UNIVERSITE DE LIMOGES

ECOLE DOCTORALE Science — Technologie — Santé

Faculté des Sciences et Techniques

XLIM Laboratoire UMR CNRS 6172

Thèse № [ ]

Thèse

pour obtenir le grade de
Docteur de l'Université de Limoges

Discipline : Informatique

présentée et soutenue par

Dmitry SOKOLOV

le 12 décembre 2006

Exploration différée de mondes virtuels sur Internet

Thèse dirigée par M. Dimitri PLEMENOS

Resumé

Dans le domaine de la synthèse d'image, beaucoup d'efforts ont été entrepris pour améliorer la qualité et le réalisme du rendu. Mais si les outils pour effectuer les rendus sont arrivés à maturité, il reste un travail important à réaliser sur l'interaction et la modélisation. En effet, habituellement, l'utilisateur doit choisir un point de vue *lui-même* pour mieux inspecter et même comprendre une scène. Le but du travail entrepris dans cette thèse était la mise en œuvre de nouvelles techniques pour l'exploration automatique différée de mondes virtuels. L'objectif de l'exploration différée est d'analyser la scène afin de permettre de contrôler la caméra virtuelle (c'est-à-dire donner des bons points de vue, créer des trajectoires etc.) en temps réel après l'analyse.

D'abord, nous présentons des fonctions de *bas niveau* (c'est-à-dire une fonction qui traite des éléments de bas niveau, tels que les triangles) pour évaluer la qualité d'un point de vue. Ensuite, nous introduisons une heuristique de *haut niveau* qui permet d'évaluer la qualité d'un point de vue en fonction de *la familiarité* de l'utilisateur avec les objets d'une scène et de *la fonctionnalité* des objets. Nous introduisons aussi deux techniques d'exploration *externe* de mondes virtuels. En créant des "films", ces techniques donnent une bonne compréhension d'un objet simple ou une compréhension générale d'une scène complexe. Nous presentons également de nouvelles techniques pour l'exploration *locale*. Ces techniques permettent à la caméra de se diriger à l'intérieur (ou bien à l'extérieur) d'une scène jusqu' à ce que la plupart de ses parties intéressantes soit visitée. Une nouvelle mesure de similitude entre les objets est également présentée dans ce mémoire. Elle est utile quand une connaissance additionnelle de la structure de la scène pourrait être fournie. Cette mesure, dite distance sémantique, évalue des relations dans la scène pour améliorer la méthode d'exploration.

Abstract

In Computer Graphics a lot of efforts are focused on improving quality and realism of renders, but rarely one focuses on the interaction and modeling. Indeed, usually, the user must choose viewpoints *himself* to better inspect and even to understand what a scene looks like. The goal of the work undertaken in this thesis was the developing of new techniques for offline automatic exploration of virtual worlds. The objective of an offline exploration is to analyze a scene in order to control the camera (i.e. to find good viewpoints, to create camera trajectories etc.) in real-time after the analysis.

First of all, we present new *low-level* functions for evaluating viewpoint quality (which process low-level data such as triangles). Next, we introduce a new *high-level* heuristic, which takes into account high-level data such as *predictability* (familiarity) of objects. Then we present two techniques of virtual world *external* exploration. Creating a movie, the proposed techniques allow to get a good comprehension of a single virtual artefact or a general comprehension of a scene. We also introduce a method of *local* scene exploration. It allows a camera to navigate *inside* (as well as outside) a model until most of interesting reachable places are visited. Finally, we present a new measure of similarity between objects. It is useful when some additional knowledge of scene structure could be provided. This measure, so called semantic distance, evaluates relationships in the scene to improve the exploration methods.

Acknowledgements

Completing this doctoral work has been a wonderful and often overwhelming experience. I have been very privileged to have undoubtedly the most intuitive, intelligent and supportive advisor anyone could ask for, namely Dimitri Plemenos. He is not only a great scientist with deep vision but also and most importantly a kind person. He has also known when (and how) to give me a little push in the forward direction when I needed it.

I am very grateful to Professors Mateu Sbert and Kadi Bouatouch for agreeing to referee my thesis, for their valuable feedback. The presence of their sharp and clear mind has been a great pleasure.

I would like also to thank Professors Michel Mériaux and Djamchid Ghazanfarpour for being members of my thesis jury and I want to express my appreciation to my thesis co-advisor Karim Tamine.

Finally I would like to express my gratitude towards the European Community and the Limousin Region for their financial support of the present work.

Table of Contents

Chapter 1

Introduction

De nos jours, la communication non verbale (surtout par images) est devenue très importante. La perception visuelle est rapide, facile, suffisamment riche et complète, et remporte les faveurs du public.

D’un point de vue graphique, l’évaluation de la recherche et des techniques a permis en quelques décennies de passer du simple rendu en fil de fer à des images tellement convaincantes que le néophyte ne peut deviner qu’il est en présence d’images virtuelles. Les représentations sont passées du simple objet à des salles complètes, pour ensuite évoluer vers des villes, voire des mondes complexes.

La réalité virtuelle est une possibilité bienvenue dans beaucoup de domaines. C’est la manière de montrer toutes les caractéristiques souhaitées — la forme et l´arrangement des objets, les relations entre eux, les qualités physiques — matérielles des objets, et aussi l´intention du créateur (visualisation artistique, etc.). Grâce à tout cela nous avons la possibilité de modéliser et de simuler des situations sans être obligés de faire de nombreux tests très coûteux ou impossibles dans la réalité. Il y a un grand nombre de domaines dans lesquels la visualisation virtuelle est ou peut être utilisée : médecine, architecture, cybernétique, sciences de l’ingénieur et aussi le jeu vidéo. Et bien évidemment les exigences sont différentes dans chaque domaine.

1.1 Motivation

Avoir un monde ou une scène en 3 dimensions est intéressant, mais inutile en l'absence de moyens de visualisation. Un élément primordial de ces mondes numériques est la notion de *caméra virtuelle*. Son principe est le même que celui de la caméra réelle, mais avec des propriétés plus développées et mieux adaptées à l'exploration et à la prise de connaissance de mondes virtuels. Encore est-il nécessaire de définir des propriétés à cette caméra virtuelle. Le plus important à ce niveau pour obtenir des résultats satisfaisants est la position de la caméra — le choix du bon point de vue, direction et prise de vue de la caméra et sa trajectoire de déplacement — et tout ça de préférence de manière automatique.

Dans le domaine de la synthèse d'image, beaucoup d'efforts ont été entrepris pour améliorer la qualité et le réalisme du rendu. Mais si les outils pour effectuer les rendus sont arrivés à maturité, il reste un travail important à réaliser sur l'interaction et la modélisation. En effet, habituellement, l'utilisateur doit choisir un point de vue *lui-même* pour mieux inspecter et même comprendre une scène.

Considérons la situation suivante : nous sommes arrivés à une ville complètement inconnue, une ville jamais vue, et nous voudrions l'observer. Sans cartes ou guides nous serons confus, beaucoup de temps sera gaspillé. La même situation se produit avec les mondes virtuels. Il y a une nécessité forte d'avoir des techniques rapides et précises pour une meilleure compréhension de divers mondes virtuels. Il serait très utile si un guide virtuel, un conseiller, pouvait être créé (voir la figure 1.1).

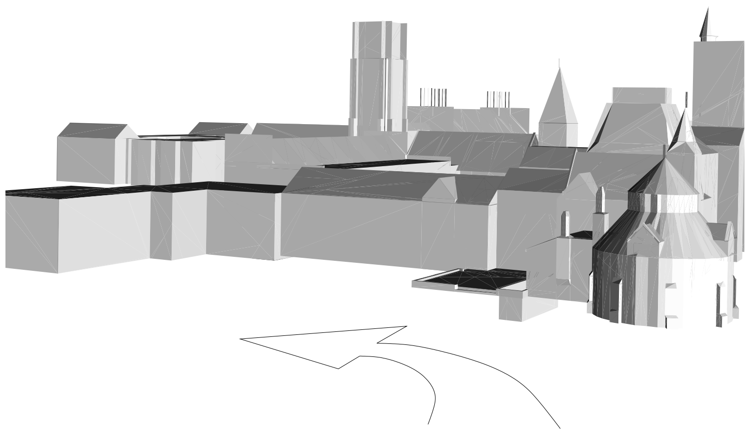


Figure 1.1: Un utilisateur explore une ville virtuelle, la flèche indique une direction recommandée, mais ne l'oblige pas à suivre l'itinéraire.

Dans ce chapitre, après avoir présenté la notion de contrôle de caméra, nous détaillons la problématique et nous fixons des objectifs pour notre travail.

1.2 Problématique

Nous définissons ici la notion de contrôle de caméra ainsi que les enjeux, les techniques et les applications qui y sont rattachés. Le terme contrôle de caméra est utilisé dans deux domaines distincts : la robotique et la synthèse d'image. Cependant, nous verrons que ces communautés partagent à la fois une problématique et des techniques de résolution similaires.

Dans le domaine de la robotique, le contrôle de caméra regroupe l'ensemble des modèles, méthodes et algorithmes permettant de calculer une position de caméra, dans un environnement réel ou virtuel. L'approche consiste généralement à déplacer la caméra dans un environnement parfois complexe, en conservant un certain nombre de propriétés valides à l'écran. Les techniques sont majoritairement issues de l'optimisation, c'est-à-dire minimiser une fonction qui permet, à partir d'une position donnée, d'atteindre un objectif. Les applications sont, par exemple, la détection et le suivi d'objets par une caméra (visual-servoing) ou le suivi d'un processus de fabrication grâce à un contrôle visuel (sensor-planning).

En synthèse d'images, le contrôle de caméra regroupe un ensemble de techniques et d'algorithmes permettant de manipuler les paramètres de la caméra, directement ou indirectement, automatiquement ou non, par le biais d'interfaces matérielles ou logicielles. Ici, la caméra est le médium de l'information par lequel l'utilisateur appréhende le monde réel ou virtuel et peut interagir avec celui-ci. Les techniques diffèrent suivant les objectifs, mais elles sont souvent basées sur des approches algébriques, des techniques d'optimisation et parfois de résolution de contraintes.

En synthèse d'images, nous dégageons trois utilisations importantes du contrôle de caméra :

1. **La prise de connaissance**, où la caméra est un médium qui permet d'appréhender le monde. Notre compréhension de ce monde est directement liée à la manière dont les informations nous sont transmises, c'est-à-dire la façon dont la caméra est positionnée et comment elle se déplace.

Cette prise de connaissance est dirigée par l'utilisateur à des degrés plus ou moins importants : complètement lorsqu'il s'agit de l'exploration d'environnements virtuels (jeux) et la visualisation de données, peu lorsque des outils calculent des positions et déplacements de caméra maximisant la pertinence des informations transmises (visualisation scientifique).

Dans le cadre du placement de caméra en temps réel, se profilent des applications pour l'industrie du jeu interactif 3D. L'objectif est de placer correctement la caméra afin de permettre à l'utilisateur d'appréhender son environnement à travers la disposition des objets et les différents évènements qui s'y produisent.

