time increases, as the number of road edges increases.

In order to accelerate this edge-based deformation process, a grid overlaid on a map is typically adopted in a map deformation for many different applications, because optimizing with the grid can provide higher performance than directly optimizing individual elements of maps. A grid overlaid on a map consists of quads that encompass the map elements (e.g., road vertices, road edges, Points-of-Interest (POIs), textures) under them.

Optimizing the grid indicates that we perform the process in terms of the grid components instead of individual road edges. In particular, grid components include quads Q, quad edges A, and quad vertices V. We

assume that indices of two vertices of each road edge in Eq. 4.1 are E^1 and E^2, and those two vertices are then represented as barycentric coordinates associated with their respective quartet of quad corner vertices. The prior residual equation (Eq. 4.1) is then reformulated as the following:

$$D_{RE} = \sum_{E \in \mathbf{E}} \left\| \left(\sum_{p=1}^{4} b_p^{\tilde{E}^1} \tilde{V}_p^{\tilde{E}^1} - \sum_{p=1}^{4} b_p^{\tilde{E}^2} \tilde{V}_p^{\tilde{E}^2} \right) - \mathbf{T} \left(\sum_{p=1}^{4} b_p^{E^1} V_p^{E^1} - \sum_{p=1}^{4} b_p^{E^2} V_p^{E^2} \right) \right\|^2, \tag{4.2}$$

where $\tilde{V}$ is a target vertex and $b_p^{E^i}$ is a barycentric coordinate of a corner $V_p^{E^i}$ of a quad containing the edge endpoint indexed by E^i.

Deformation can deviate from the original shape of the map, so preservation of the shape of quads is applied [13]. Its objective, D_{QE}, called quad error, is defined as:

$$D_{QE} = \sum_{Q \in \mathbf{Q}} \sum_{i=1}^{4} \left\| \tilde{B} - \mathbf{K} \tilde{A}_i \right\|^2 \tag{4.3}$$

where $\tilde{A}_i$ is a deformed quad Q edge under $\mathbf{T}$, and $\tilde{B}$ is one of edges of the quad Q; the top edge is typically selected for this purpose. $\mathbf{K}$ is the original relationship computed by original edges, i.e., B and A_i.

Additionally, a smoothing objective can be considered to maintain the rigidity of the grid and implicitly avoid quad overlaps. The smoothing objective, D_{SM}, is a matrix form of the Laplacian smoothing [72]:

$$D_{SM} = \sum_{V \in \mathbf{V}} \left\| \mathbf{L} \tilde{V} \right\|^2, \tag{4.4}$$

$$\mathbf{L}_{ij} = \begin{cases} -1/Valence(V_i) & \text{if } V_i \text{ and } V_j \text{ are adjacent,} \\ 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, the optimization of the map is achieved by minimizing the weighted sum of all the objectives:

$$D = w_{RE} D_{RE} + w_{QE} D_{QE} + w_{SM} D_{SM}. \tag{4.5}$$

Depending a type of applications, this overall objective can be appended with additional objectives (Sec. 4.5).

4.4 Our Method

Uniform grids are popular thanks to ease of use and fast results in deformation with a coarse resolution. However, manual setting of the grid resolution and thus many trials would be required as to produce the desired effect. On one hand, a high resolution setting results in a high quality warping, but low computational efficiency. On the other hand, a low resolution setting results in a low quality warp, but high computational efficiency. Given this trade-off, Lin et al. show a grid resolution comparison and let the user tune the parameter [13].

Our adaptive grid tries to identify regions where we can set higher or lower resolutions in order to achieve a high quality warping along with a high computational efficiency. Fig. 4.4 shows these characteristics of using uniform grids with varying resolutions and our adaptive resolution in the application of mental maps. Compared to the quality of the high resolution grid (Fig. 4.4e), our adaptive grid structure (Fig.4.4b) produces the same overall shape with less edges and errors while performing faster. This pattern scales as shown in Fig. 4.4f: a slower relative growth rate in computational time. We now discuss how to construct our adaptive grids and its optimization.

4.4.1 Non-uniform Grid Construction

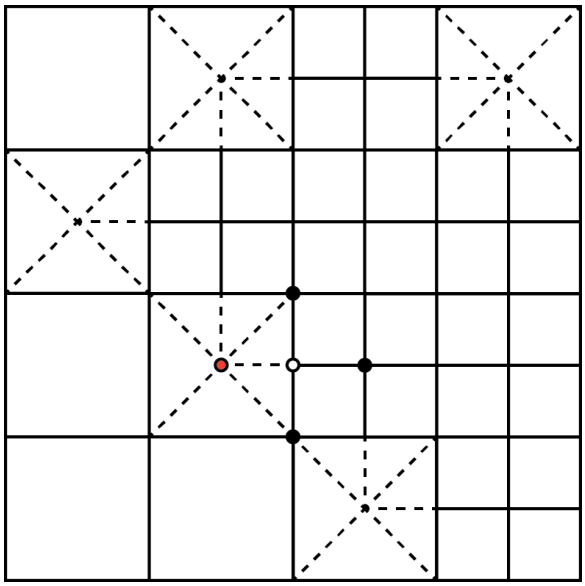


Figure 4.1: [A non-uniform grid with support edges (dashed). For smoothing our non-uniform grid, a T-junction (white vertex) consisting of only three neighbor vertices (black) is smoothed by considering the support vertex (red) as a fourth orthogonal neighbor.]

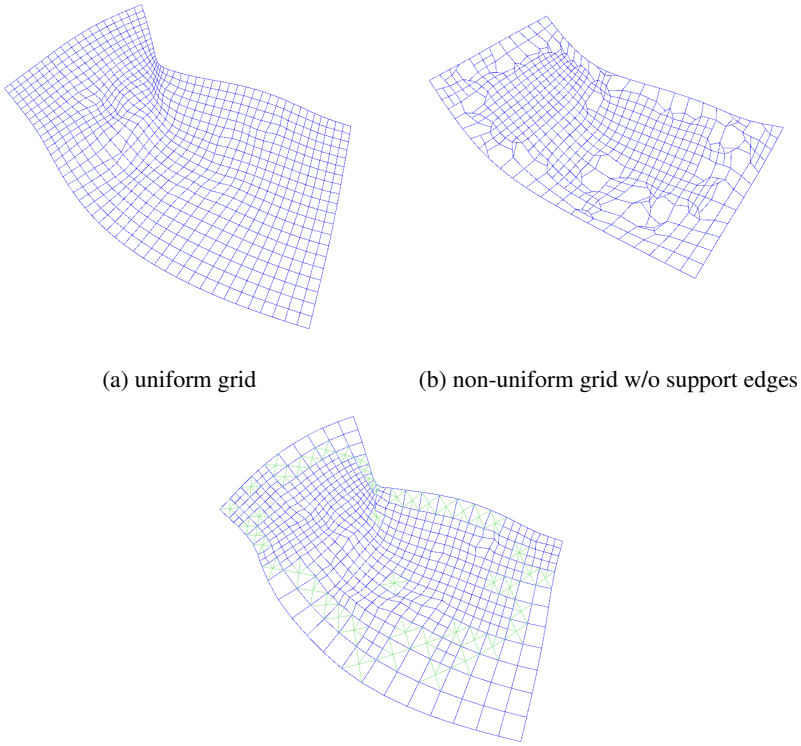
We construct our grid from user defined dimensions by a single root quad by default or by multiple quads if necessary. For each quad, we identify whether the area under the quad is significant by checking for 1) its enclosed road vertex count is greater than c, or 2) inclusion of a special vertex (e.g. guide pattern vertices, which are shown in the inset of Fig.4.4a). When either one of criteria satifsies, we divide the quad into four sub-quads, resulting in finer subdivision.

Recursive subdivision can result in an unbalanced quadtree. In our experiments, we observe that the overall shape quality can be improved by balancing the tree through conversion into a restricted grid, wherein neighbors of a quad must be within one level of each other in terms of the tree depth. This restriction prevents a sudden change of deformation rate over a surface, resulting in artifacts minimized [73].

In the uniform grid approach, quads would deform without shape collapsing, because the surrounding quads have the same size. However, in a non-uniform grid like in a quadtree, quads do not necessarily have the same sizes, producing T-junctions that can cause "cracks". Inspired by techniques from terrain rendering, we apply triangulation and provide support edges to remove such cracks. Our goal is to decrease the number of edges and thus increase the optimization performance, while maintaining the accuracy of map deformation. Following the behaviour of the uniform grid, a neighborhood of same sized quads would support themselves. We therefore apply triangulation support edges only to quads with smaller neighbors (Fig. 4.1). The support edges are included to a quad's edges and are considered to maintain the shape of quads based on Eq. (4.3).

Smoothing. Laplacian smoothing is applied to each vertex with its orthogonal neighbor vertices. In the case of non-uniform grids, T-junctions may occur due to non-uniform subdivision. To complete its 4-way neighborhood, we consider the centroid vertex, e.g., the red vertex shown in Fig. 4.1, of the adjacent bigger quad, to serve as a fourth neighbor to these T-junctions.

While useful in maintaining rigidity and implicitly avoiding overlaps, smoothing comes with a problem



(a) uniform grid (b) non-uniform grid w/o support edges

(c) non-uniform grid w/ support edges

Figure 4.2: [Support edges preserve the shapes locally and globally.]

known as the shrinking, brought to by boundary vertices converging to the centroid of its neighbors towards its internal vertex neighbor. We address this by applying a simple method of Taubin [74] by projecting the Laplacian to the vertex normal.

Fig. 4.2 shows the deformation produced using a uniform grid, non-uniform grid without support edges and a non-uniform grid with support edges.

4.4.2 Optimization

We encode the system of linear equations from Eq. 4.5 in the form $\mathbf{Ax} = \mathbf{b}$, which can be described as a large sparse system. For such a system, we employ a common iterative algorithm used in layout optimization [53, 13] called the conjugate gradient method, which requires a symmetric matrix. In most cases, $\mathbf{A}$ is not symmetric, so we transform to the normal equation form $\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$ to obtain a positive-semidefinite matrix $\mathbf{A}^T\mathbf{A}$.

We refer to the conjugate gradient algorithm described in [75]. Shewchuk [76] describes specific implementation details to improve the stability of the optimization. We implemented the algorithm both in CPU and in CUDA / cuBLAS / cuSPARSE [77]. Fig. 4.3 shows that our GPU implementation shows up to 88% increase in performance in our tested settings. A GPU implementation of the conjugate gradient method is essential in the goal of scaling the data size while keeping interactive rates.

Preventing foldovers. While layout optimization generates a best local solution for minimization of residue, it is not concerned for a foldover-free result, which is visually undesirable. Haunert et al. [6] prevents foldovers by including non-crossing objectives, but uses quadratic programming, which is slow. Most linear programming approaches prefer checking and correcting foldovers after the fact. Some of them include *rewinding* to the time

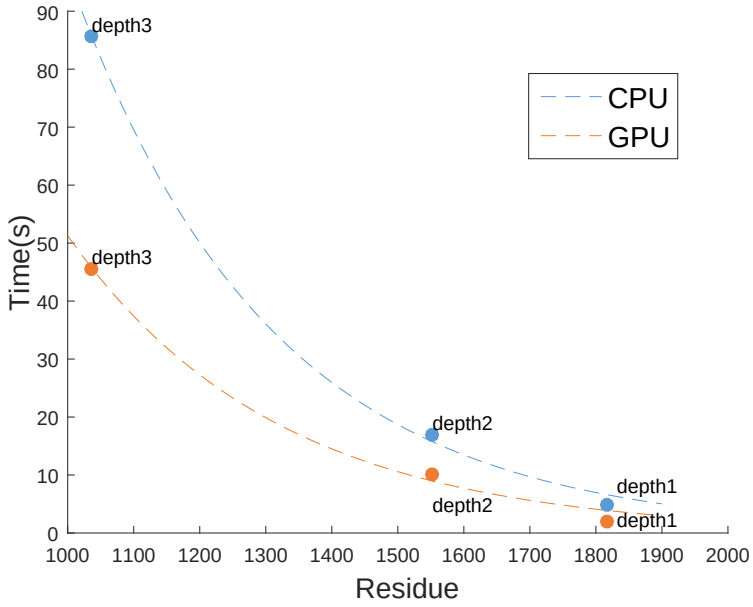


Figure 4.3: [Performance Comparison: CPU vs GPU. As resolution increases and residue decreases, GPU's computational time shows a slower growth rate than the CPU. Three different points labeled as depth1, depth2 and depth3 indicate pairs of running time and residue for our method.]

of earliest point of the contact[10, 78], Procrustes projection [79], and Laplacian smoothing [13]. In a large system like ours, Laplacian smoothing implicitly provides foldover prevention by distributing neighbor spacing, resulting in a high performance due to only involving matrix multiplications. To further ensure solutions in cases of non-convergence, an additional relaxation step can be performed after each iteration: a simple mass-spring minimization[80]. As with [13], these foldover steps are optional in order to maintain efficient computation for interactive systems.