La difficulté majeure réside dans les problèmes de collisions avec l'environnement et d'occlusions (un objet en cache un autre). Mais avant de pouvoir placer correctement la caméra, il est nécessaire d'évaluer les évènements importants dans la scène et la manière de les restituer. Car il s'agit avant tout d'assister l'utilisateur dans la compréhension de l'environnement pour ainsi guider ses actions. Cette problématique se trouve renforcée avec l'émergence des jeux massivement multi-joueurs. Malgré les résultats obtenus dans les jeux, il reste un travail important à réaliser dans ce domaine.

2. **La manipulation de la caméra.** Comment interagir avec un environnement virtuel, c'est-à-dire sélectionner, déplacer, modifier des entités, si on ne peut visualiser correctement le travail effectué ? En contrôle de caméra, la problématique de la manipulation consiste à positionner et déplacer la caméra autour d'entités manipulées par l'utilisateur en créant des outils puissants et intuitifs. En partant du contrôle bas niveau avec modification directe des paramètres de la caméra et en passant par des outils tel l'arcball [Sho92] pour arriver à des techniques qui automatisent le placement de caméra en évitant les occlusions, les approches assistent l'utilisateur, de manière plus ou moins transparente, afin qu'il se consacre entièrement à sa tâche de manipulation.

3. **La construction de trajectoires de caméra.** Pour offrir des films de synthèse au cinéma, à la publicité ou la démonstration, il est nécessaire de créer des trajectoires de caméra dans des environnements 3D. Or, placer et déplacer une caméra afin d'offrir une composition photographique correcte, ou mettre en valeur certaines propriétés, est une tâche ardue. Elle requiert un impor-

> tant travail d'essai-erreur entre le positionnement de la caméra et le résultat désiré à l'écran. Il est donc naturel qu'un nombre important de travaux soient menés pour assister le créateur dans cette phase. L'étude bibliographique du chapitre suivant décrit et compare les diverses approches de la littérature scientifique. Que les méthodes soient basées sur des approches matérielles, des métaphores, des modèles physiques, des méthodes d'optimisation ou de résolution de contraintes, l'objectif est d'abstraire le modèle mathématique intrinsèque à la représentation de la caméra et de sa trajectoire afin d'offrir un outil de haut niveau permettant de créer aisément un ensemble de trajectoires. Concernant la spécification de trajectoires de caméra, les applications sont majoritairement destinées aux logiciels de modélisation 3D.

Dans tous ces cas, et dans bien d'autres, il est nécessaire de présenter automatiquement une vision de la scène. Il faut, pour cela, disposer d'outils puissants proposant une exploration basée sur le mouvement d'une caméra virtuelle.

Alors qu'aujourd'hui la majorité des modeleurs 3D se satisfont d'une approche bas-niveau pour contrôler la caméra, notre objectif est de proposer un outil de haut niveau afin d'assister l'utilisateur dans sa phase de création et de manipulation de trajectoires de caméra.

Quelles sont les principales difficultés des objectifs que nous nous sommes fixés pour spécifier et calculer des trajectoires de caméra? Nous les identifions rapidement avant de présenter l'état de l'art du contrôle de caméra, qui nous permettra d'avoir une idée générale des différentes approches possibles.

Pour assister à diverses tâches telles que la manipulation ou la mise en évidence de propriétés d'un objet ou d'une scène, il est nécessaire de calculer un bon point de vue. Dans le domaine de la manipulation directe en trois dimensions, le calcul d'un bon point de vue permet d'améliorer l'interactivité avec l'utilisateur.

Le problème des techniques existantes vient du fait que, soit les programmes génèrent une vue unique qui peut gêner a la compréhension (par exemple, l'illusion du cube de Necker, la figure 1.2), soit ils travaillent sous la contrainte du temps réel et ne prennent en compte que le plus petit élément de la scène (la facette) en générant une connaissance locale de la scène.

Assister le contrôle de caméra est aussi une tâche difficile car avec sept degrés de liberté pour une caméra statique, l'espace de recherche est très important.

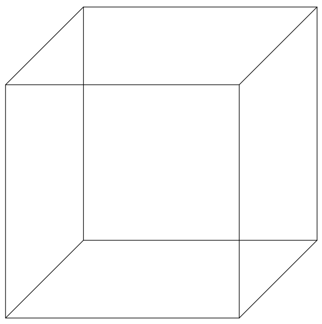


Figure 1.2: Cube de Necker : toutes les arêtes du cube sont d'égale épaisseur et aucune perspective n'est indiquée ; on peut voir la face avant de ce cube transparent dirigée, soit vers le haut à gauche, ou vers le bas à droite. Aucun élément ne permet de favoriser une interprétation, mais on passe assez aisément d'une vision à l'autre quoiqu'elles soient incompatibles.

La problématique de la planification de trajectoires (motion-planning) est sous-jacente aux problèmes de placement/déplacement de caméra. Or, malgré les nombreux travaux en robotique, planifier une trajectoire en sept dimensions reste à ce jour une tâche ardue.

En plus, les qualifications de bons points de vues sont bien évidemment dépendantes des applications. Par exemple, W. Bares [BL97, BZRL98] cherche à contrôler la caméra de manière à mettre l'accent sur le pouvoir pédagogique et explicatif d'une position de caméra dans un environnement virtuel. Une application dans laquelle des mécanismes cellulaires ont été modélisés et pour laquelle la caméra est utilisée comme médium pédagogique valide l'approche.

Plus précisément, notre objectif est l'exploration différée d'un monde virtuel par une caméra. En effet, on peut concevoir deux possibilités pour l'utilisateur qui découvre une scène sur internet et qui aimerait la comprendre.

1. **Prise de connaissance directe.** L'outil de prise de connaissance essaie d'explorer la scène directement, a partir des connaissances disponibles, sans analyse préalable de la scène. C'est le mode de prise de connaissance utilisé lorsque l'utilisateur souhaite explorer la scène immédiatement. Ce mode peut souffrir de manque de précision.

2. **Prise de connaissance différée.** Très souvent, l'utilisateur ne souhaite pas une exploration immédiate de la scène. Dans ce genre de situations, l'outil de prise de connaissance peut analyser la scène et proposer une (ou plusieurs)

trajectoires pour la caméra. L'analyse de la scène permet ici de disposer d'informations beaucoup plus précises sur la scène, permettant à la fois d'obtenir une meilleure prise de connaissance et une exploration en temps réel. Le temps d'analyse et de traitement de la scène n'a pas beaucoup d'importance, l'objectif étant une exploration de qualité et, autant que possible, en temps réel.

1.3 Conclusion

Ce mémoire se place dans cette problématique. Nous avons décidé de décomposer la manière de procéder en deux grandes étapes. La première partie est le calcul d'un ensemble minimal de points de vue importants et il est nécessaire qu'ils permettent de voir le maximum d'objets de la scène.

La seconde étape est la création d'une trajectoire tenant compte de ces points de vue. Cette trajectoire se doit d'être optimisée, principalement en termes de distance et de distance sémantique, la notion que nous allons introduire au chapitre 6, et doit lever les ambiguïtés qui peuvent subsister sur la scène.

Voilà présentée la problématique du contrôle de caméra, avec les principaux objectifs que nous nous fixons et les difficultés sous-jacentes. Avant de présenter notre approche, nous détaillons dans le chapitre suivant, les travaux relatifs au domaine.

Chapter 2

State of the art

2.1 Introduction

Computation of good viewpoint is important in many fields: computational geometry, robot motion, graph drawing etc. In this chapter we describe definitions of good view that arose in the existing literature.

With advances in 3D model acquisition technologies, databases of 3D models are evolving to very large collections. Accordingly, the importance of automatically crafting best views that maximally elucidate the most important features of an object has also grown for high-quality representative first views, or sequence of views. A number of papers have addressed the problem of automatically selecting a viewpoint for looking at an object.

The key-role in the domain of automatic explorations belongs to estimating viewpoint quality procedure, so, we start with reviewing advances in this domain.

2.2 Viewpoint quality

The first works on visual scene understanding were published at the end of the years 1980. Kamada and Kawai [KK88] have proposed a fast method to compute a point of view, which minimizes the number of degenerated edges of a scene. Refer to figure 2.1 for an illustration. They consider a viewing direction to be good if parallel line segments lie in a projection as far away from each other as possible. Intuitively, the viewer should be as orthogonal to every face of the 3D object as possible. As this is hard to realize, they suggest to minimize the maximum angle deviation (over all the faces) between a normal to the face and the line of the viewer's sight. However, this method does not ensure that the user sees a large amount of details [BDP00b].

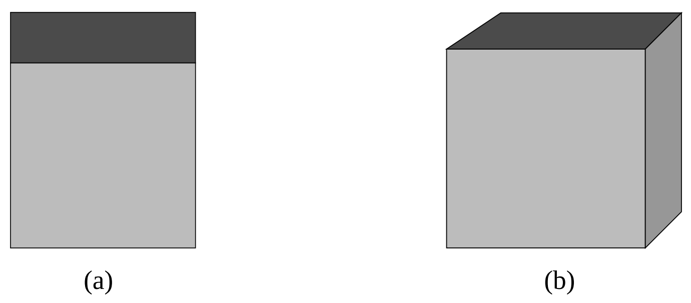


Figure 2.1: View **(b)** is better than **(a)**, because it does not contain degenerated faces.

Stoev and Straßer in [SS02] consider a different approach that is more suitable to viewing terrains, in which most surface normals in the scene are similar, and visible scene depth should be maximized.

Colin [Col88] has proposed a method initially developed for scenes modelled by octrees. The purpose of the method was to compute a good point of view for an octree. The method uses the principle of "direct approximate computation" to compute a good direction of view. This principle can be described as follows:

1. Choose the three best directions of view d_1, d_2 and d_3, among the 6 directions corresponding to the 3 coordinates axes passing through the center of the scene.

2. Compute a good direction in the pyramid defined by the 3 chosen directions, taking into account the importance of each one of the chosen directions (see Figure 2.2).

A direction of view is estimated better than another one if this direction of view allows to see more details than the other. Otherwords, the method chooses a viewpoint which shows the highest amount of voxels.

In [PB96] Plemenos and Benayada have proposed a heuristic that extends the definition given by Kamada and Kawai. The heuristic considers a viewpoint to be

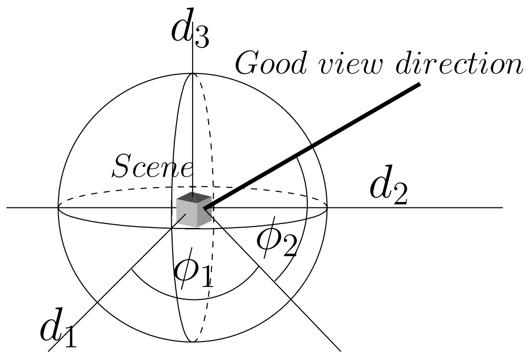


Figure 2.2: Direct approximate computation of a good direction of view.

good if it gives a high amount of details in addition to the minimization of the deviation. They show that if only the minimization is considered, then resulting views could hide important information of a scene. Therefore, they add another parameter to the maximizing function which counts observed details. The parameter added is the number of visible faces from a viewpoint. According to [PB96], the viewpoint quality can be computed by the following formula:

$$C(p) = \frac{\sum_{i=1}^{n} \left\lceil \frac{P_i(p)}{P_i(p)+1} \right\rceil}{n} + \frac{\sum_{i=1}^{n} P_i(p)}{r}, \tag{2.1}$$

where:

1. $C(p)$ is the viewpoint quality for the given viewpoint p,
2. $P_i(p)$ is the number of pixels corresponding to the polygon number i in the image obtained from the viewpoint p,
3. r is the total number of pixels in the image (resolution of the image),
4. n is the total number of polygons in the scene.
5. $\lceil a \rceil$ means the ceiling function, i.e the smallest integer number $a_c \in \mathbb{N} : a_c \geq a$.