4.5 Applications

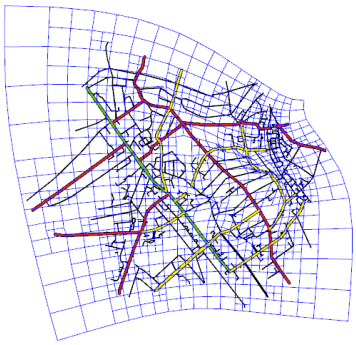
Our proposed structure can be applied easily in grid based deformation techniques and we demonstrate three such cases.

Evaluation. To show different behaviors of uniform grids and our adaptive grids, we show the performance time graph as a function of the residue of each method.

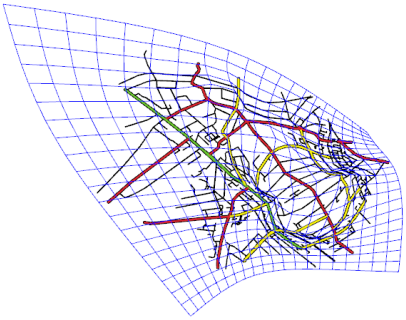
Computing residues using Eq. 4.5 for a uniform grid and a non-uniform grid differs due to the unequal number of grid elements (e.g. quads and support edges), but the number road elements and guide vertices are the same for both approaches. To perform a fair quantitative comparisons between these two different methods in terms of residue values, we consider only residues computed from the unweighted road edges (Eq. 4.1), and the unweighted guide objective when it exists. These residue values are used only for comparisons (e.g., the time vs. residue graph) between two methods, and are different from the ones used for the minimization of residue optimization mentioned in Sec. 4.4.2.

The time complexity of the conjugate gradient method is shown to be $\mathcal{O}(m\sqrt{\kappa})$, where m is the nonzero entries in the matrix $\mathbf{A}$ and κ is its condition number [76]. We employ a positive-semidefinite matrix $\mathbf{A}^T\mathbf{A}$, and its the condition number is the square of the condition number of $\mathbf{A}$, making our method's time complexity to be

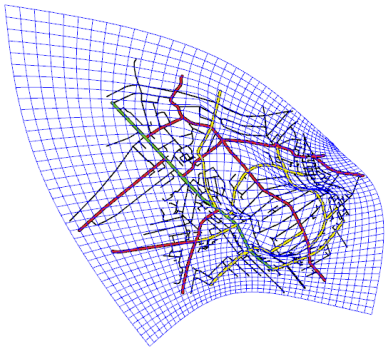
(a) Map featuring Abbey Road. Inset: guitar guide pattern



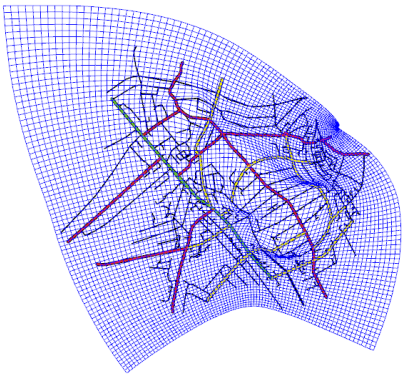
(b) Ours depth3 (residue=683.22, 1.64s)



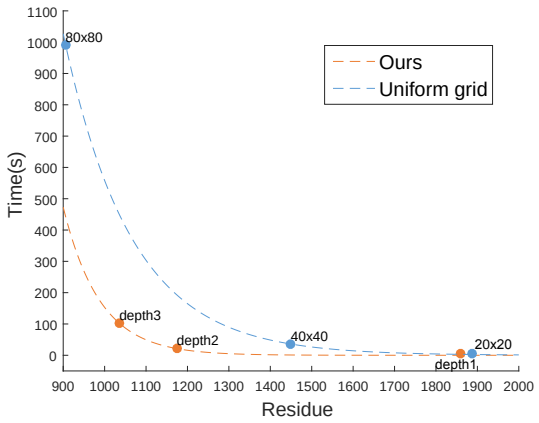
(c) 20x20 (residue=880.31, 0.17s)



(d) 40x40 (residue=714.90, 1.35s)



(e) 80x80 (residue=688.85, 2.20s)



(f) Performance graph. As resolution increases and residue decreases, our adaptive grid's computational time shows a slower growth rate than the uniform grid. The base grid for our adaptive approach is 10x10. Three different points labeled as depth1, depth2 and depth3 indicate pairs of running time and residue for our method.

Figure 4.4: [Abbey road mental map]

$\mathcal{O}(m\kappa)$. Our computation setup features a 3.60GHz CPU and an NVDIA GTX 680 GPU.

4.5.1 Mental and Destination Maps

Mental maps or destination maps are classified into thematic maps that convey a specific theme or mental image of particular regions.

Mental maps are defined as maps that feature a pattern formed by roads to help users form a mental image of the location. The pattern is supplied as target vertices that have corresponding real road vertices in the input map. The optimization is guided by those input vertices. Lin et al. [13] designed a uniform grid based deformation to create such maps.

This approach adopts an overlaid grid, while preserving road and quad shapes. The road edge and quad shape preservation objectives are defined as in Eq. 4.2 and 4.3. In addition, a guide objective, D_{GV}, is enabled to direct specified road vertices to the input vertex pattern of a mental map, and is shown in below:

$$D_{GV} = \sum_{G \in \mathbf{G}} \left\| \tilde{R} - G \right\|^2, \tag{4.6}$$

where G is a vertex in the set of target vertices, R is its corresponding road vertex, and $\tilde{R}$ is the deformed version of R. In terms of a quad barycentric coordinate, Eq.4.6 becomes:

$$D_{GV} = \sum_{G \in \mathbf{G}} \left\| \sum_{p=1}^{4} b_p^{\tilde{R}} \tilde{V}_p^{\tilde{R}} - G \right\|^2. \tag{4.7}$$

In our method, we replace the uniform grid into our non-uniform grid and apply the optimization process (Sec. 4.4.2).

As an example, Fig. 4.4 shows the famous Abbey road having the same name of one of the Beatles' albums. In this example, we want to use the guitar pattern (the inset of Fig. 4.4a) as a mental map to form the shape of the road. We compare our work to a uniform grid and measure their performance (Fig. 4.4f). In our tests, we found that as the resolution increases and the residue decreases, the computation time of our adaptive grid has a slower relative growth rate than that of the uniform grid.

Another application of road deformation is to generate destination maps, defined as maps that show a sketch of a simplified route towards a destination. In this case, the input is a sketch pattern and a geographic map, and the output is a deformed geographic map satisfying the route sketch pattern. For this application, we test the Malaysia map (Fig. 4.6) with a sketch (Fig. 4.5).

We also test prior uniform grids with varying resolutions against our adaptive method. We found that the time (Fig. 4.6e) as a function of residue chart shows that our approach is more scalable, although our adaptive grid has shown slow computational time in low resolutions. This low performance at a lower resolution is mainly due to the overhead brought to by including support edges in the objectives (Sec. 4.4.1). Nonetheless, as we use higher resolutions, our method shows a slower relative growth rate than that of the uniform grid.

4.5.2 Focus region maps

A focus region map is a focus+context map wherein a focused region is highlighted by making it standout from the rest of the region (context). In the case of Haunert et al. and Van Dijk et al. [6, 10], a focus region is defined by scaling the road edges in that region. The scaling factor for the focus region edges is a user input, and the rest of the map as the context is variably scaled to preserve road edge lengths. Their optimization works originally at the road edge-level like the one shown in Eq. 4.1:

$$D_{RE} = \sum_{E_i \in \mathbf{E}} \left\| \tilde{E}_i - s_i E_i \right\|^2, \tag{4.8}$$

Figure 4.5: [Target pattern (magenta) over the original map, shown with displacement vectors for creating a destination map.]

where s_i is the scaling factor.

For the use with a grid, we expand the edge-level objective function into the grid-level, in a similar manner to derive Eq. 4.2:

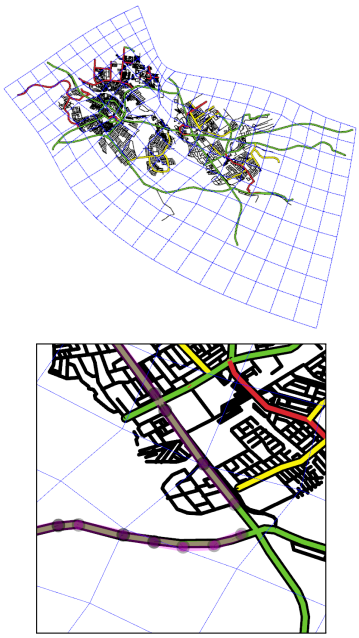
$$D_{RE} = \sum_{E_i \in \mathbf{E}} \left\| \left(\sum_{p=1}^{4} b_p^{E_i'^1} V_p^{E_i'^1} - \sum_{p=1}^{4} b_p^{E_i'^2} V_p^{E_i'^2} \right) - s_i \left(\sum_{p=1}^{4} b_p^{E_i^1} V_p^{E_i^1} - \sum_{p=1}^{4} b_p^{E_i^2} V_p^{E_i^2} \right) \right\|^2 . \tag{4.9}$$

The difference in this application is that the user inputs the scale factor Z for a set of edges E_f in the specified focus region such that $s_i = Z$. We test using the Boston road map with multiple focus regions and compare the road edge-based, the uniform grid and the non-uniform grid approaches (Fig. 4.7). In our tests (Fig. 4.7e), we found that as the resolution increases and the residue decreases, our adaptive grid's relative growth rate is slower than that of the uniform grid.

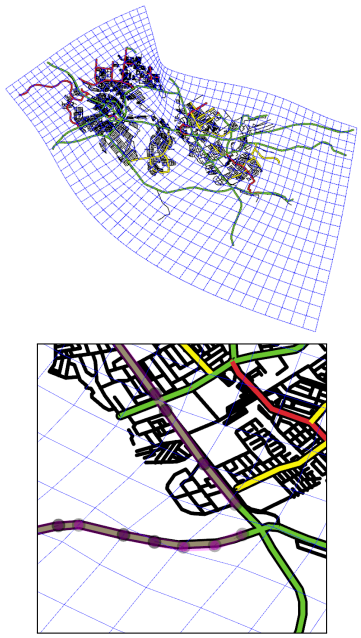
4.6 Conclusion

We proposed a non-uniform grid that adaptively adjusts resolution for significant areas in the map without needing for resolution parameter tuning. We also applied support edges for coarser quads, which help to stabilize and preserve the quad shapes of our adaptive grid. In different map deformation applications, our adaptive grid's performance scales better than a uniform grid, displaying a slower relative growth rate. We also showed that the performance further increases by implementing the optimization in CUDA.

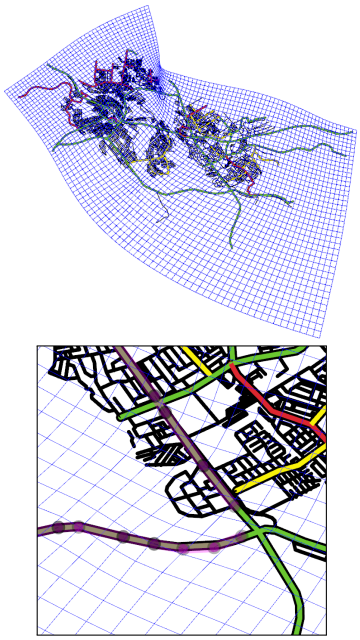
In a future work, we would like to apply our approach to include deforming octilinear metro networks [53] with road maps. We would also like to consider other user specific data like genre preferences, GPS locations in customizing road maps.