In the context of computer vision, Weinshall and Werman [WW97] define two measures on views: view likelihood and view stability. View likelihood measures the probability that a certain view of a given 3D object is observed; it may be used to identify typical, or “characteristic” views. View stability measures how little the image changes as the viewpoint is slightly perturbed; it may be used to identify "generic" views. They show an equivalence between the most stable and most likely view of an object, and show that this is the view in which an object is flattest.

Sbert et al. [SFR+02, Váz03, FdABS99, Fei02, VFSH01] introduced an information theory-based approach to estimate the quality of a viewpoint. To select a good viewpoint they propose to maximize the following function, so called "viewpoint entropy":

$$I(p) = \sum_{i=0}^{N_f} \frac{A_i}{A_t} \cdot \log_2 \frac{A_t}{A_i}, \tag{2.2}$$

where:

1. p is the viewpoint,
2. N_f is the number of faces of the scene,
3. A_i is the projected area of the face number i,
4. A_0 is the projected area of background in open scenes,
5. A_t is the total area of the projection.

The function is the Shannon's entropy, where projected areas of faces are taken as a discrete random variable. Thus, the maximum entropy is obtained when a certain point can see all the faces with the same relative projected area A_i/A_t. So, if the background is not visible from the viewpoint, one has a maximum entropy of $\log_2 N_f$. By optimizing the value of entropy in images, Sbert et al. try to capture the maximum number of faces under the best possible orientation.

In [SPFG05] Sbert et al. have proposed an algorithm, based on the Kullback-Leibler distance. Its objective is to find the minimum representative set of views for a given object or scene. The *relative entropy* or Kullback-Leibler (KL) distance between two probability distributions $p = \{p_i\}$ and $q = \{q_i\}$ over the set $\mathcal{X}$ is defined by

$$KL(p\|q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i},$$

where, from continuity, the authors use the convention that $0 \log 0 = 0$, $p_i \log \frac{p_i}{0} = \infty$ if $a > 0$ and $0 \log \frac{0}{0} = 0$. The KL distance satisfies the *information inequality* $KL(p\|q) \geq 0$, with equality only if $p = q$.

To define the measure of viewpoint quality, the authors use the KL distance, where probability distribution p is given by the relative area of the projected faces over the sphere S of directions centered in the viewpoint V and the probability distribution q is given by the relative area of faces. Thus, viewpoint quality is defined by

$$KL(V) = \sum_{i=1}^{N_f} \frac{a_i}{a_t} \log \frac{\frac{a_i}{a_t}}{\frac{A_i}{A_T}},$$

where a_i is the projected area of face i, $a_t = \sum_{i=1}^{N_f} a_i$, A_i is the actual area of face i and $A_T = \sum_{i=1}^{N_f} A_i$ is the total area of the scene or object. The viewpoint quality measure can be interpreted as the distance between the normalized distribution of projected areas and the ideal projection, given by the normalized distribution of the actual areas. Thus, to select views of high quality means to minimize $KL(V)$.

Recently Chang Ha Lee et al. [LVJ05] have introduced the idea of mesh saliency as a measure of regional importance for graphics meshes. They define mesh saliency in a scale-dependent manner using a center-surround operator on Gaussian-weighted mean curvatures. The human-perception-inspired importance measure computed by the mesh saliency operator gives more pleasing results in comparison with purely geometric measures of shape, such as curvature.

Blanz et al. [BTB99] have conducted user studies to determine the factors that influence the preferred views for 3D objects. They conclude that selection of a preferred view is a result of complex interactions between task, object geometry, and object familiarity. Their studies support visibility (and occlusion) of salient features of an object as one of the factors influencing the selection of a preferred view. Gooch et al. [GRMS01] have built a system that uses art-inspired principles and some of the factors suggested by Blanz et al. [BTB99] to automatically compute initial viewpoints for 3D objects. Such systems can greatly benefit from purely geometrical computational models.

It is interesting to have functions which evaluate viewpoint qualities, but useless without methods which select the best viewpoint for an arbitrary scene. Thus, Plemenos [Ple91, PB96] proposed an iterative method of automatic viewpoint calculation. The scene is placed at the center of a sphere, whose surface represents all the possible points of view. The sphere is divided into 8 spherical triangles (see figure 2.3) and the best one is chosen according to the view qualities of the vertices of a triangle. Then, the selected spherical triangle is recursively subdivided. The best vertex is chosen to be the best point of view at the end of the process (figure 2.4).

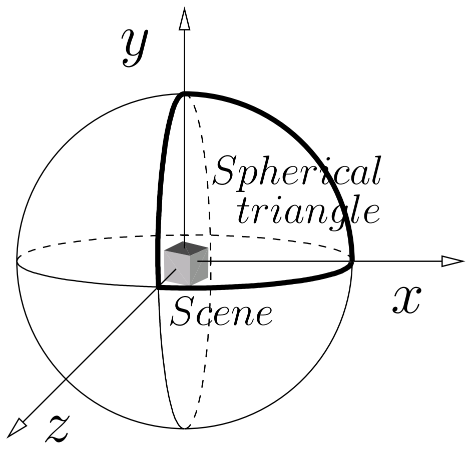


Figure 2.3: The sphere of points of view is divided into 8 spherical triangles.

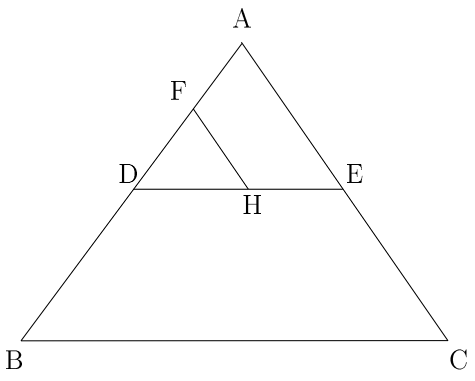


Figure 2.4: Recursive subdivision of the “best” spherical triangle.

Finding the optimal set of views of an object for purposes of image-based rendering has also been considered, using measures such as those providing best coverage of the scene [FCOL99], and those that provide the most information [VFSL02].

2.3 Dynamic exploration

Some commercial products such as Quick Time VR [Che95] offer panoramas, which are 360 degrees images taken from a single point of view. However, the user is restricted to camera movements of rotation and zooming, it is not possible to freely navigate and therefore the shown information depends on the initially taken photographs. Moreover, the user must jump from one viewpoint to another and therefore it is possible to be confused, to loose the notion of structure of the scene after some jumps.

2.3.1 Global exploration

A single good point of view is generally not enough for complex scenes, and even a list of good viewpoints does not allow the user to understand a scene, as frequent changes of viewpoint may confuse him (her). To avoid this problem, virtual world exploration methods were proposed.

Plemenos et al. and Dorme [BDP99, BDP00b, Dor01, BDP00a] proposed a method, where a virtual camera moves in real time on the surface of a sphere surrounding the virtual world. The exploration is on-line, the scene is being examined in incremental manner during the observation. All the polygons of the virtual world are taken into account at each step of the exploration. The method is based on heuristics $w_c = \frac{v_c}{2} \cdot (1 + \frac{d_c}{p_c})$, where:

1. w_c is the weight of the current camera position,
2. v_c is the viewpoint complexity of the scene from the camera's current position,
3. p_c is the path traced by the camera from the starting point to the current position,
4. d_c is the distance from the starting point to the current position.

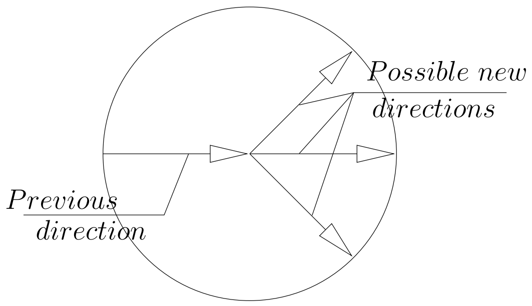


Figure 2.5: Only 3 directions are considered to ensure a smooth movement of the camera.

In order to avoid fast returns of the camera to the starting position, the importance of a viewpoint is made inversely proportional to the camera path from the starting to the current position. Also, for a smooth movement of the camera, only three new viewpoints are considered while computing the next position (see figure 2.5). A result of the technique application is shown at figure 2.6, which is taken from [PSF04].

Vázquez et al. in [VS03, Váz03] have proposed an exploration method that is very similar to the previous one. Similarly to Barral et al. [BDP00b], subsequent moves are chosen between three possible new directions, according to the

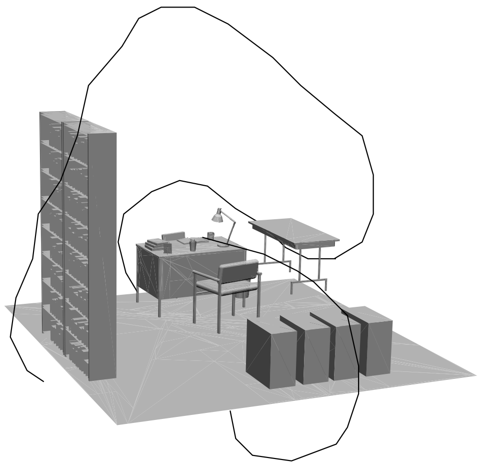


Figure 2.6: The virtual office exploration trajectory is taken from [PSF04] article.

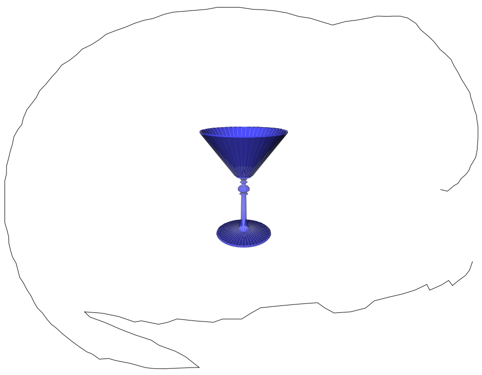


Figure 2.7: Exploration path around a martini glass.

last movement, to ensure a smooth displacement of the camera. The difference is that the next viewpoint is chosen in dependence of the entropy (equation (2.2)) and the number of faces not yet visited. To evaluate the qualities of the next three possible positions, they multiply the viewpoint entropy of each point by the number of new faces that appear with respect to a set of faces already visited. In the case, where none of the three possible viewpoints show a new face, they choose the one lying furthest from the initial position.