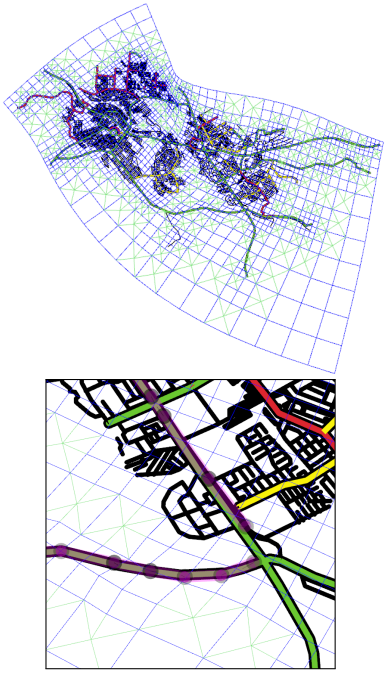
(a) 20x10 (residue=1865.83, 1.20s)



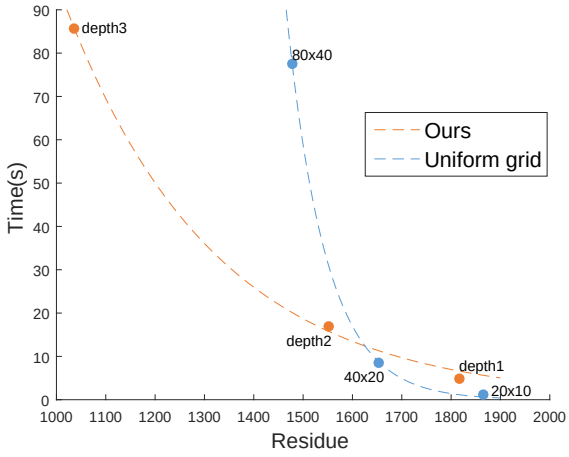
(b) 40x20 (residue=1653.79, 8.62s)



(c) 80x40 (residue=1477.70, 77.63s)

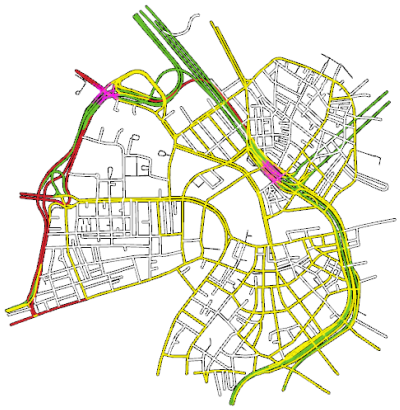


(d) Ours depth3 (residue=1036.36, 85.61s)



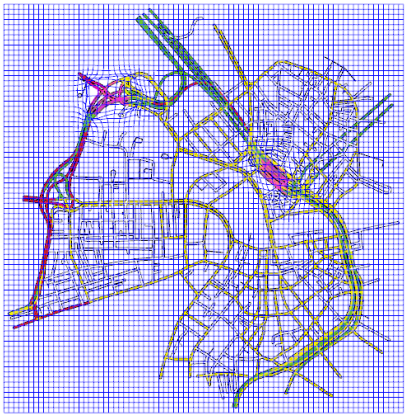
(e) Performance graph. As resolution increases and residue decreases, our adaptive grid's computational time shows a slower growth rate than the uniform grid. The base grid for our method is 20x10, and three points labeled as depth1, depth2 and depth3 are shown to indicate pairs of running time and residues for our method.

Figure 4.6: [Destination maps (Malaysia) with different resolutions. (a-d) Top shows the overall grid shape, while the bottom shows a zoom-in view. Input target routes are shown in magenta.]

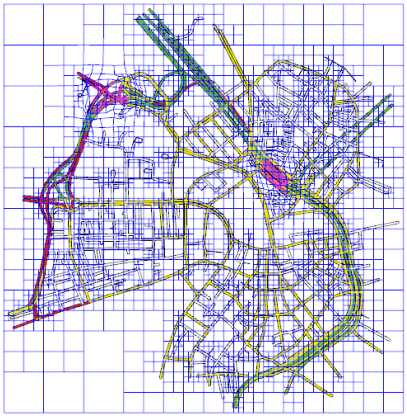


(a) Original

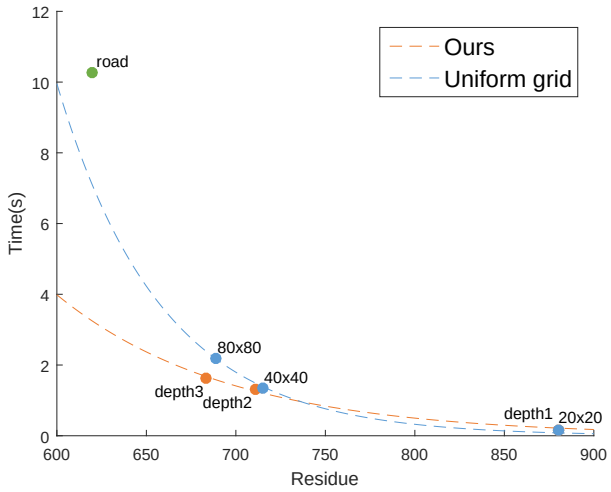
(b) Road edge (Van Dijk et al.) (residue=619.45, 10.28s)



(c) Uniform grid 80x80 (residue=688.85, 2.20s)



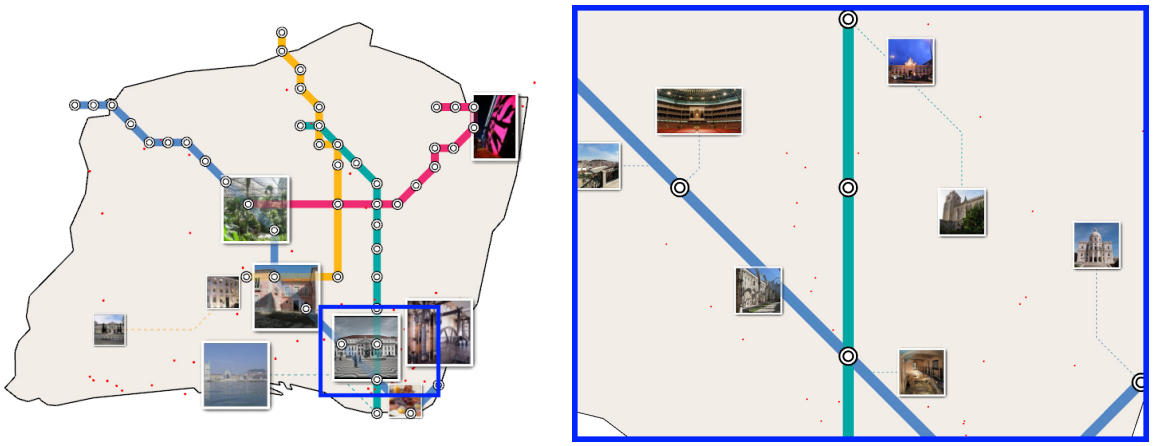
(d) Non-uniform grid depth3 (Ours) (residue=683.22, 1.64s)



(e) Performance graph. As resolution increases and residue decreases, our adaptive grid's computational time shows a slower growth rate than the uniform grid. The green dot indicates the running time and residue of the road edge-level computation. The base grid for our method is 10x10 and three points labeled as depth1, depth2 and depth3 are shown to indicate pairs of the running time and residue for our method.

Figure 4.7: [Boston map with multiple focus regions. (a-d) Magenta edges are set to scale factor $Z = 3$.]

Chapter 5. Metro Transit-Centric Visualization for City Tour Planning



(a) Default zoom level (b) Zoomed-in view

Figure 5.1: [This figure shows our maps for Lisbon with different zoom levels.]

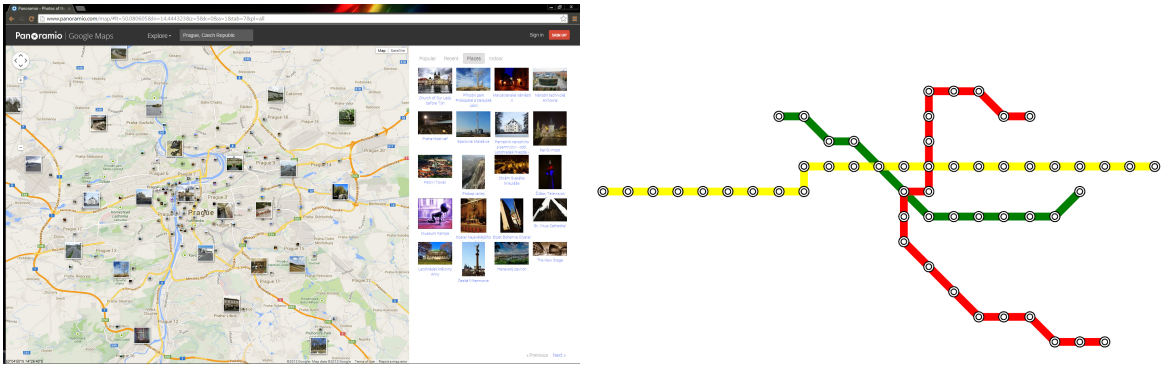
5.1 Introduction

Thanks to advances of various transportation means and increased leisure time of common people, people have more opportunities of traveling to popular tourist destinations. In traditional city trip planning, tourist destination discovery comes typically from looking at reviews or geographic map highlights (Fig. 5.2-(a)), and results in a list of preferred destinations. Planning a tour route among those identified tourist destinations is the next separate step, by looking at transportation maps (Fig. 5.2-(b)).

This decoupled approach may result in changes on the preferred destinations depending on distances and transportation availability. In addition, swapping from a geographic map to a transportation map creates a mental gap, as the planner tries to remember details from one map, and places and translates them into a different map. Unfortunately, prior visualization techniques for metro network [14] and their labels [19] mainly focused on visualizing individual components of common city trip planning. As a result these approaches are not mainly designed for providing holistic visualization for this kind of trip planning.

Geo-tagged photo websites (Fig.5.2-(a)), e.g., *flickr.com* and *panoramio.com*, assist tourist destination discovery by highlighting popular photos in geographic maps. Some tour planner websites (e.g., *planner.com*) also offer popular destinations in a similar photo discovery in geographic maps. In these websites, photos are clustered and usually presented with a representative image. Display of these photos, however, are not composed by explicitly considering reachability from metro or other transportation means.

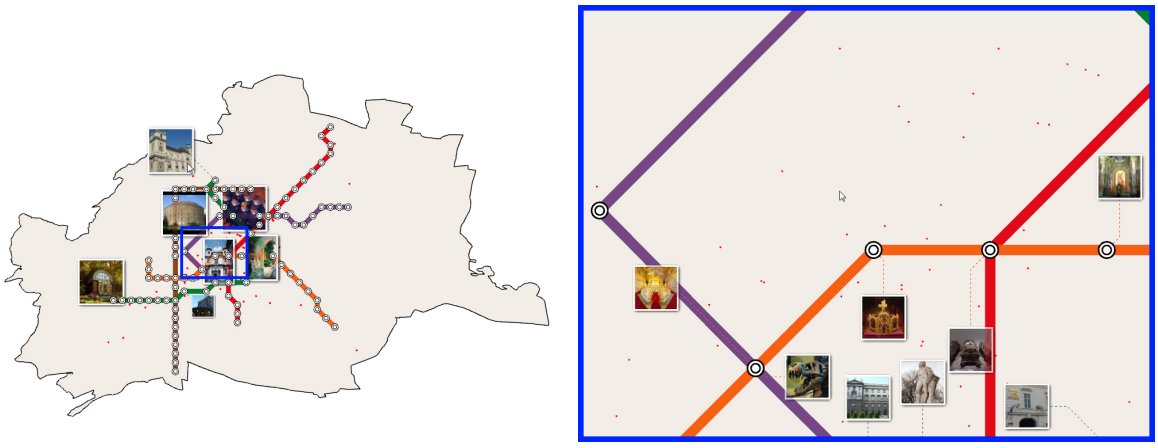
To address this problem, the metro network is commonly featured in the geographic map. Nonetheless, it can be easily buried in photos and other geographic map information such as various labels and roads. It is important to effectively show reachability from a metro map to tourist destinations during the city trip planning. An approach demonstrated in Google maps shows transit lines and an external strip of POI photos, pointing to a POI location one-at-a-time. As opposed to the approach in geo-tagged photo websites, this form belongs to



(a) Tourist destinations map (b) Schematic metro map

Figure 5.2: [Two separate maps are commonly used for planning. (a) is a screenshot of *panoramio.com* Swapping between these two different maps can cause a mental gap for city tour planning.]

external annotation, occupying additional precious screen estate, such that when applied to a smaller display, the screen could get easily cluttered.



(a) Default zoom level (b) Zoomed-in view

Figure 5.3: [This figure shows our maps for the Vienna city with different zoom levels.]

Main contributions. We propose a holistic visualization technique that supports tour planners to effectively identify preferred tourist destinations and their reachability from the metro map of a chosen city (Fig. 5.1). To achieve our goals, our method uses the metro network as the center of planning and provides a schematic diagram of a metro map populated with representative photos. To help tour planners to effectively navigate the metro network and tourist destinations, we provide them an automated octilinear layout of the metro. In addition, we identify and highlight popular tourist areas that are also in the vicinity of metro stations.