Figures 2.7 and 2.8 give exploration trajectories for a martini glass, a candlestick and a mug from the SGI Open Inventor toolkit.

In many cases, the on-line exploration is not necessary because the user has enough time to wait for pre-computing of interesting viewpoints for a virtual world and even interesting trajectories. Thus, Jaubert [Jau04, JTP06] proposed an off-line exploration method based on the pre-computing of a minimal set of good viewpoints. The computed viewpoints are sorted in importance order and stored together with the virtual world. The stored order is used each time an exploration of the virtual world is needed.

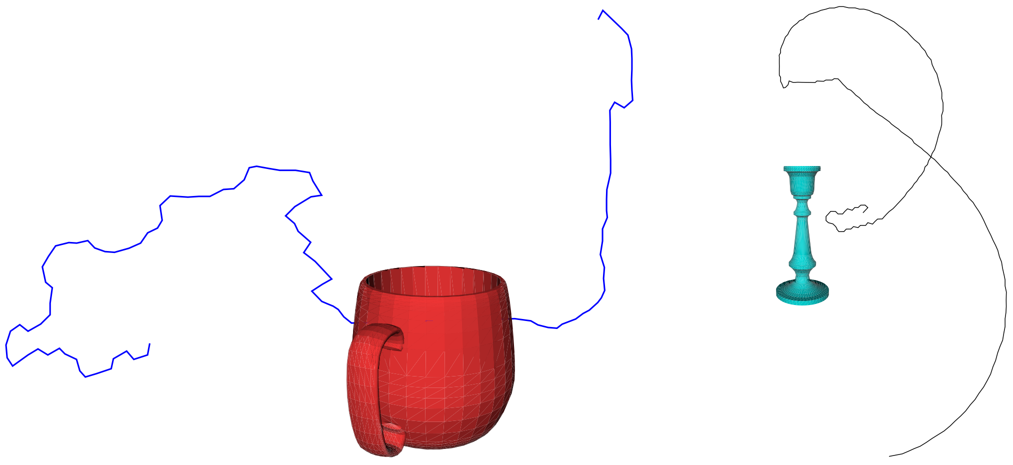


Figure 2.8: Exploration path around a mug **(a)** and a candlestick **(b)**.

2.3.2 Local exploration

Often virtual worlds are designed à priori to be visited inside. Examples are very numerous: worlds, populated by avatars, such as video games; simulations, such as virtual museums etc.

In this kind of exploration one can point at three main problems:

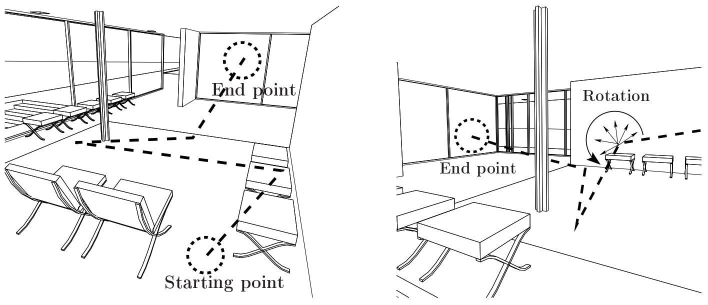
1. Obstacle avoidance, the camera should not pass through objects,
2. The camera has to explore important parts of the scene,
3. The camera has to show as good views as possible.

Obstacles may be static (walls, objects — in general, inaccessible places) or dynamic (obstacles may change their position or their shape in the time, other avatars, for example).

Yunfang et al [YZW03] have presented an algorithm based on synthetic vision, which is used to dynamically determine the walking path of the avatar in the Intelligent Virtual Environment. They use synthetic vision and scene octree to simulate the avatar's vision sense of the environment and memory of the scene, respectively.

Vázquez and Sbert [VS03] propose an automatic method of indoor scene exploration with limited degrees of freedom (in order to simulate a human walk-through).

The method is based on the viewpoint entropy. Exploring process stops at the moment when the camera is unable to discover new information. Figure 2.9 shows an example of indoor scene exploration.



(a) Room exploration path (b) Rotation of the camera near the obstacle

Figure 2.9: Images **(a)** and **(b)** show two close-ups of some parts of the path. Here some of the camera operations are marked with white arrows. Note, how in image **(b)** the camera stops when it is about to collide with room column and rotates until it finds a new way.

In [MC00, VFSH02, PGJT05] image-based techniques are used to control the camera motions in a changing virtual world. The problem faced in these papers is the adaptation of the camera behavior to the changes of the world.

For more details, a state of the art paper on virtual world global exploration techniques is available [Ple03], whereas viewpoint quality criteria and estimation techniques are presented in [PSF04].

2.4 Conclusion

In this chapter we have briefly reviewed existing techniques of automatic scene exploration, such as good viewpoint selection and creating elucidative movies. Unfortunately, there are very few papers which face this problem from the computer graphics point of view, whereas several papers have been published on the robotics artificial vision problem.

As we have mentioned above, the purpose of a virtual world exploration in computer graphics is completely different from the objectives of techniques used in robotics. In computer graphics, the purpose of the program which guides a virtual camera is to allow a human being, the user, to understand a new world by using an automatically computed path, depending on the nature of the world. The main interaction is between the camera and the user, a virtual and a human agent and not between two virtual agents or a virtual agent and his environment.

Existing methods have a lot of drawbacks, which will be discussed in the following chapters. Therefore, the methods could be mightily improved.

In the following chapters we are going to investigate how the scene can be interpreted as a source of information and how the camera can be considered as the receiver of this information.

Chapter 3

Viewpoint quality

3.1 Introduction

Although realistic rendering algorithms have been studied for a long time, most rendering systems place the camera at an arbitrary viewpoint, without taking into account which information is displayed to the user at this position. Unfortunately, developers often forget to supply expository information on how to deal with created worlds. Commonly, the user has to place the camera in a good position to better inspect a certain scene. The selection of good views in Computer Graphics has longly remained almost forgotten by researchers.

There is a strong necessity to have fast and accurate techniques for better exploration and clear understanding of various virtual worlds. The keyrole in the domain of automatic explorations belongs to estimating viewpoint quality procedure.

Quality of a view direction is a rather intuitive term and, due to its inaccuracy, it is not easy to precise for a selected scene, which viewpoints are "good" and which are not. Over the last decades, many methods were proposed to evaluate qualities of view directions for a given scene and to choose the best one. All of them are based on the fact that the best viewpoint gives to the user maximum amount of information about a scene. And again, an imprecise term "information" is met.

The viewpoint evaluation methods could be divided into three groups, characterized by the nature of input information:

1. *Low-level methods*, which consider only quantity of visible surfaces;

2. *Middle-level methods*, which take into account geometry of visible surfaces;

3. *High-level methods*, operating with the visibility of scene *objects*.

Of course, the proposed classification may be discussed, for example, the middle or high-level methods are the methods where a new level of abstraction was added. One can perform this with (almost) all methods from the low-level group, but it is reasonable to classify the works that *are* already done, not ones that *could* be done.

So, the first group of strategies is the most numerous one. All previously proposed methods belong to this group. Directly or implicitly, all the methods from the low-level group use only two global parameters as input:

1. Quantity of visible surfaces (projected area, amount of voxels, angle between direction of sight and normal to a face),

2. Number of visible faces.

In other words, all of them consider a viewpoint quality as a sum of qualities of separate faces, but don't take into account *how* a polygon is connected to the adjacent ones.

The number of visible faces is a quite weak criterion for a viewpoint quality estimation. For example, if we consider a very simple scene that consists of one square (figure 3.1(a)), then the viewpoint entropy equation gives us $I(p) = 0$ for a viewpoint p lying on the perpendicular to the square's center. If we subdivide the square (figure 3.1(b)), topology of the scene does not change, but $I(p)$ grows.

Thus, the methods using a number of faces to evaluate a viewpoint quality, depend on initial scene subdivision. Using the projected area of a face as a criterion of quality, the dependence appears also if we don't use an additive metric, i.e., the sum of areas. In [SP05] and [SPT06] we have introduced two new groups of methods — the middle-level and the high-level group, respectively.

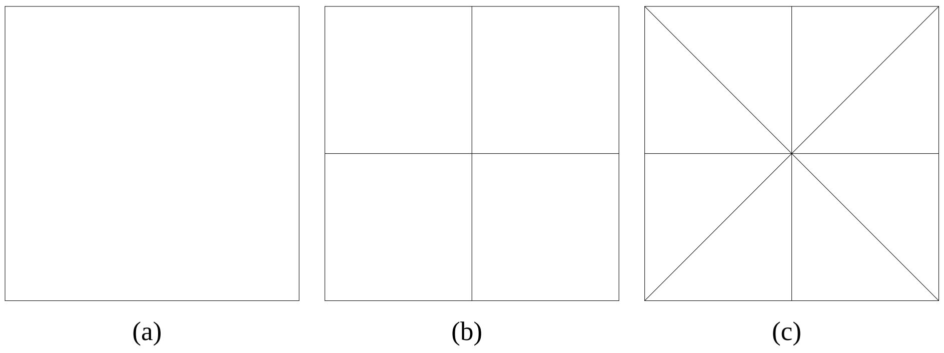


Figure 3.1: Three scenes represent the same square **(a)**, but subdivided into 4 parts **(b)** and 8 parts **(c)**. The viewpoint entropy equation gives us $I(p) = 0$ for **(a)**, $I(p) = \log 4$ for **(b)** and $I(p) = \log 8$ for **(c)**.

3.2 Middle-level methods [†]

3.2.1 Definition

As we have said before, it would be good if the estimation routine could be broadened by the knowledge of how a face is connected to the adjacent ones. We propose to use the curvature of a surface as such a knowledge. Intuitively, this choice is good since more mesh bends are visible from the viewpoint, more details of a scene could be seen. Unfortunately, we can't apply differential geometry definitions due to a discrete nature of our scene, so, the equations should be redefined. There are many ways to define a total curvature for a discrete mesh — for example, a curvature along edges, so-called extrinsic curvature (equation (3.1)) and intrinsic curvature for a set of vertices (equation (3.2)). Excellent surveys on curvature approximation methods for discrete meshes are available in papers by Calladine [Cal86] and Hamann [Ham93]. Let us suppose that a concave mesh brings to a user the same amount of information as a convex one, so, we propose to consider absolute values instead of signed ones. Then the first one can be written as:

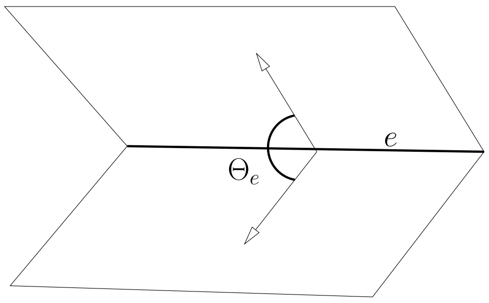
$$C_{ext} = \sum_{e \in E} (1 - \cos \Theta_e) \cdot |e|, \tag{3.1}$$

[†]The results are published in [SP05]

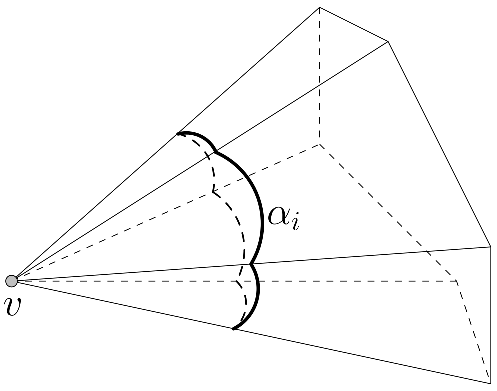
where E is the set of edges of the scene, $|e|$ is the edge length, Θ is the angle between normals to the faces sharing the edge e. The second one can be defined as follows:

$$C_{int} = \sum_{v \in V} \left| 2\pi - \sum_{\alpha_i \in \alpha(v)} \alpha_i \right|, \tag{3.2}$$

where $\alpha(v)$ is the set of angles adjacent to the vertex v (see figure 3.2).



(a) Extrinsic curvature is equal to the product of edge length and angle Θ between faces sharing the edge.



(b) Intrinsic curvature is equal to the sum of angles adjacent to the vertex minus 2π.

Figure 3.2: Two simple ways to define total curvature for a discrete mesh: curvature along edges and curvature in vertices.

Thus, our heuristic could be expressed as follows:

$$I(p) = C\left(F(p)\right) \cdot \sum_{f \in F(p)} P(f), \tag{3.3}$$

where:

1. $I(p)$ is the importance of the viewpoint p,
2. $F(p)$ is the set of visible faces from the viewpoint p,
3. $C(F(p))$ is the total curvature for the mesh visible from p, where either of the equations (3.2) and (3.1) may be used,
4. $P(f)$ is the projected area of the face f.

In order to calculate the curvature $C(F(p))$, both equations (3.2) and (3.1) may be used. They give almost equal results. Considerably different results may be obtained only in case of very unnatural, synthetic scenes.

The proposed heuristic is invariant to any subdivision of a scene keeping the same topology. Indeed, if we subdivide a flat face to several ones, then all inner edges and vertices will not be taken into account due to zero angles. The heuristic works for the outdoor and indoor explorations equally. An important property of such a kind of viewpoint quality definition is the possibility to extend it, using the total integral curvature $\int\limits_{\Omega} |K| dA$, into the class of continuous surfaces, such as NURBS etc., which nowadays become more and more usable.

The definition is very convenient in many cases, but it may be quite time consuming if it is necessary to estimate a lot of viewpoints. If we are not allowed to use point-to-region (viewpoint-to-face) visibility calculations, then, in order to reduce the complexity, the following heuristic may be used:

$$I(p) = C\left(F(p)\right). \tag{3.4}$$

If intrinsic curvature is used, this definition implies *point-to-point* (viewpoint-to-vertex) visibility problem. A case of extrinsic curvature usage implies *point-to-segment* (viewpoint-to-edge) visibility computations. Thus, extrinsic curvature is more expensive, but to reduce the complexity, we can suppose, that if two endpoints of an edge are visible, then all the edge is visible. Of course, it is not true, but if a (natural) scene consists of a big amount of polygons, this assumption does not deteriorate results, since few edges masked in the middle do not affect the main mass of edges. Even with this assumption extrinsic and intrinsic curvatures give almost equal results in majority of cases, only unnatural scenes are handled differently. Thus, the definition (3.4) greatly increases performance in many cases, an algorithm with analysis is given in section 4.2.

However, in distinction from equation (3.3), equation (3.4) do not differ distant objects from closer ones, figure 3.3 shows an illustration for this drawback.

As a cheap solution keeping an eye on projected sizes, we propose to use extrinsic curvature in formula (3.4) with projected edge lengths instead of absolute ones. This proposition explains why we have introduced not only intrinsic curvature, despite all its simplicity and efficiency.

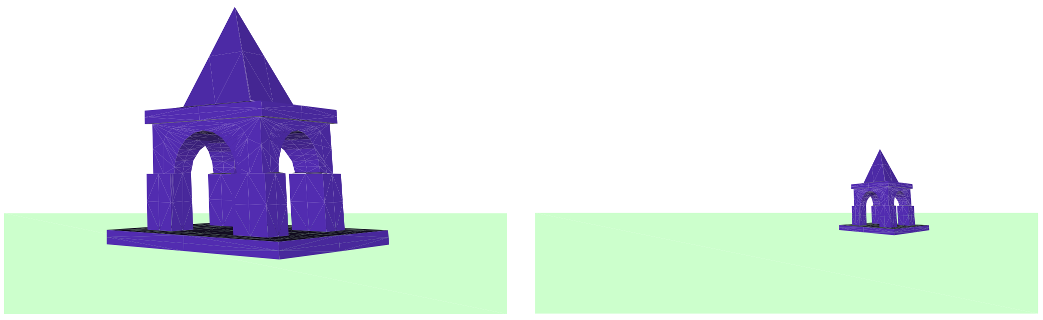

(a) Elucidative photo (b) Bad photo

Figure 3.3: The left photo is better than the right one. Heuristic, based on equation 3.3, selects photo **(a)** as the best one, while equation 3.4 gives equal values for the both photos.

3.2.2 Comparison with previous results

Projected area versus our heuristic

Figure 3.4 shows us that maximum of projected area may give to a user large amount of pixels drawn, but few details of a scene.

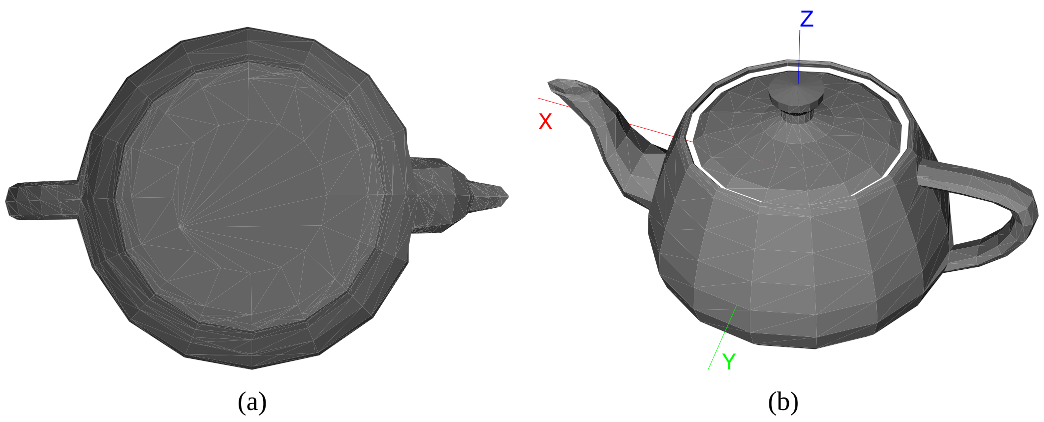


Figure 3.4: Maximum area **(a)** versus our method of viewpoint selection **(b)**.

Viewpoint entropy versus our heuristic

A candlestick from SGI Open Inventor models is drawn at figure 3.5. Figure 3.5(a) is taken from [Váz03], this viewpoint maximizes the viewpoint entropy. A view direction, maximizing our heuristic is shown at figure 3.5(b).

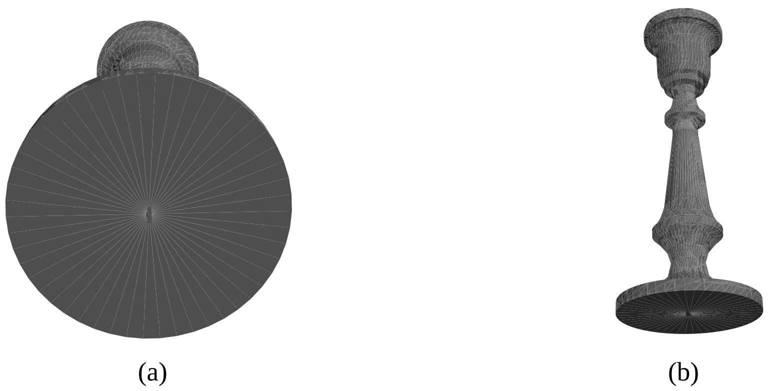


Figure 3.5: The best views of the candlestick are computed by entropy-based method **(a)** and our method **(b)**.

Viewpoint complexity versus our heuristic

A martini cup from SGI Open Inventor toolkit was tested. The viewpoint complexity [PB96] and the viewpoint entropy [Váz03] techniques give the same result, which is shown at figure 3.6(a). The result of our method application is given at figure 3.6(b).

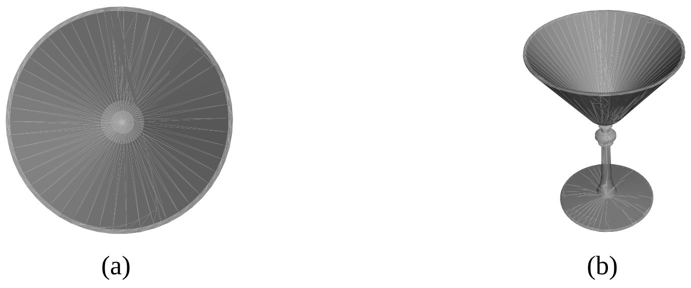


Figure 3.6: The viewpoint complexity and the entropy-based methods give the same result **(a)**; our method application **(b)**.

Stand-alone results

Figures 3.7 and 3.8 present results of viewpoint quality estimations for two models: the Utah teapot and a virtual office model. The figures show qualities of viewpoints lying on surrounding sphere. In order to better represent the 3D data, each figure shows a scene from different viewpoints. To distinguish values in black-and-white picture, we have connected viewpoints by lines; more the line is dark and thick, more the viewpoints are good.