Our method starts from both a metro network, represented with a graph, and Points-Of-Interest (POIs) provided by trip reviews websites such as *tripadvisor.com*. Each POI is associated with its name, its geographic location, genre, photos, etc. Our method identifies popular tourist areas based on kernel density estimation. We then assign high visual worths to such areas for allocating more visualization space for them (Sec. 5.4.1). For computing the schematic metro map in the octlinear layout, we apply a mixed integer programming approach for computing a near-optimal solution (Sec. 5.4.2). Once the schematic metro map is constructed, we position POIs in the map by warping geographic locations to the computed map (Sec. 5.4.3). We finally perform a hier-

archical clustering method for computing representative images out of densely populated POIs in a small region (Sec. 5.4.4). Fig. 5.4 summarizes our method.

To validate our approach, we have conducted a user study surveying 70 individuals. We have factored out our six design guidelines and used them as a basis for designing our method. We have found that our method receives a high user satisfaction rating average of 1.097 in the range $[-2, 2]$ of our Likert scale. These results demonstrate the benefits of our proposed method.

5.2 Related Work

We review prior methods directly related to our approach in this section.

5.2.1 Metro Network Layout

Different approaches have been proposed for computing layouts for metro network [14]. Among them, octilinear layouts using diagonal edges as well as horizontal and vertical edges have been widely adopted for visualizing metro networks.

Octilinear layouts. Automatic generation of octilinear metro map layouts has been studied in recent years. One of the initial works was of Hong et al. [15]. This approach utilizes mass spring techniques, where forces dictate relative positions to achieve octilinear layouts. Stott et. al. [16] used a hill climbing algorithm to optimize the positions of the stations.

Nollenburg et al. [1] viewed the layout problem as an optimization problem, and employed a mixed integer programming (MIP) approach to find a global solution maximally satisfying design constraints. Its resulting layouts show a strict compliance to octilinearity and feature almost uniform spacing. Similar MIP techniques have been used in related layout problems [17]. Our work also utilizes MIP based optimization for computing a global solution given our design criteria.

Annotation placement. Map annotation has been well investigated, yet annotation placement in octilinear metro maps is another problem raised with automated octilinear layout generation. Wu et al. [18] used flow networks to put external annotations around the map boundaries connected by lines to the stations, while avoiding intersections with metro lines. More recently, Wu et al. [19] investigated internal annotations: placing them in the vicinity of the sources; an MIP framework proposed by Nollenburg et al. [1] is utilized to place annotations without overlaps, while drawing octilinear line connectors. Although the resulting internal annotations may not retain relative topology, the spacing is well distributed thanks to the additional aesthetic criteria such as alternating distribution and closed region avoidance. Nonetheless, these techniques are not designed for interactive tour planning and computing representative POIs.

Map warping. Jenny et al. [50] used map warping, Helmert transformation, to analyze geometric distortions of maps, and analyzed octilinear metro maps compared to its geographic counterpart. Bottger et al. [22] used distortion by moving least squares to fit a street map in an octilinear metro map. We also apply map warping techniques for aligning POIs in the computed octilinear layout. A simple map warping method like Jenny et al. suffices for our purposes of transforming point positions. Yet more sophisticated methods exist such as Bottger et al., which can be substituted in order to perform non-overlapping transforms.

5.2.2 Removing Clutters

Metro maps can be populated with many associated attributes such as labels, pictures, etc. Focus+Context and computing representative images have been studied to optimize the display of these annotations.

Focus+Context. Focus and context is one of the staple techniques in the visualization community. This technique magnifies graphs and grids to focus on certain areas, while retaining the normalcy of the remaining areas as a context [51]. In particular, Sarkar and Brown [52] defined a visual worth attribute for each vertex to determine its importance in visibility. In the work of Wang et al. [53], as applied in metro maps, visual worth are set to be higher for edges in the selected route.

Representative images. In a map featuring hundreds of photos, a representative image is usually used to represent an area, as commonly applied in geo-tagged image websites (e.g., *flickr.com* and *panoramio.com*). An approach by Jaffe et al. [4] is to display summaries of a large collection of images by traversing a hierarchy resulting from the Hungarian hierarchical clustering method[54]. The representative images of a cluster are defined by their scores coming from criteria such as relevance. We find the Hungarian method useful as a hierarchical clustering method, which creates a hierarchy handily used as an LOD tree. At the same time, this method does not need a defined number of clusters as input. These techniques have not been applied to city tour planning, the main application of our method.

5.3 Background

In this section we go through a domain of applications addressing map switching. In particular, we view tools that explore solutions on combining metro maps with tourist/street maps.

Reilly et al. [21] investigated image map morphing as a tool to reconcile two different maps of the same place. One-to-one mapping points are defined to guide the image morphing process and intermediate transition images are recorded. Bottger et al. [22] used distortion by moving least squares to fit a street map in an octilinear metro map space. Zooming-in morphs the image until it converges to the undistorted geographically accurate map. As these applications apply image distortion to the maps, resulting artifacts include curvy street lines, cross-blending. We take inspiration from these works, but we do not use image space in distorting our maps.

Our work strives to improve on creating an optimized tourist map in a single octilinear space for use in trip planning. We parallel our design principles on complementary works on trip planning such as route maps [11, 12], which discard detailed map solutions for a clear and easy-to-read format in conveying information. Also by working in a single octilinear space through interpolation, the difficulty of adjusting between map conventions and spaces is absolved. This design can bring added usability. For example, near a tourist center, there may be many POIs and nearby metro stations. One can easily identify different routes to cover preferred POIs by taking metro transits or walking, by looking at the single octilinear layout populated with POIs.

Note that most tourist maps are still handcrafted despite automated solutions exist. Aforementioned works may not be popularly applied on their own applications yet. Nonetheless, automatically generated layouts can be used as basis by artists, as described in [19].

Tasks As discussed in Chap. 1, we would want to address the functions of the tourist map in a holistic application. To recap it involves providing digestible info, POI discovery in a metro-centric map, and effective route planning from an octilinear metro layout. Accordingly, we define the following goals and tasks for the users.

1. **Map Information** What are the names of the POIs? Which regions are tourist spots?
2. **POI Discovery** Which are the top POIs in a specific area? Which are the nearby minor POIs?
3. **Route Planning** Which POI are reachable from a metro? How to reach from one POI to another POI via metro?

5.4 Our Approach

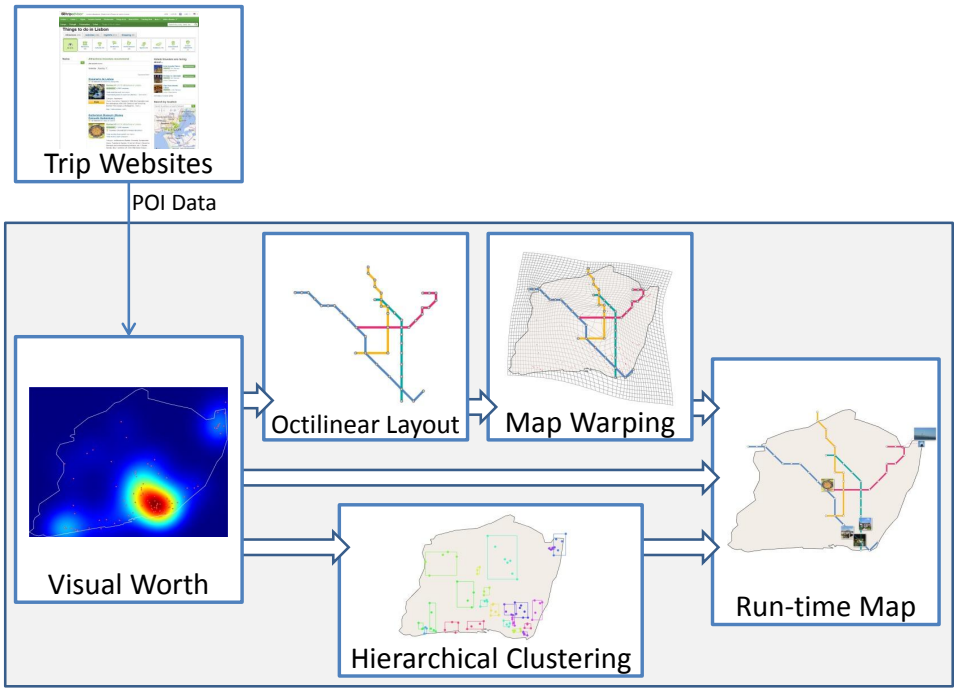


Figure 5.4: [This figure shows the overall structure of our system.]

In this section we elaborate each part of our method. For designing these parts, we have factored out various design guidelines, which are discussed in Sec. 5.5.1. Based on those observations, we adapt our final design choices.

As the base information to our method, we harvest a metro network map and POIs representing interesting locations of a city from trip reviews websites like *tripadvisor.com*. In addition to harvesting basic information (e.g., name and location) for each POI, we also collect its popularity ranking among those identified POIs.

5.4.1 Visual Worth Computation

We aim to magnify the spaces of popular tourist areas for allocating larger visualization working spaces to them. To effectively identify such areas, we transform the problem of identifying them into a density estimation problem. Specifically, we use the variable kernel density estimation method [81] with the POI geographic coordinates as samples (Fig. 5.5).

We also utilize available ranking information, r_i, for i-th POIs and consider it in our density estimation. In particular, we emphasize highly ranked POIs. To realize our purpose, we use the POI ranking as a factor to a constant kernel bandwidth, k. In other words, lower ranked POIs are used with larger bandwidths, and thus affects visual worths on their nearby regions in low weights. We also consider the proximity, ρ_i, of POIs from the nearest metro station, since close POIs from metro stations can be also preferable for tourists.

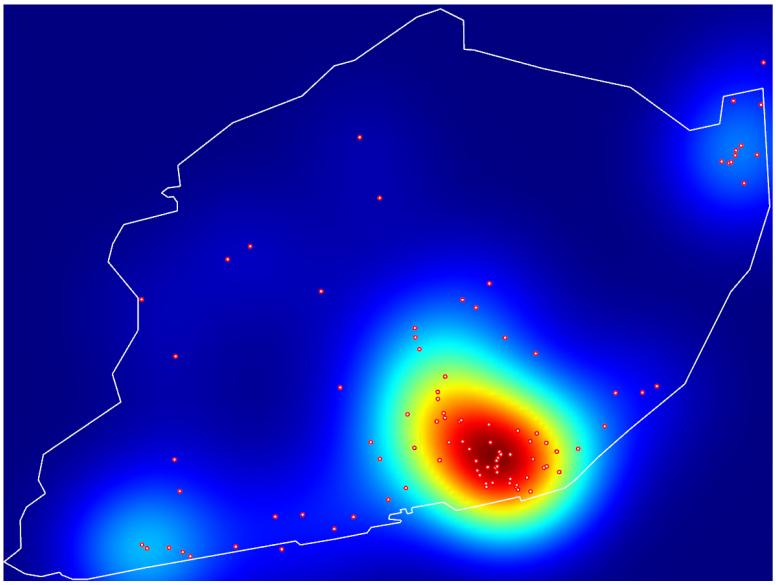


Figure 5.5: [This figure shows a heat map of Lisbon resulting from our kernel density estimation with POIs (shown in red dots).]

Given a point u on a geographic map, we estimate its visual worth $vw(u)$, as the following:

$$\begin{aligned} vw(u) &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{(h_i)^2}K(POI_i,u,h_i), \\ h_i &= k(w_r r_i + w_\rho \rho_i), \\ K(POI_i,u,h_i) &= \frac{1}{2\pi h_i^2} e^{-\frac{d(POI_i,u)}{2h_i^2}}, \end{aligned} \tag{5.1}$$

where n is the number of samples, h_i is the variable bandwidth, K is a 2D Gaussian kernel, POI_i represents the location of i^{th} POI, and $d(POI_i,u)$ is the Euclidean distance function. w_r and w_ρ are the weights for the terms ranking r_i and proximity ρ_i.

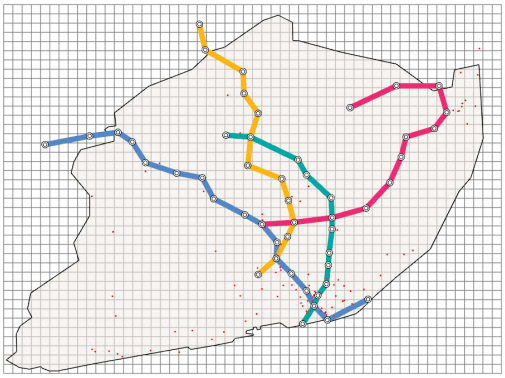
Fig. 5.5 shows an example of our visual worth computation for a city. We treat high-density areas (shown in red colors) as tourist centers of the city and allocate more space for those areas in later parts of our method.

Recalling Fig. 5.4, the computed visual worth values directly affect later stages of our method. On one hand, we use visual worth to control edge lengths in the octilinear layout computation. On the other hand, we use visual worth to extract prominent clusters in the active cluster set.