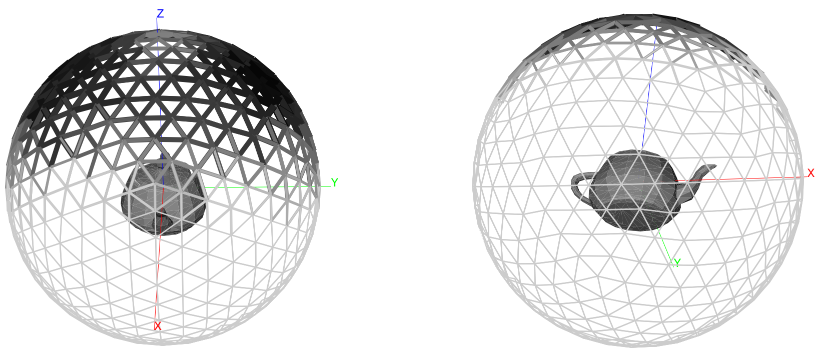
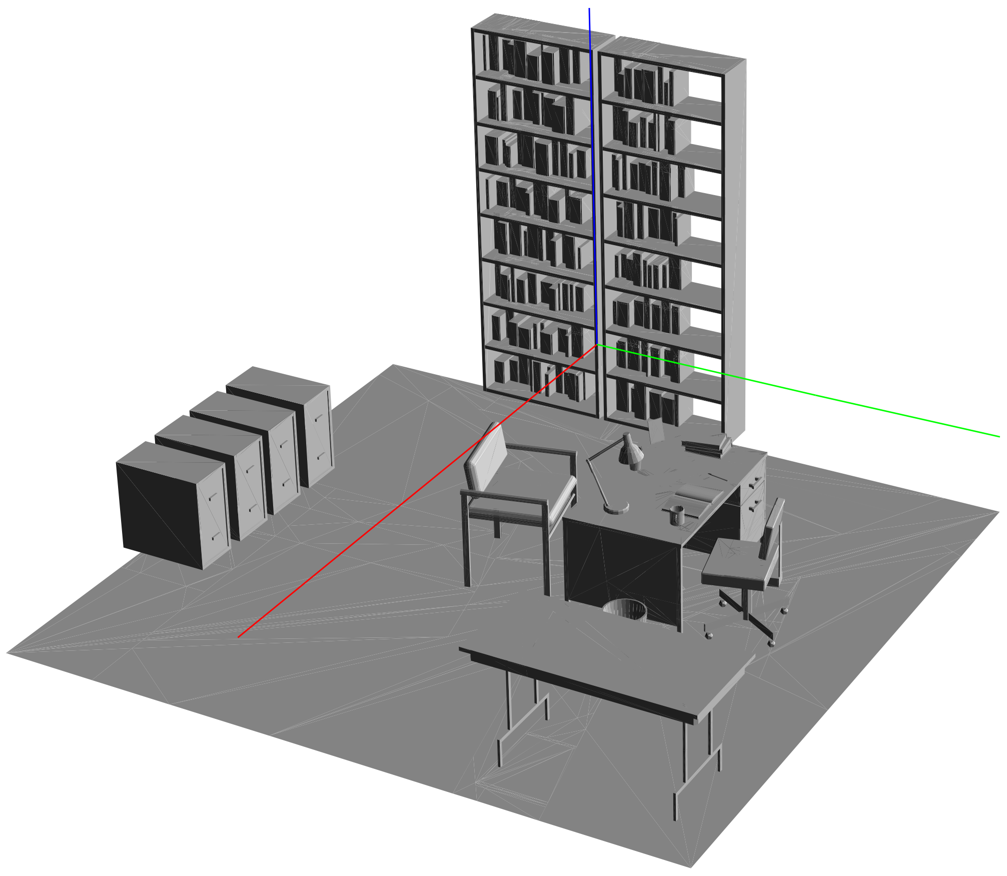


Figure 3.7: The figure shows the Utah teapot from two points of view. The best view is available in figure 3.4(b). Dark and thick lines indicate good viewpoints.



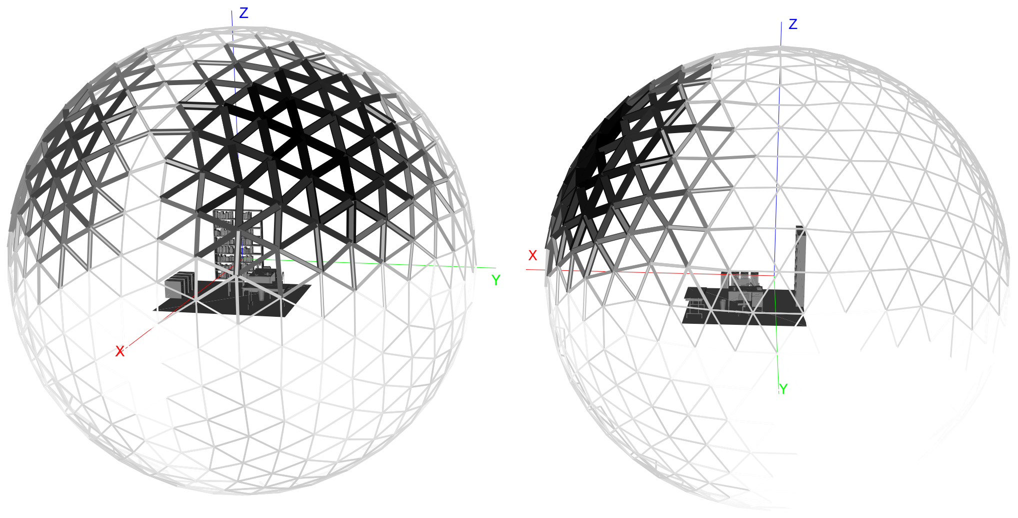


Figure 3.8: The figure shows the virtual office model from different points of view. First image gives the scene from the best viewpoint, the second and the third ones show the viewpoint qualities. Viewpoints are connected by lines; more the line is dark and thick, more the viewpoints are good.

3.3 High-level methods [†]

3.3.1 Definition

Although the above methods allow to obtain interesting results, they are low-level methods, as they use inadequate “vocabularies”. Let us consider an example.

The scanning process at 300 dpi resolution produces about 9 megapixels per A4 page. Nobody uses elementary pixel configuration to describe content and structure of this page. Instead, we apply Optical Character Recognition methods and represent content by characters and structural markup. Computer Graphics suffers from weak representation of 3D data. Development of proper “vocabularies” for a new generation of meta-data, capable to characterize content and structure of multimedia documents, is a key-feature for categorization, indexing, searching, dissemination and access.

It would be a grand challenge if a complete semantic 3D model could be used instead of projection in lower dimensions (image, text, animations) or structureless collection of polygons.

Now let us suppose that, having a complex scene, there exists some proper (in human perception) division of a scene into a set of objects. Figure 3.9 shows us an example of such a scene, where the subdivision into a set of objects is shown by different colors. These objects are: the computer case, the display, the mouse, the mouse pad, two cables, the keyboard and several groups of keys.

Only 20% of the display surface is visible, but it does not embarrass its recognition, because if we are familiar with the object, we could *predict* what does it look like. Thus, we could conclude that if there exists a proper subdivision of a scene into a set of objects, the visibility of the objects could bring more information than just the visibility of the faces, and this leads us to the third group of methods — *the high-level group*.

The requirement of a scene division into objects does not limit the area of the method application. There are many ways to get it. First of all, complex scenes often consist of non-adjacent simple primitives, and this leads to the first disjunction of a scene. Otherwise, if a scene (or some parts of a scene) is represented by a

[†]The results are published in [SPT06]

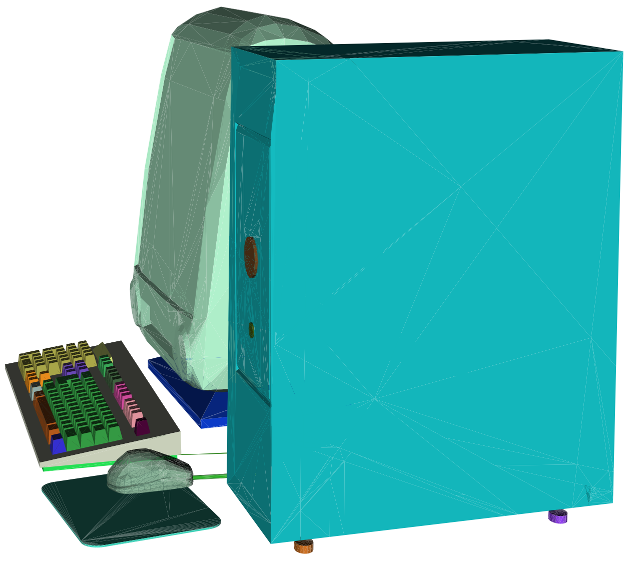


Figure 3.9: The scene is subdivided into a set of objects. The display is almost completely hidden by the case, but we could clearly recognize it.

huge mesh, it could be decomposed. The domain of surface decomposition is well-studied and there are a lot of methods giving excellent results. One can point at results of Zuckerberger et al. [ZTS02] and Chazelle et al. [CDST95].

The method also could be very useful in declarative modelling by hierarchical decomposition (refer to [Ple95] for the main principles). In such a case, the decomposition could be provided by a modeler directly, or, probably, it can be combined with the information extracted from a scene geometry.

An accurate definition of the new heuristic is given further. Let us suppose that for each object ω of a scene Ω importance $q(\omega) : \Omega \to \mathbb{R}^+$ is specified.

We would like to generalize the method and do not want to be limited by a strict definition of the importance function, because it could be done in different ways, especially, if some additional knowledge about a scene is supplied. For example, if the method is incorporated into some dedicated declarative modeler, the importance of an object could be assigned in dependence on its functionality. Moreover, after the first exploration the importances could be rearranged in a different manner to see various parts of a scene more precisely than during the previous exploration.

If no additional information is provided and the user takes into account scene geometry only, then the size of object bounding box could be considered as the importance function:

$$q(\omega) = \max_{u,v\in V_\omega} |u_x - v_x| + \max_{u,v\in V_\omega} |u_y - v_y| + \max_{u,v\in V_\omega} |u_z - v_z|,$$

where V_ω is the set of vertices of the object ω.

It is also necessary to introduce a parameter characterizing the *predictability* (or *familiarity*) of an object: $\rho_\omega : \Omega \to \mathbb{R}^+$. In other words, the parameter determines the quantity of object surface to be observed in order to well understand what the object looks like. If an object is well-predictable, then the user can recognize it even if he sees a small part of it. bad predictability forces the user to observe attentively all the surface.

The ρ_ω parameter sense could be also interpreted in a different manner. Even if an object is well-predictable (for example, it is a famous painting), the parameter ρ_ω could be chosen as for an object with bad predictability. This choice forces the camera to observe all the painting.

We propose to consider the function $f(t) = \frac{\rho_\omega+1}{\rho_\omega+t}t$ as the measure of observation quality for an object, where $0 \le t \le 1$ is the explored fraction of the object (for example, the area of the observed surface divided by the total area of the object surface). Refer to figure 3.10 for an illustration. If the percentage t for the object ω is equal to zero (the user has not seen the object at all), then $f(t)$ is zero (the user cannot recognize the object). If all the surface of the object ω is observed, then $f(t)$ is 1, the observation is complete.

If nothing is known about a scene except its geometrical representation, then in order to observe it, the parameter ρ could be taken as rather small value, for example, $\rho_\omega \equiv 0.1 \forall \omega \in \Omega$. In such a case even exploration of a part of an object gives a good comprehension.

Now let us suppose that there exists some additional knowledge, for example, a virtual museum is considered. Then for all the paintings the parameter could be taken equal to 1000 and, for all the walls, chairs, doors equal to 0.1. Now, in order

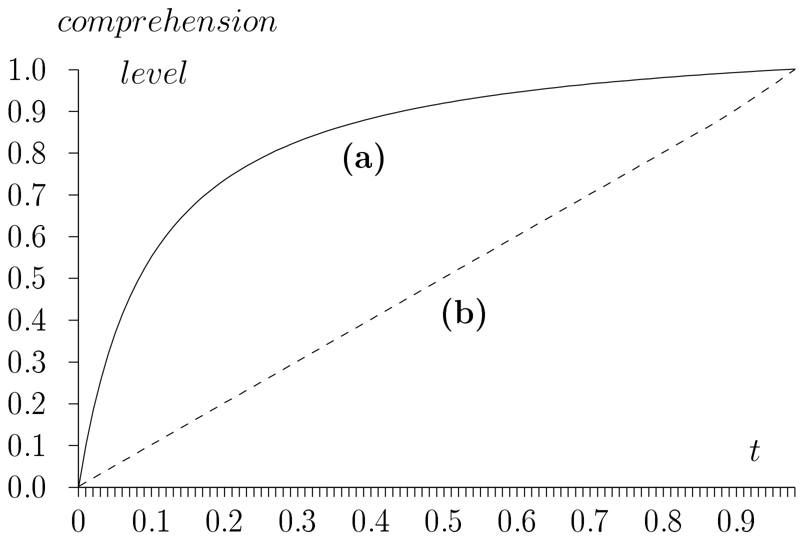


Figure 3.10: The behavior of the function $f(t) = \frac{\rho+1}{\rho+t} \cdot t$ for two values of the parameter ρ. **(a)** $\rho = 0.1$, even a part of an object provides a good knowledge. **(b)** $\rho = 1000$, the user should see all the object to get a good knowledge.

to get a good comprehension of a painting, one should observe all its surface, but only a small part of door or wall is necessary to recognize them.

Let us consider a viewpoint p. For scene objects this point gives a set of values $\Theta(p) = \{0 \leq \theta_{p,\omega} \leq 1, \omega \in \Omega\}$, where $\theta_{p,\omega}$ is the fraction of visible area for the object ω from the viewpoint p. $\theta_{p,\omega} = 0$ if the object is not visible and $\theta_{p,\omega} = 1$ if one can see all its surface from the viewpoint p.

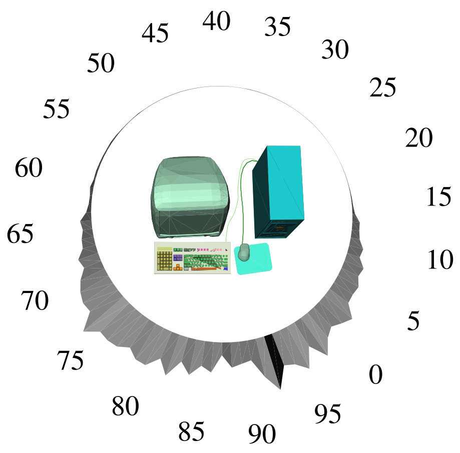
The fraction $\theta_{p,\omega}$ may be computed in various ways. The simplest one is to divide the area of the visible surface by the total area of an object. A bit more complicated way is inherited from the middle-level method (see equation (3.4)). If we divide the curvature of the visible surface by the total curvature of an object, we obtain the fraction equal to 0 if an object is not visible at all and equal to 1 if we could see all its surface.

Thus, we propose to evaluate the viewpoint quality as the sum of scene object importances with respect to their visibility:

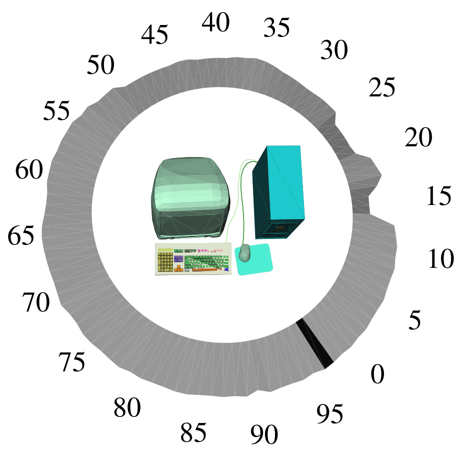
$$I(p) = \sum_{\omega\in\Omega} q(\omega) \cdot \frac{\rho_\omega + 1}{\rho_\omega + \theta_{p,\omega}} \theta_{p,\omega}. \tag{3.5}$$

3.3.2 An example of applying the technique

In this section, the computer model, that we have met before, is considered more precisely. Figures 3.11 and 3.12 show the results of applying two techniques from the middle and the high level classes. For the new method no additional information is provided, so, the bounding box sizes are taken as the importance function $q(\omega)$ and $\rho_\omega \equiv 0.1 \forall \omega \in \Omega$.

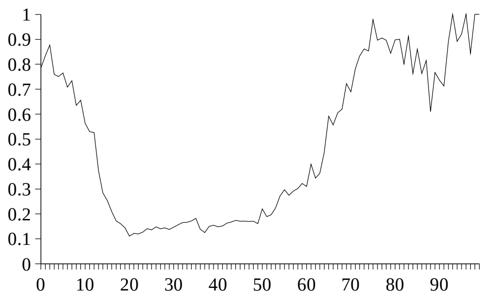


(a) The middle-level method.

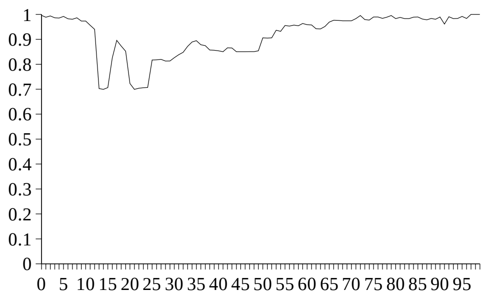


(b) The new high-level method.

Figure 3.11: The qualities for 100 viewpoints equally distanced from the center of the model. The best viewpoint is shown by the black sector.



(a) The middle-level method.



(b) The new high-level method.

Figure 3.12: The plain graphs of the qualities for 100 viewpoints around the scene (see figure 3.11).

The best viewpoints, chosen with the two methods, are quite close (the picture is shown at figure 3.13), but there are significant differences in the estimation of other view directions.

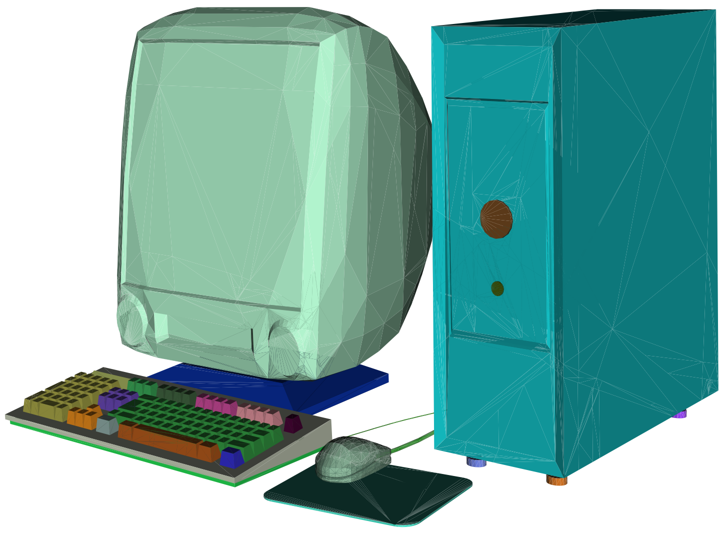


Figure 3.13: The best viewpoint for the computer model.

Compare figure 3.9, showing the scene from the viewpoint number 10, and figure 3.13, showing it from the best viewpoint. It is clear that the viewpoint 10 is less attractive, but it still gives a good representation of the scene. The function on figure 3.12(b) decreases smoothly in this region, while figure 3.12(a) shows a drastic fall. At the viewpoint 17 the function from figure 3.12(b) grows, because a back side of the display and a part of the keyboard are visible simultaneously. Then it decreases again because the case covers the keyboard. The new method also shows a better quality than the old one from the back side of the scene. From each viewpoint some parts of the mouse or of the keyboard are visible, so the estimation should not be so small as at figure 3.11(a).

3.4 Conclusion

At the beginning of this chapter we have presented new criteria of evaluating a viewpoint quality, which do not depend on changes in a scene keeping the original topology. The criteria are extendable into the class of continuous surfaces such as NURBS etc., more and more usable nowadays. Belonging to the middle-level heuristics, the proposed criteria allow to get a good comprehension of a single virtual artefact or a general comprehension of a scene.

A new high-level heuristic is proposed at the end of the chapter. The criterion uses a subdivision of a scene into a set of objects. To estimate contribution of each object to total viewpoint quality, the high-level method uses definitions of middle-level criteria. This technique could be particularly helpful in virtual heritage projects. For example, in a virtual museum different objects should have different importances. Obviously, the artefacts should have significantly greater importances than walls, chairs and so on. Having a proper division of a virtual museum model into a set of objects, we obtain good heuristics for an automatic (or guided manual) exploration.

Chapter 4

External exploration [†]

4.1 Introduction

Exploring virtual worlds becomes a more and more important research area in Computer Graphics these last years, mainly due to continuous development of Internet services. Efficient algorithms for automatic exploration of virtual worlds may help the user in many ways. For example, by allowing him (her) to understand a scene found on the web or to make a guided visit to a virtual museum. In all cases, an automatic exploration could be proposed to the user, taking into account exploration criteria such us view quality, smoothness of the camera path and so on.

The purpose of a virtual world exploration in computer graphics is completely different from the objectives of techniques used in robotics. In computer graphics, the purpose of the program which guides a virtual camera is to allow a human being, the user, to understand a new world by using an automatically computed path, depending on the nature of the world. The main interaction is between the camera and the user, a virtual and a human agent and not between two virtual agents or a virtual agent and his environment.

Two main classes of automatic virtual world exploration techniques may be distinguished:

- Global exploration techniques, where the camera remains outside the world to be explored. These techniques allow a global view of a virtual world but

[†]The results are published in [SPT06]

may fall in exploration of some never visible details.

- Local exploration techniques, where the camera moves inside the explored world. These techniques allow to reach all possibly visible details of the world but they don't give a global view of it.

Local exploration may be useful, and even necessary in some cases, but only global exploration could give the user a general knowledge on a scene. This chapter presents our results in developing global exploration techniques.

4.2 Analytic visibility graph

Let us suppose that there is an unknown scene, and the user would like to get a general comprehension of its structure. Since the user would like to explore the exterior of the scene, it is reasonable to restrict the space of possible viewpoints to a surrounding sphere. Moreover, viewpoint quality is a quite smooth function, so the sphere could be easily discretized. Thus, the scene is placed in the center of the sphere, whose discrete surface represents all the possible points of view.

It would be very convenient for many quality estimation routines if the undirected bipartite graph $G = (S \bigcup F, E)$ can be obtained, where the first part S corresponds to the set of viewpoints of the discrete sphere and F corresponds to the set of faces of the scene. The set of arcs E represents visibility between objects from S and F, i.e., G is the analytic visibility graph.

Unfortunately, computing such a graph is a quite expensive task, because the proposed methods of viewpoint quality estimation operate with quite expensive point-to-region visibility and approximate calculation of visible parts using a Z-Buffer. Figure 4.1 illustrates the technique. The scene is rendered from a viewpoint, coloring each face with a unique color ID and using flat shading. In the resulting rendered scene, each pixel represents the color code of the face visible in this pixel. OpenGL allows to obtain a histogram which gives an information on the number of displayed colors and the ratio of the image occupied by each color. See [BDP99] and [NRTT95] for more detailed description.