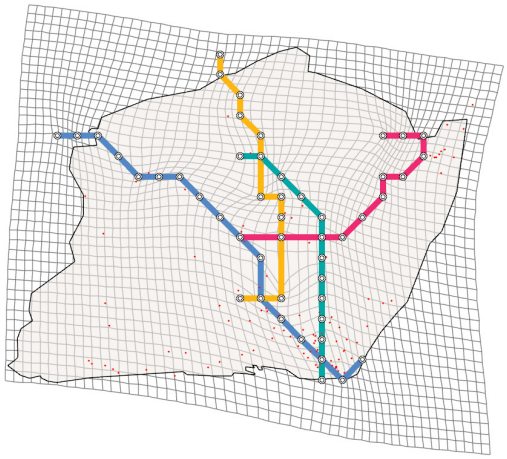
5.4.2 Octilinear Layout Computation for Metro Network

We use the given metro network as the center of our visualization. To support tour planners to effectively navigate the metro network and tourist destinations, we compute an octilinear layout of the given metro network as its schematic representation. In particular, we use an MIP based optimization method for computing the octilinear layout, while minimizing costs that are measured against various octilinear layout design criteria [1].

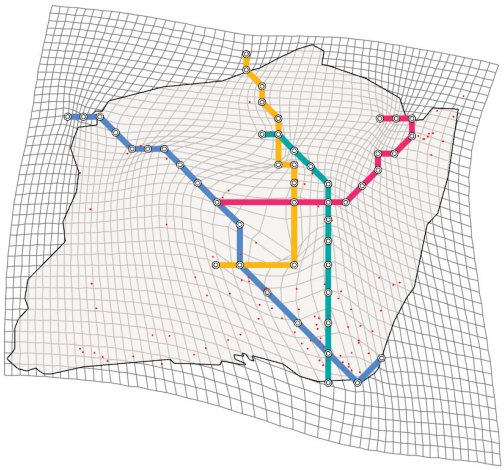
For our purpose, we aim to minimize bending angles between the physical metro line and its corresponding edge in the layout, $cost_{bend}$, to preserve the original topology of the metro network, $cost_{topo}$, and to reduce the sum



(a) Input



(b) Our octilinear layout w/ the uniform edge length



(c) Our octilinear layout w/ the variable edge length

Figure 5.6: [(b) and (c) show computed octilinear layouts of the Lisbon metro network (a) constructed with uniform or variable edge lengths. POIs, shown in red dots, are also transformed according to the computed octilinear layout.]

of edge lengths and avoid excessively long edges, $cost_{length}$. They are defined in the following equations:

$$\begin{aligned} cost_{bend} &= \sum_{L \in \mathscr{L}} \sum_{uv,vw \in L} bd(u,v,w), \\ cost_{topo} &= \sum_{uv \in E} rpos(u,v), \\ cost_{length} &= \sum_{uv \in E} \lambda(u,v), \end{aligned} \tag{5.2}$$

where u, v, and w are station vertices, E is a set of metro edges in the layout. L is a geographical metro line in the set of physical metro lines, $\mathscr{L}$, and $bd(u,v,w)$ measures the bending angle between two different edges corresponding to two sub-lines, uv and vw, sampled from a geographical metro line L. $rpos(u,v)$ measures the angle difference between the edge (u,v) in the layout and its corresponding line in the metro network, and λ returns the edge length cost.

As the final cost function to the MIP optimization process, we combine these three costs with different weights; we put highest weights for $cost_{topo}$, since maintaining the relative topology of the metro network is one of the most important visualization criteria for our purpose. For all the experiments we use 1, 50, and 1 for $cost_{bend}$, $cost_{topo}$, and $cost_{length}$, respectively, as weights for the MIP optimization process.

Edges between stations are kept to have a length that is equal or higher than the minimum edge length, ℓ_{uv}. We then compute an edge length according to visual worths of two end points of the edge. The following equations define two aforementioned constraints used within our MIP optimization process for east horizontal cases:

$$x(u) - x(v) \geq \ell_{uv}, \tag{5.3}$$

$$\ell_{uv} = \ell_{const} \frac{vw(u) + vw(v)}{2}, \tag{5.4}$$

where $x(u)$ represents the x coordinate of the vertex u and ℓ_{const} is a constant factor converting from the visual worth to the edge length; we set 10 for ℓ_{const} in practice.

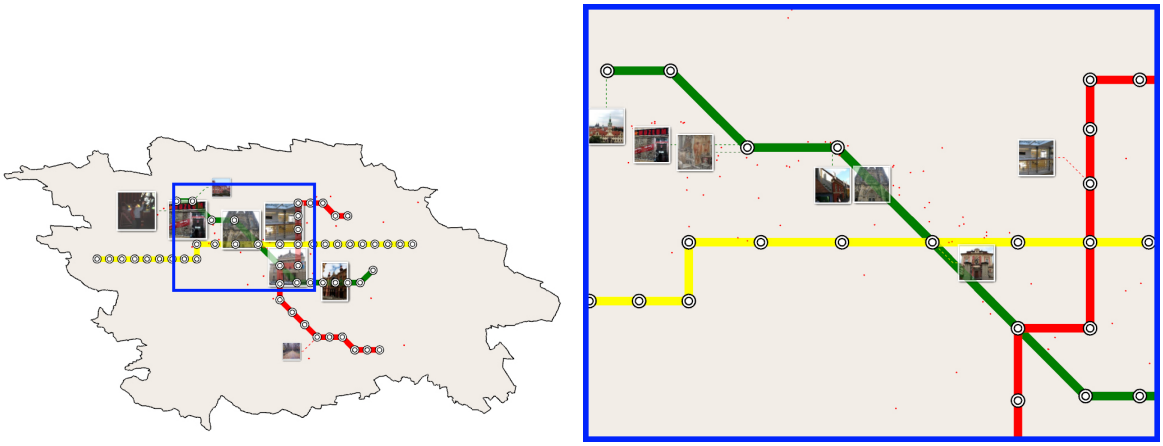
We also have similar equations for the other seven directions. The MIP optimization process uses constraints depending on configurations among the eight possible octilinear directions of an edge. We do not check edge overlaps, expensive constraints within our MIP optimization process, since we have found that the other used constraints combined with a small edge length, produce layouts without edge overlaps in our tested benchmarks. In a default setting, we show labels associated with metro stations, only when users click stations. This design choice is made to avoid excessive clutters. Nonetheless, we can naturally consider labels within our MIP optimization process, as mentioned in a related prior work [1].

Fig.5.6-(b) and -(c) show octilinear layouts of the Lisbon metro network that are computed by considering uniform edge lengths and ones considering the visual worth. The one constructed by considering the visual worth dedicates larger spaces for popular tourist areas.

5.4.3 Map Warping

In the prior section, we use the MIP optimization process to compute the octilinear layout of the given metro network. These deformed metro layouts, unfortunately, cannot be visualized together with the original coordinates of POIs. One can treat POIs as labels or other attributes associated with stations or vertices of the metro network graph, and compute their locations within the octilinear layout. We found, however, that this approach requires prohibitively excessive amount of computation time and even fails given its high time complexity, since the number of POIs can be very high (e.g., hundreds of POIs).

Instead of handling them within the MIP optimization, we adopt another step that maps POIs according to the computed octilinear layout. To map those elements into a space aligned with the computed octilinear layout, we



(a) Default zoom level (b) Zoomed-in view

Figure 5.7: [This figure shows our maps for Prague with different zoom levels.]

utilize the Helmert transformation, a popular transformation method in geodesy, which is essentially a 4-parameter Euclidean transformation [50].

Given the Helmert transformation, we use the newly computed station positions as control points for transforming the rest of the map elements such as POI positions, city outline polygon vertices, etc. Positions of the input control points in the octilinear layout, $\bar{x}_i = \bar{x}(u)$ and $\bar{y}_i = \bar{y}(u)$, and their original positions, $x_i = x(u)$ and $y_i = y(u)$, are used to compute the four unknown parameters representing 2D translation (t_x and t_y), 1D scaling (s), and 1D rotation (θ):

$$\begin{bmatrix} \bar{x}_i \\ \bar{y}_i \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \begin{bmatrix} s\cos\theta & -s\sin\theta \\ s\sin\theta & s\cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}.$$

Given the linear system, we use a common least-squares method for computing the unknown parameters.

The next step after computing the transformation matrix for each vertex of the metro network is to construct an interpolation scheme for transforming POIs into the octilinear layout. For this purpose we adopt a multiquadric interpolation [82] for achieving high-quality interpolation. Specifically, we define our interpolation equation to be a quadric basis function:

$$\bar{x}_i - x_i = \sum_{j=1}^{n} \alpha_j \left((\bar{x}_i - \bar{x}_j)^2 + (\bar{y}_i - \bar{y}_j)^2\right)^{-1/2}, \tag{5.5}$$

$$\bar{y}_i - y_i = \sum_{j=1}^{n} \beta_j \left((\bar{x}_i - \bar{x}_j)^2 + (\bar{y}_i - \bar{y}_j)^2\right)^{-1/2}, \tag{5.6}$$

where n is the number of control points. We estimate unknown coefficients α and β based on the least-squares method.

Once we setup our interpolation equations, we then compute discrepancies, $\bar{p} - p$ and $\bar{q} - q$, for any point, (p,q), in the original map. In other words, we use the following equations to compute transformed positions $(\bar{p},\bar{q})$ given the transformed metro layout:

$$\bar{p} - p = \sum_{j=1}^{n} \alpha_j \left((\bar{p} - \bar{x}_j)^2 + (\bar{q} - \bar{y}_j)^2\right)^{-1/2},$$

$$\bar{q} - q = \sum_{j=1}^{n} \beta_j \left((\bar{p} - \bar{x}_j)^2 + (\bar{q} - \bar{y}_j)^2\right)^{-1/2}.$$

We visualize this transformation by showing a regular mesh and its deformed mesh in the original and octilinear layout space in Fig. 5.6.

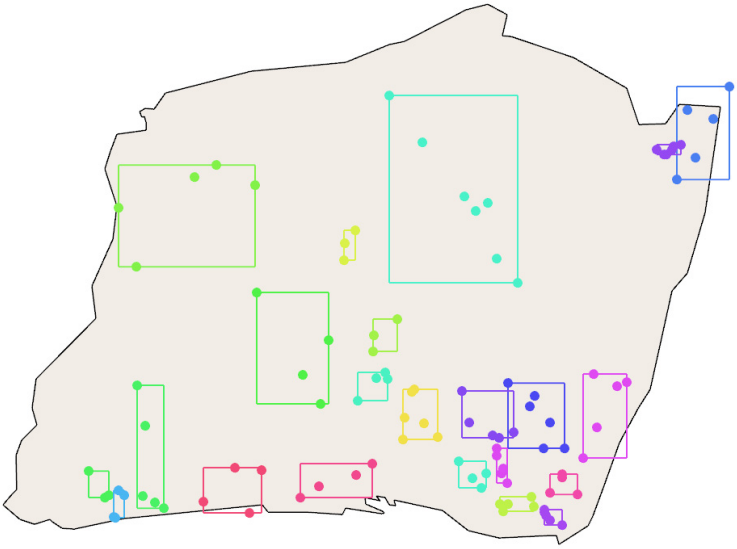


Figure 5.8: [This figure shows our clustering result at an intermediate level.]

5.4.4 Hierarchical Clustering

Displaying all POI photos can clutter the visualization result and overload users with a lot of information. Instead, we summarize the collection by computing and showing only representative images. To realize this, we use a hierarchical clustering algorithm, namely the Hungarian clustering method [54].

The Hungarian clustering method performs clustering based on the cycle cover problem [54]. In this clustering method, we start with the POI positions as inputs. We then perform the following procedures. *a*) These points are treated as initial clusters. *b*) For each cluster, we construct an edge from the cluster to the nearest cluster. We then solve a minimal cycle cover for the computed graph, resulting in a set of cycles that are treated as clustering. *c*) This is repeated for the resulting clusters in the prior step, until we cannot merge clusters anymore.

This recursive process results in a clustering tree, where each node represents a cluster, and the leaves contain individual POIs. The resulting hierarchical clustering tree is used to display a portion of the POI collection.

Runtime. At runtime, a user can zoom-in/out and translate the visualization window that shows the full or a portion of our map (Fig. 5.7). Given a chosen visualization window, we choose and visualize biggest POI clusters whose size is smaller than that of the window and is contained in the window. Fig. 5.8 shows chosen clusters given an example visualization window. To compute such clusters, defined as *active clusters*, we traverse the clustering tree from the root in the top-down manner. When we visit clusters satisfying the criteria, we treat them as active clusters and terminate the traversal from those clusters.

There can be too many active clusters that cannot be visualized given the visualization window. We therefore choose top M clusters. For ordering active clusters, c_i, we compute a prominence of cluster, $p(c_i)$:

$$p(c_i) = \frac{vw(c_i)}{\sum_{j=1}^{n} vw(c_j)},$$

where $vw(c_j)$ is the visual worth of the cluster c_j, which is defined as the biggest visual worth of POIs contained in the cluster. Additionally, we display the next M POIs in a smaller size to give previews of other interesting destinations. We have found that $M = 10$ works well for our purpose without cluttering the visualization. An accompanying video shows an example of using our interactive framework.

In our implementation, we only use a POI's geographic coordinate, which is assigned by tourist websites,

City	# of Stations	# of Edges	# of POIs
Lisbon	49	51	120
Prague	54	54	120
Vienna	90	96	120

Table 5.1: [This table shows the characteristics of our tested benchmarks.]

as a single point to locate it in the map. Unfortunately, this would mean that if the point is outside the viewing window, the POI would not be displayed, even if the actual POI area is bigger. This can be addressed if a bounding box of each POI is available.

5.5 Results and Analysis

We have implemented our methods on a machine with a quad-core i7 3.6GHz CPU. The MIP solver used is Gurobi Optimizer (*gurobi.com*).

We have tested three benchmarks: Prague, Lisbon and Vienna data sets as shown in Fig. 5.7, 5.1, and 5.3, respectively. The characteristics of these three data sets are summarized in Table 5.1. The total preprocessing time of the visual worth computation, the MIP optimization, warping, and clustering is $0.52s$, $3.22s$ and $12.35s$ for Prague, Lisbon, and Vienna, respectively. In the visual worth computation, we set $w_r = 0.7$ and $w_\rho = 0.3$, as we have found effective based on our design factors **DF5** and **DF6**, which are discussed below. Our runtime computation that computes the most prominent, active clusters takes less than 1 ms.

5.5.1 Design Factors and User Study

We have designed and conducted our study to isolate design factors. Their summary is shown in Fig. 5.9. In order to assess our design factors, we have conducted a user study through the actual use of our interactive map prototype. We have surveyed 70 individuals: 38 females and 32 males, 16 to 35 years old. In addition to designing related questions, we have inquired about their trip planning habits. We screened participants in such a way that they actually use metro-transit in their trips and are not well acquainted in the tested cities.

1. **(DF1)** Geographic vs. our map: we showed the users a geographically accurate metro map and our octilinear layout metro map, both with all the POI images. More than half of the users prefer using our map in terms of looking at a metro network and POI images. We presume that our approach might be uncomfortable to users without getting accustomed to it.

2. **(DF2)** All vs. representative photos: we showed the users our map filled with all POIs in the city and with representative POIs. We find that more users like the map with representative images. All the POIs displayed at once clutter the screen, while representative images give a clean organized look.

3. **(DF3)** Representative photos vs. representative photos+points: we showed the users our map with representative images, and with representative images and non-representative POIs displayed as small points. Users like to see what they might have been missing out, so the latter one is preferred.

4. **(DF4)** Uniform (Fig.5.6-(b)) vs. variable (Fig.5.6-(c)) edge lengths: we showed the users our map with the uniform minimum edge length and with variable minimum edge lengths. Users are slightly leaning towards variable edge lengths. As for more dense tourist areas, the variable edge factor is more valuable. This deserves further exploring in more visual perception studies.

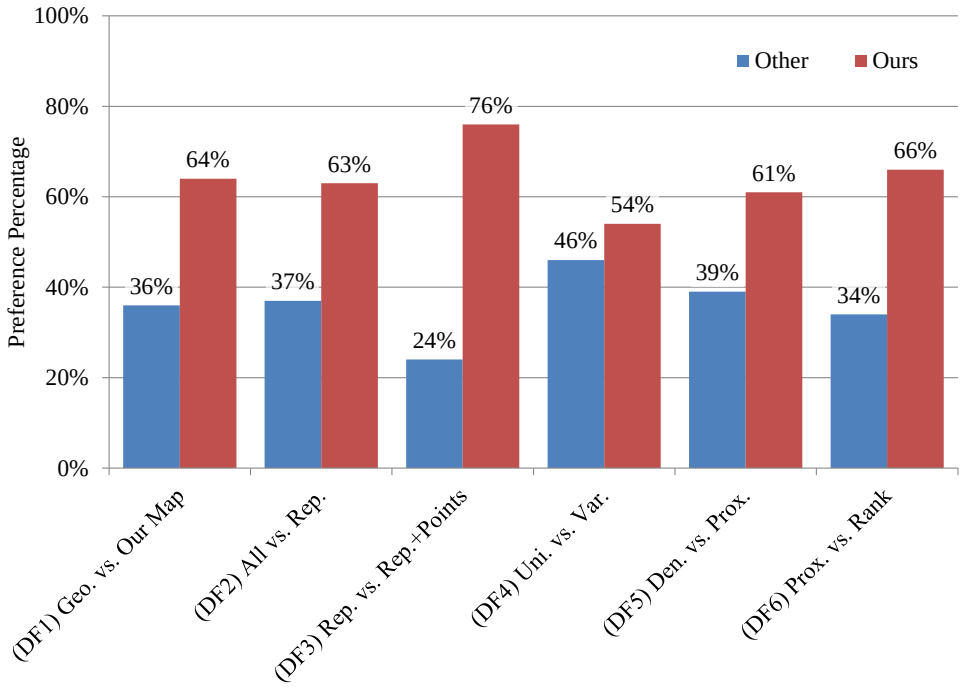


Figure 5.9: [Design factors study results.]

5. **(DF5)** Density vs. proximity: we showed the users our map with a small set of POIs near the metro, and with many POIs including ones far from the metro. More users prefer to visit and see POIs nearby a metro.

6. **(DF6)** Proximity vs. ranking: we showed the users our map with a region containing far but high ranked POIs, and with another region containing near but low ranked POIs. More users prefer the high ranking tourist areas, even though it is far from the metro.

We find that significantly more participants picked ours ($p\text{-}value < 0.05$), and in particular, DF3 and DF6 trials have highly significant results ($p\text{-}value < 0.01$). However, the DF4 case has no deducible statistical significance ($p\text{-}value = 0.20$).

Observations from DF5 and DF6 became our basis for defining weights used in our visual worth computation (Sec. 5.4.1). All the other design factors have been incorporated in our method described in the prior section.

Visual Worth. Considering visual worth permits flexibility to distort the map according to multiple spatial factors. By computing visual worth with the density estimation formulation, we can naturally support proximity and ranking. It can be extended to support other factors such as POI genre, to magnify certain map areas in a different manner.

In our tested user study, many users took a while in choosing between uniform and variable edge lengths, and 54% of them selected variable edge lengths. We believe this as a result of a non-optimized parameter setting for considering proximity and ranking. Optimizing them could improve the visual worth, including the lengths of edges of the metro. Further visual cognition studies, in particular eye tracking, would greatly assist in tuning these parameters.

User Satisfaction. We have also let our users to evaluate our prototype. Fig. 5.10 shows the user satisfaction rating with 99% confidence intervals. Most users agreed that our map features the clarity of information and our map avoids complexity brought by overloaded information. Users find that our map would help them plan a tour-by-metro well. While our work is still a prototype, many rate our UI highly on ease of use.

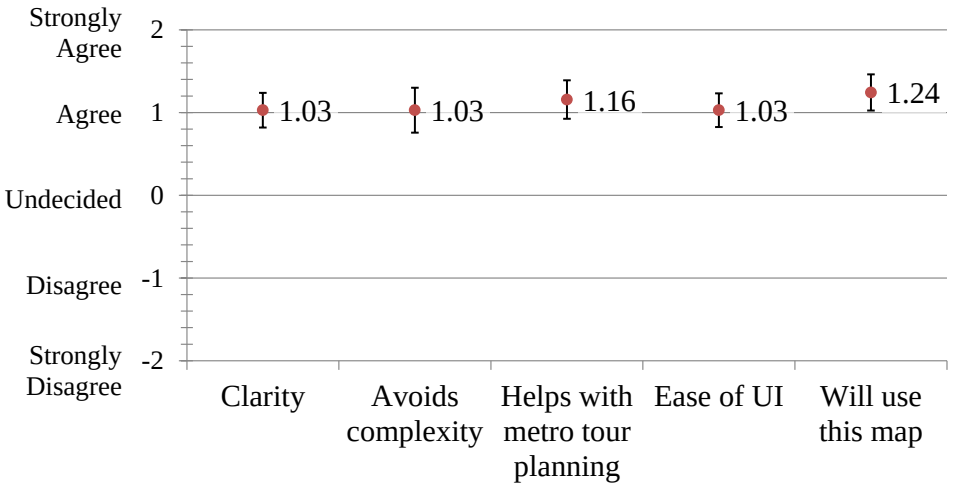


Figure 5.10: [Average user satisfaction rating with 99% confidence intervals.]

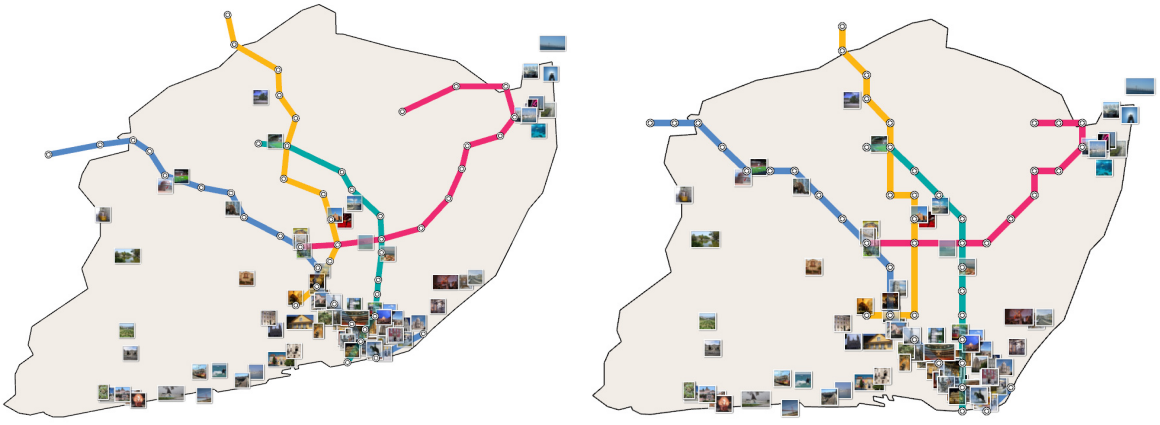
Limitations. Our method has drawbacks, although it has been demonstrated to show benefits for our purpose. Our approach for visualizing annotations of POIs is *on-site*, exactly placing POIs where they are located. This approach results in unavoidable overlaps with the metro stations and edges. Our current MIP optimization is reasonably fast, but its layouts can be complex and produce edge crossings. We withhold displaying labels of stations and POIs in order to allocate more space for the metro and representative photos. Since our work provides interactive map visualization, we can conveniently show pop-up labels upon selection of a station or a POI. Our method also inherits the limitations of our employed techniques. Nollenburg's method [1] produces high quality globally optimized metro layouts, yet it has not been demonstrated to be applicable for complex metro networks such as Tokyo, New York or Paris.

5.6 Conclusion

We have proposed an interactive approach for supporting tourists to effectively identify popular tourist destinations and travel to them based on the metro network. For our holistic visualization of city trip planing, we have used visual worth computation, MIP-based octilinear layout computation combined in showing representative POIs in the layout. Additionally, we have used a hierarchical clustering method to efficiently compute representative images given a user-specified visualization window and zoom level. To validate our proposed method and factor out important design criteria that are basis for finalizing our method, we have conducted a user study, which confirmed the benefits of our method.

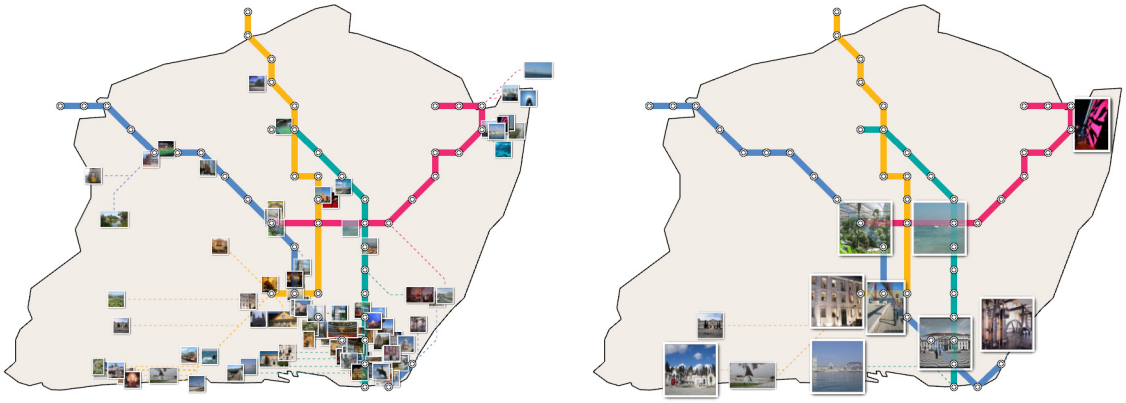
We have found that the proposed direction for city tour planning is promising and deserves further studies. As more geo-data is collected (location check-in, geo-tagged photos, etc.), more advanced interfaces are necessary to help tourists in planning their trips. In addition, the continued increase of smart mobile devices adaption will contribute to changing tour planning, and further trigger high demand on interactive tour guides for modern tourists [83]. The next logical improvement to our current system are paths to the POIs, which would require solving a different set of difficult challenges such as road simplification and layouting.

5.7 Supplemental Images



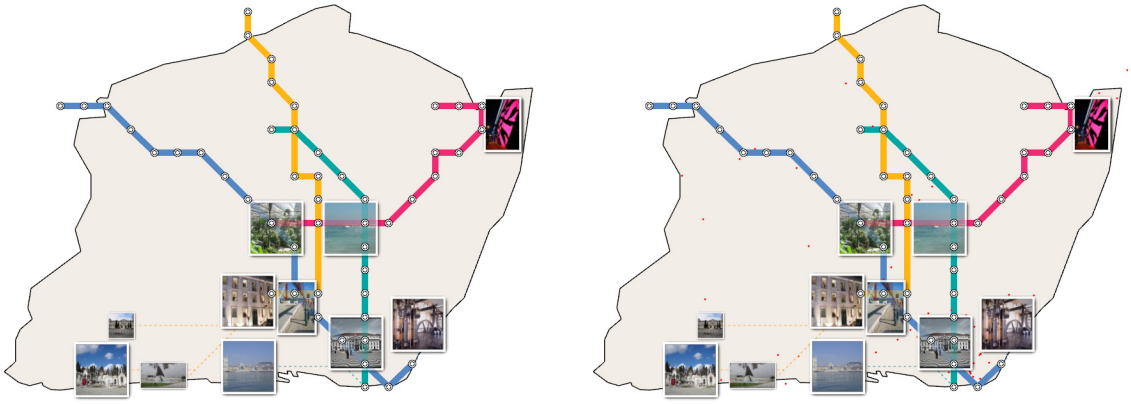
(a) Geographic (b) Octilinear

Figure 5.11: [Geographic vs. Octilinear Layout]



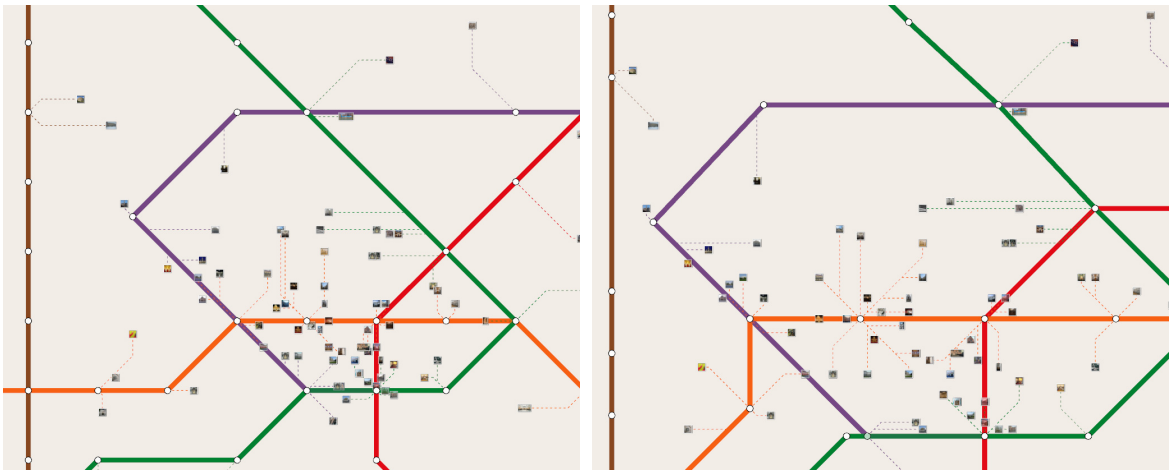
(a) All (b) Representative

Figure 5.12: [All vs. Representative Photos]



(a) Representative (b) Representative Photos + Points

Figure 5.13: [Representative Photos vs. Representative Photos + Points]



(a) Uniform (b) Variable

Figure 5.14: [Uniform vs. Variable]

Chapter 6. Conclusion

6.1 Limitations

Considering the scope and the benefits of the methods utilized, this work also comes with limitations as with any other approach.

The limitations of using modeling as an optimization problem applies also to this work. Arriving to a solution is not guaranteed. A way to address this is by fall back to the previous layout which may be the same as the original. Also, complete collision avoidance can influence the deformation to diverge from the optimal solution by satisfying that constraint. Allowing a tolerance value, especially for grids with coarse contents, can improve quality. Also by applying mass-spring methods with small steps can improve the fidelity.

In some cases there could be some overhead in using our method in low resolutions because of the additional support edges. However, the difference is small in absolute terms and goes away as the resolution increases. The benefit is seen in the scalability of the approach.

6.2 Future work

The methods described could be further enhanced upon application to smart cities. Availability of real-time data to real-time visualization would greatly benefit users, especially mobile tourists. In the area of personal urban mobility, map visualization is essential in giving timely information to guide users on the move. There are several possible research directions that can be taken.

1. **Time-sensitive POI discovery.** With the rise of online social activity, development of trends can be observed real-time. Information on current concerts, events, and happenings can be collected and aggregated to form new insights. These can be applied to POI discovery map applications that would suggest (mobile) users of latest during-the-moment trends that they should not miss.

2. **Smart route recommendation.** In addition to online social media, a smart city with sensors deployed around the city can bring convenience to travelers. Weather or traffic reports would affect the displayed routes real-time. Crowded stations or areas would be avoided in touristic routes in order to give users a relaxed travel. Connected bus and metro tracking services to display closest transfer trips would lessen idle time for travelers.

3. **Task-based map warping.** A full application demonstrating dynamic map morphing. Depending on a given task, a specific map will be displayed. While on a metro train, an octilinear map can be displayed, which can transition into a geographic road map to look for road directions. Depending on map search queries, certain buildings can be magnified or popped-out in 3D. Depending on map data, a cartogram can display intuitive value comparisons.

4. **Gamification.** Recent boom of augmented reality games like Pokemon GO showed that gadget users are willing to perform physical activity when presented virtual incentives. Applying gamification to tourist walking tours could boost tourism in participating regions and could also introduce less traveled paths to a new audience.

6.3 Conclusion

Creating an effective tourist map would assist a traveler in touring of a city: learn local spatial information, discover POIs and plan a route accordingly. Instead of going around using traditional paper tourist maps, an array of enhancements can be experienced through using a dynamic digital map that employs a combination of approaches. Additionally, map personalization that demands real-time task-dependent map representations require deformation techniques mainly based on uniform grids which can be insufficient for such real-time demands. For higher flexibility of allocating resolutions resulting in higher performance, a content-aware adaptive grid is proposed.

An initial investigation on adaptive approach on deforming models is performed in the context of view-dependent rendering integrated with occlusion culling for large-scale crowd scenes. In order to provide varying resolutions on each animated, articulated model, it is proposed to use a cluster hierarchy in the view-dependent representation. The cluster hierarchy is constructed by adaptively clustering regions with similar significance values in terms of simplification quality. This results in a high quality simplification that can achieve interactive performance at run-time.

Lessons from deforming 3D models are applied to map deformation. The proposed content-aware non-uniform grid leads to adaptive resolutions on the map deformation. Finer subdivisions on more significant regions such as guide road patterns, and coarser subdivisions with less quad edges on less significant regions are applied. This adaptive approach results in both higher road deformability and performance.

Finally, a dynamic and interactive framework that holistically combines presentations of POIs and a metro network is proposed. The idea is to identify popular POIs based on visual worth computation, and to introduce POI discovery for effectively identifying POIs within reach of a metro network for users. Octilinear layouts are used to highlight the metro network, and representative POI images are shown in the layout space visualized within a user-specified viewing window.

As more map related data become more available and a ubiquity of smaller display in mobile devices, the approaches proposed would be more apt in further development of dynamic and interactive tourist maps. In addition, when augmented reality becomes prevalent in mobile devices, 3D maps would be more utilized and would require more efficient deformation.

Bibliography

[1] M. Nollenburg and A. Wolff, "Drawing and labeling high-quality metro maps by mixed-integer programming," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 5, pp. 626–641, 2011.

[2] L. Yan and M. Y. Lee, "Are tourists satisfied with the map at hand?," *Current Issues in Tourism*, vol. 18, no. 11, pp. 1048–1058, 2015.

[3] J. Jones, B. Seefeld, P. Hofmann, Z. Bailiang, W. Van Lancker, A. Leicht, and T. Campbell, "Apparatus and method for personalizing maps," Apr. 3 2014. US Patent App. 13/632,938.

[4] A. Jaffe, M. Naaman, T. Tassa, and M. Davis, "Generating summaries and visualization for large collections of geo-referenced photographs," in *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, MIR '06, (New York, NY, USA), pp. 89–98, ACM, 2006.

[5] D. Yamamoto, S. Ozeki, and N. Takahashi, "Focus+glue+context: An improved fisheye approach for web map services," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, (New York, NY, USA), pp. 101–110, ACM, 2009.

[6] J.-H. Haunert and L. Sering, "Drawing road networks with focus regions," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, pp. 2555–2562, Dec 2011.

[7] F. Grabler, M. Agrawala, R. W. Sumner, and M. Pauly, "Automatic generation of tourist maps," in *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pp. 100:1–100:11, ACM, 2008.

[8] M. Birsak, P. Musialski, P. Wonka, and M. Wimmer, "Automatic generation of tourist brochures," *Computer Graphics Forum*, vol. 33, no. 2, pp. 449–458, 2014.

[9] B. Tversky, "Distortions in cognitive maps," *Geoforum*, vol. 23, no. 2, pp. 131 – 138, 1992.

[10] T. C. van Dijk and J.-H. Haunert, "Interactive focus maps using least-squares optimization," *International Journal of Geographical Information Science*, vol. 28, no. 10, pp. 2052–2075, 2014.

[11] M. Agrawala and C. Stolte, "Rendering effective route maps: Improving usability through generalization," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, (New York, NY, USA), pp. 241–249, ACM, 2001.

[12] J. Kopf, M. Agrawala, D. Bargeron, D. Salesin, and M. Cohen, "Automatic generation of destination maps," in *ACM SIGGRAPH Asia 2010 Papers*, SIGGRAPH ASIA '10, (New York, NY, USA), pp. 158:1–158:12, ACM, 2010.

[13] S.-S. Lin, C.-H. Lin, Y.-J. Hu, and T.-Y. Lee, "Drawing road networks with mental maps," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 20, pp. 1241–1252, Sept 2014.

[14] A. Wolff, "Drawing subway maps: A survey," *Informatik - Forschung und Entwicklung*, vol. 22, no. 1, pp. 23–44, 2007.

[15] S.-H. Hong, D. Merrick, and H. A. D. do Nascimento, "Automatic visualisation of metro maps," *J. Vis. Lang. Comput.*, vol. 17, pp. 203–224, June 2006.

[16] J. Stott, P. Rodgers, J. Martinez-Ovando, and S. Walker, "Automatic metro map layout using multicriteria optimization," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 1, pp. 101–114, 2011.

[17] C. Binucci, W. Didimo, G. Liotta, and M. Nonato, "Orthogonal drawings of graphs with vertex and edge labels," *Computational Geometry*, vol. 32, no. 2, pp. 71 – 114, 2005.

[18] H.-Y. Wu, S. Takahashi, C.-C. Lin, and H.-C. Yen, "Travel-route-centered metro map layout and annotation," *Comp. Graph. Forum*, vol. 31, pp. 925–934, June 2012.

[19] H.-Y. Wu, S. Takahashi, D. Hirono, M. Arikawa, C.-C. Lin, and H.-C. Yen, "Spatially efficient design of annotated metro maps," *Computer Graphics Forum*, vol. 32, no. 3pt3, pp. 261–270, 2013.

[20] P. Claudio and S.-E. Yoon, "Metro transit-centric visualization for city tour planning," *Computer Graphics Forum*, vol. 33, no. 3, pp. 271–280, 2014.

[21] D. F. Reilly and K. M. Inkpen, "Map morphing: Making sense of incongruent maps," in *Proceedings of the 2004 Graphics Interface Conference*, GI '04, (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada), pp. 231–238, Canadian Human-Computer Communications Society, 2004.

[22] J. Bottger, U. Brandes, O. Deussen, and H. Ziezold, "Map warping for the annotation of metro maps," *IEEE Computer Graphics and Applications*, vol. 28, no. 5, pp. 56–65, 2008.

[23] R. Pajarola and E. Gobbetti, "Survey of semi-regular multiresolution models for interactive terrain rendering," *The Visual Computer*, vol. 23, no. 8, pp. 583–605, 2007.

[24] P. S. Heckbert and M. Garland, "Survey of polygonal surface simplification algorithms," tech. rep., DTIC Document, 1997.

[25] Q. Bouts, T. Dwyer, J. Dyke, B. Speckmann, S. Goodwin, N. Riche, S. Carpendale, and A. Liebman, "Visual encoding of dissimilarity data via topology-preserving map deformation," *Visualization and Computer Graphics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[26] P.-Y. Laffont, J. Y. Jun, C. Wolf, Y.-W. Tai, K. Idrissi, G. Drettakis, and S. eui Yoon, "Interactive content-aware zooming," in *Graphics Interface*, 2010.

[27] P. Claudio, D. Kim, T.-J. Kim, and S.-e. Yoon, "Vdr-am: View-dependent representation of articulated models," *Journal of WSCG*, vol. 21, no. 3, 2013.

[28] P. Claudio and S.-E. Yoon, "A content-aware non-uniform grid for fast map deformation." Submitted.

[29] M. Garland and P. Heckbert, "Surface simplification using quadric error bounds," *ACM SIGGRAPH*, pp. 209–216, 1997.

[30] S. Kircher and M. Garland, "Progressive multiresolution meshes for deforming surfaces," in *Symp. on Computer animation*, pp. 191–200, 2005.

[31] F.-C. Huang, B.-Y. Chen, and Y.-Y. Chuang, "Progressive deforming meshes based on deformation oriented decimation and dynamic connectivity updating," in *ACM Symp. on Computer Animation*, pp. 53–62, 2006.

[32] A. Mohr and M. Gleicher, "Deformation sensitive decimation," *University of Wisconsin Graphics Group. Technical Report*, 2003.

[33] C. DeCoro and S. Rusinkiewicz, "Pose-independent simplification of articulated meshes," in *Symp. on Interactive 3D Graphics*, pp. 17 – 24, 2005.

[34] E. Landreneau and S. Schaefer, "Simplification of articulated meshes," *Computer Graphics Forum*, pp. 347–353, 2009.

[35] S. Pilgrim, A. Steed, and A. Aguado, "Progressive skinning for character animation," *Computer Animation and Virtual Worlds*, vol. 18, no. 4-5, pp. 473–481, 2007.

[36] S. Dobbyn, J. Hamill, K. O'Conor, and C. O'Sullivan, "Geopostors: a real-time geometry/impostor crowd rendering system"," *ACM Transactions on Graphics*, vol. 24, pp. 933–933, July 2005.

[37] L. Kavan, S. Dobbyn, S. Collins, J. Zara, and C. O'Sullivan, "Polyposters: 2d polygonal impostors for 3d crowds," in *ACM Symp. on Interactive 3D Graphics and Games*, pp. 149–155, 2008.

[38] K. Yuksel, A. Yucebilgin, S. Balcisoy, and A. Ercil, "Real-time feature-based image morphing for memory-efficient impostor rendering and animation on gpu," *The Visual Computer*, vol. 29, pp. 131–140, 2013.

[39] S.-E. Yoon, E. Gobbetti, D. Kasik, and D. Manocha, *Real-Time Massive Model Rendering*. Morgan & Claypool Publisher, 2008.

[40] H. Hoppe, "View dependent refinement of progressive meshes," in *ACM SIGGRAPH*, pp. 189–198, 1997.

[41] S.-E. Yoon, B. Salomon, R. Gayle, and D. Manocha, "Quick-VDR: Interactive View-dependent Rendering of Massive Models," in *IEEE Visualization*, pp. 131–138, 2004.

[42] D. Cohen-Or, Y. Chrysanthou, C. Silva, and F. Durand, "A survey of visibility for walkthrough applications," *IEEE Tran. on Visualization and Computer Graphics*, vol. 9, pp. 412–431, 2003.

[43] S. Yoon, B. Salomon, and D. Manocha, "Interactive view-dependent rendering with conservative occlusion culling in complex environments," in *Proc. of IEEE Visualization*, 2003.

[44] L. P. Chew, "Guaranteed-quality mesh generation for curved surfaces," in *Proceedings of the Ninth Annual Symposium on Computational Geometry*, SCG '93, (New York, NY, USA), pp. 274–280, ACM, 1993.

[45] R. Sivan and H. Samet, "Algorithms for constructing quadtree surface maps," in *Proc. 5th Int. Symposium on Spatial Data Handling*, pp. 361–370, 1992.

[46] K.-J. Choi and H.-S. Ko, "Research problems in clothing simulation," *Computer-Aided Design*, vol. 37, no. 6, pp. 585–592, 2005.

[47] D. Hutchinson, M. Preston, and T. Hewitt, "Adaptive refinement for mass/spring simulations," in *7th Eurographics Workshop on Animation and Simulation*, pp. 31–45, 1996.

[48] X. Zhang, C. Bajaj, and W. Blanke, "Scalable isosurface visualization of massive datasets on cots clusters," in *IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, 2001.

[49] Y. Lee, S.-E. Yoon, S. Oh, D. Kim, and S. Choi, "Multi-resolution cloth simulation," *Computer Graphics Forum (Pacific Graphics)*, vol. 29, no. 7, pp. 2225–2232, 2010.

[50] B. Jenny and L. Hurni, "Cultural heritage: Studying cartographic heritage: Analysis and visualization of geometric distortions," *Comput. Graph.*, vol. 35, pp. 402–411, Apr. 2011.

[51] T. Keahey and E. Robertson, "Techniques for non-linear magnification transformations," in *Information Visualization '96, Proceedings IEEE Symposium on*, pp. 38–45, 1996.

[52] M. Sarkar and M. H. Brown, "Graphical fisheye views of graphs," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, (New York, NY, USA), pp. 83–91, ACM, 1992.

[53] Y.-S. Wang and M.-T. Chi, "Focus+context metro maps," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2528–2535, 2011.

[54] J. Goldberger and T. Tassa, "A hierarchical clustering algorithm based on the hungarian method," *Pattern Recogn. Lett.*, vol. 29, pp. 1632–1638, Aug. 2008.

[55] D. Thalmann, C. O'Sullivan, B. Yersin, J. Maim, and R. McDonnell, "Populating virtual environments with crowds," in *Eurographics Tutorial*, 2007.

[56] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," in *SIGGRAPH Asia*, pp. 1–8, 2009.

[57] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner, *Level of Detail for 3D Graphics*. Morgan-Kaufmann, 2002.

[58] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha, "Iccd: Interactive continuous collision detection between deformable models using connectivity-based culling," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, no. 4, pp. 544 –557, 2009.

[59] G. Ryder and A. M. Day, "Survey of real-time rendering techniques for crowds," *Computer Graphics Forum*, vol. 24, no. 2, pp. 203–215, 2005.

[60] S. Rusinkiewicz and M. Levoy, "Qsplat: A multiresolution point rendering system for large meshes," *SIGGRAPH*, pp. 343–352, 2000.

[61] J. P. Charalambos, J. Bittner, M. Wimmer, and E. Romero, "Optimized hlod refinement driven by hardware occlusion queries," in *Advances in Visual Computing*, pp. 106–117, Springer, Nov. 2007.

[62] V. E. McGee, "Multidimensionnal scaling of n sets of similarity measures : A nonmetric individual differences approach," *Multivariate Behavioral Research*, vol. 3, pp. 233–248, 1968.

[63] J. Assa, Y. Caspi, and D. Cohen-Or, "Action synopsis: pose selection and illustration," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 667–676, 2005.

[64] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 19, pp. 509–517, 1975.

[65] J. Zelsnack, "Glsl pseudo-instancing," *Tech. rep., NVIDIA Corporation*, 2004.

[66] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surface," *Computer Graphics Forum*, pp. 167–174, 1998.

[67] C. Erikson, D. Manocha, and B. Baxter, "Hlods for fast display of large static and dynamic environments," *Proc. of ACM Symposium on Interactive 3D Graphics*, 2001.

[68] R. Rodriguez, E. Cerezo, S. Baldassarri, and F. J. Seron, "New approaches to culling and lod methods for scenes with multiple virtual actors," *Computers and Graphics*, vol. 34, no. 6, pp. 729 – 741, 2010.

[69] D. L. James and C. D. Twigg, "Skinning mesh animations," *ACM Trans. on Graphics (SIGGRAPH)*, vol. 24, no. 3, 2005.

[70] S. Yoon, B. Salomon, M. C. Lin, and D. Manocha, "Fast collision detection between massive models using dynamic simplification," in *Eurographics Symposium on Geometry Processing*, pp. 136–146, 2004.

[71] A. Ballatore and M. Bertolotto, "Personalizing maps," *Communications of the ACM*, vol. 58, no. 12, pp. 68–74, 2015.

[72] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pp. 381–389, ACM, 2006.

[73] B. Von Herzen and A. H. Barr, "Accurate triangulations of deformed, intersecting surfaces," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pp. 103–110, 1987.

[74] G. Taubin, "Linear anisotropic mesh filtering," *Res. Rep. RC2213 IBM*, vol. 1, p. 4, 2001.

[75] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. Philadelphia, PA: SIAM, 1994.

[76] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," 1994.

[77] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, pp. 40–53, Mar. 2008.

[78] B. Tiddeman, N. Duffy, and G. Rabey, "A general method for overlap control in image warping," *Computers & Graphics*, vol. 25, no. 1, pp. 59 – 66, 2001. Shape Blending.

[79] T. Dwyer and G. Robertson, *Graph Drawing: 17th International Symposium, GD 2009, Chicago, IL, USA, September 22-25, 2009. Revised Papers*, ch. Layout with Circular and Other Non-linear Constraints Using Procrustes Projection, pp. 393–404. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[80] T. Liu, A. W. Bargteil, J. F. O'Brien, and L. Kavan, "Fast simulation of mass-spring systems," *ACM Trans. Graph.*, vol. 32, pp. 214:1–214:7, Nov. 2013.

[81] G. R. Terrell and D. W. Scott, "Variable kernel density estimation," *The Annals of Statistics*, pp. 1236–1265, 1992.

[82] R. Hardy, "Theory and applications of the multiquadric-biharmonic method 20 years of discovery 1968-1988," *Computers & Mathematics with Applications*, vol. 19, no. 8-9, pp. 163–208, 1990.

[83] D. Wang and D. Fesenmaier, "Transforming the travel experience: The use of smartphones for travel," in *Information and Communication Technologies in Tourism 2013* (L. Cantoni and Z. P. Xiang, eds.), pp. 58–69, Springer Berlin Heidelberg, 2013.

Acknowledgment

Curriculum Vitae

Name : Claudio, Benjamin Pio

E-mail : brclaudio@kaist.ac.kr

Educations

2002. 6. – 2006. 4. University of the Philippines (B.S.)

2008. 2. – 2010. 8. KAIST (M.S.)

Academic Activities

1. **Pio Claudio** and Sung-Eui Yoon, *Tourist Map Visualization*, under submission, 2016.
2. **Pio Claudio** and Sung-Eui Yoon, *Metro Transit-Centric Visualization for City Tour Planning*, Eurovis 2014, Swansea Wales, UK, June, 2014.
3. **Pio Claudio** and Sung-Eui Yoon, *Octilinear Layouts for Metro Map Visualization*, 2014 International Conference on Big Data and Smart Computing (BIGCOMP), Bangkok, Thailand, January, 2014.

Publications

1. **Pio Claudio** and Sung-Eui Yoon, *A Content-Aware Non-Uniform Grid for Fast Map Deformation*, under submission, 2016.
2. **Pio Claudio** and Sung-Eui Yoon, *Metro Transit-Centric Visualization for City Tour Planning*, Computer Graphics Forum, June, 2014.
3. **Pio Claudio** and Sung-Eui Yoon, *View-Dependent Representation of Articulated Models*, WSCG 2013, Czech Republic, July, 2013.
4. B.C. Moon, Y.Y. Byun, T.J. Kim, **Pio Claudio**, H.S. Kim, Y.J. Ban, S.W. Nam, and S.E. Yoon, *Cache-Oblivious Ray Reordering*, ACM Transactions on Graphics, 2010.