A quality estimation routine for a single viewpoint, applying these methods, runs in $O(n_f \cdot Z)$ time, where $n_f = |F|$ is the number of faces in the scene and

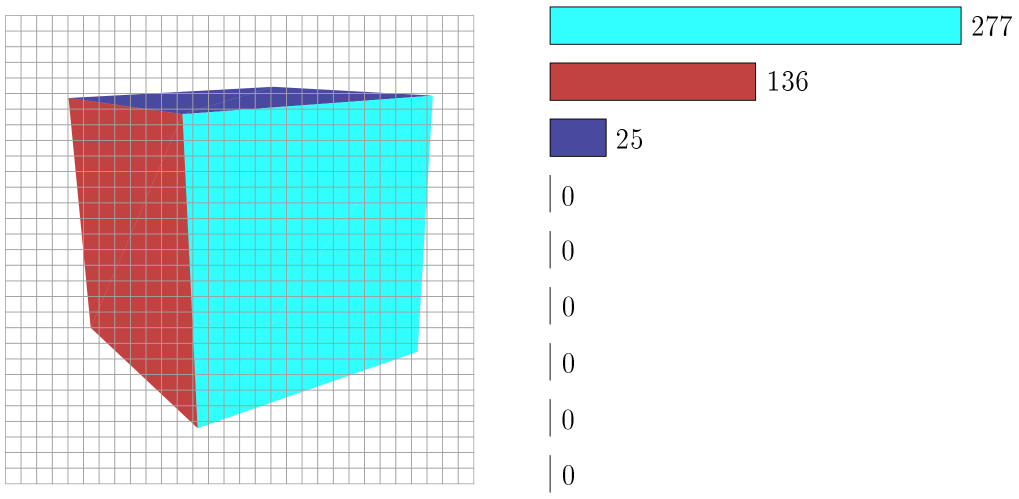


Figure 4.1: Detecting visible faces by rendered image analysis. Left image represents a rendered cube, right image shows the histogram which tells us how many pixels are occupied by each face of the cube.

Z is the resolution of the Z-Buffer. For $n_s = |S|$ viewpoints the running time is $O(n_f \cdot n_s \cdot Z)$. Note that Z should be significantly greater than n_f in order to have at least few pixels to display each face. The complexity of the algorithm forces the authors to use adaptive search algorithms, which may give inexact results, even for a single good viewpoint selection.

A very important advantage of the heuristic (3.4) is the replacing point-to-region (viewpoint-to-face) with point-to-point (viewpoint-to-vertex) visibility computations. Likewise, when the intrinsic curvature is used in order to compute visible fractions, viewpoint quality equation (3.5) requires point-to-point visibility computations only.

Since the visibility of faces does not play the main role anymore, the visibility graph is transformed to $G = (S \bigcup V, E)$, where V corresponds to the set of vertices of the scene. The set of arcs E represents visibility between objects from S and V. The graph could be computed in $O\left(n_f \cdot \sqrt{n_s} \cdot n_v\right)$ operations, where $n_v = |V|$ is the number of vertices of the scene.

Let us suppose that the surrounding sphere is pixelized; each element represents a viewpoint. To evaluate viewpoint qualities, it is necessary to find all vertices of the scene which are visible from each element of the sphere. In order to perform it rapidly, a reverse problem is considered, all visible viewpoints are to be found for each vertex of the scene. It allows to use the structuredness of the rasterized sphere

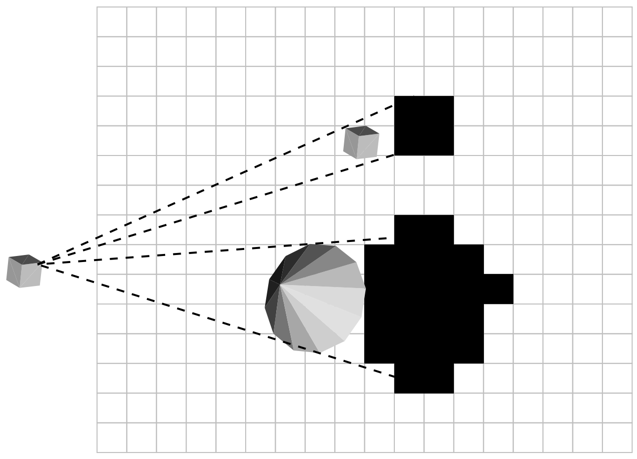


Figure 4.2: The scene consists of two cubes and a cone, the rasterized plane represents a set of viewpoints. The given vertex of the scene is not visible from viewpoints colored black.

for fast elimination of hidden areas. Without loss of generality a rasterized plane could be considered instead of the sphere and a triangulated scene (a polygonal mesh could be triangulated in linear time using the algorithm presented in [Cha91]).

The algorithm iterates through each vertex of the scene and finds all pixels of the plane which are visible from the given vertex (see figure 4.2). The main step of the algorithm is the determination of a set of viewpoints, which are visible from a given vertex. The naive way is to project all the triangles on the plane, to fill up the projections and then to eliminate coloured areas. This way is the simplest one, but it is expensive. In such a case, the main step of the algorithm runs in $O(n_f \cdot n_s)$ time in the worst case, because it could colour pixels several times.

We propose to keep a matrix of numbers, where each element of the matrix corresponds to a pixel of the plane. Initially the matrix is to be filled up with zeroes. At the main step of the algorithm a projection boundary is to be found for each triangle of the scene. The boundary could be obtained using the Bresenham's algorithm [Bre65] of digital line drawing. Then the boundary could be divided into two parts: a left part and a right part (see Figure 4.3). The matrix elements which correspond to the elements of the left part of the boundary are to be increased, the right ones are to be decreased. Figure 4.4 gives the detailed illustration.

When all the projections are drawn, the inner parts of the projections are to be eliminated for each row of the matrix. This task is similar to the brackets sequence

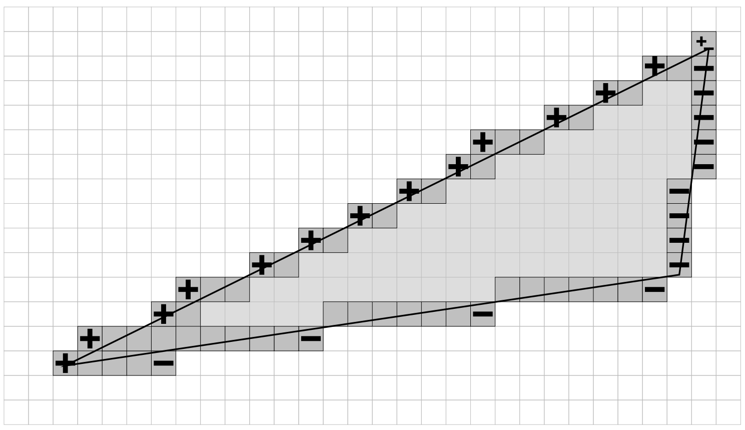


Figure 4.3: The triangle is drawn on the rasterized plane, the boundary is shown by the dark gray color, the inner part by the light gray. The symbols “+” indicate the left part of the boundary, the “−” shows the right one.

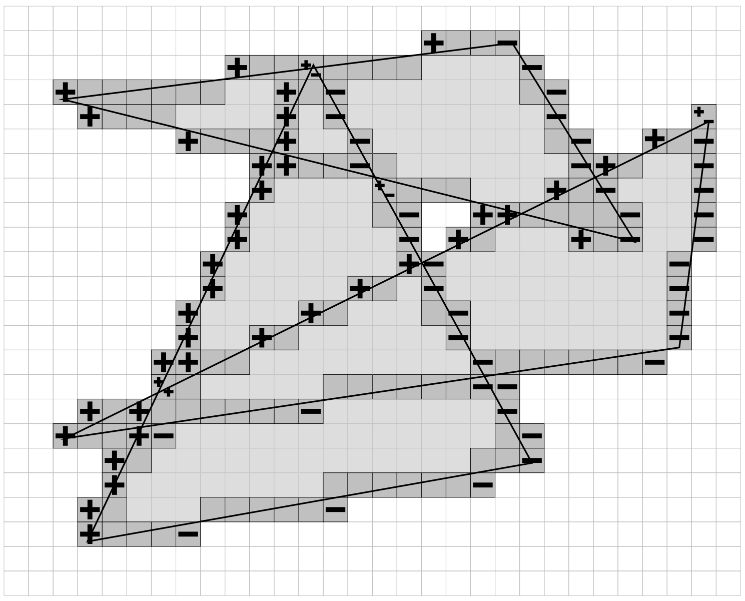


Figure 4.4: Three faces are drawn on the plane. Symbol “+” means increasing of the corresponding element of the matrix, symbol “−” means decreasing.

task, where each row represents a string, each increasing of a matrix element means insertion of an opening bracket into the string and each decreasing means insertion of a closed bracket. Algorithm 1 shows the scheme.

```
Input: The set of faces $F$, the rasterized sphere $S$
Output: The analytic visibility graph $G = (S \bigcup V, E)$

$q_s \leftarrow 0 \; \forall s \in S$
for each vertex $v$ of the scene do
  for each triangle $f \in F$ do
    Find the projection $P_f$ of the triangle $f$ onto the plane
    Increase matrix elements corresponding to the left part of the projection
    Decrease matrix elements corresponding to the right part of the projection
  end for
  Determine the set $A$ of the inner parts according to the calculated matrix (per-
  form the brackets sequence task for each row of the matrix)
  $E \leftarrow E \bigcup (s, v) \; \forall s \in S \setminus A$
end for
```

Algorithm 1: The fast visibility computation method.

If the plane consists of n_s pixels, then the maximal boundary drawing time is $O(\sqrt{n_s})$ for a triangle. Having n_f triangles and n_v vertices in a scene, the total running time of the algorithm is $O(n_f \cdot \sqrt{n_s} \cdot n_v)$ operations. Thus, the running time is reduced from $O(n_f \cdot Z \cdot n_s)$ to $O\left(n_f \cdot n_v \cdot \sqrt{n_s}\right)$ operations, $Z \gg n_f$.

Let us show how this technique can be used for the rasterized sphere. Let us suppose that there is a tesselated sphere S; then a graph $G_t = (S, E_t)$ can be constructed where the set of nodes S corresponds to the tesselation parts. Edge $(s_1, s_2) \in E_t$ if and only if the part s_1 is adjacent to s_2. For example, each pixel of usual screen has 8 neighbours. If there is such a kind of graph and two points of the sphere are given, then the shortest path in the graph corresponds to geodesic curve (or shortest line) connecting the points. Since the graph has the very special structure, the shortest path of length l can be found in $O(l)$ operations using an adaptation of the Bresenham's algorithm. So, where we have projected 3 vertices of a triangle on the sphere, we can find 3 shortest paths representing the boundary of the projection. The paths are stored and the next triangle is projected. After the main loop, when all the boundaries are computed, it is easy to remove from the graph all the nodes met, at least once, in the set of paths. Then the graph is splitted into a set of linked components (for example, we can split it by constructing a depth-first search tree) and the components corresponding to internal parts of the triangles are to be removed.

4.3 Static visualizations

4.3.1 Set of images

There are many ways to visualize a scene in order to understand it. They could be separated into two classes of methods: static and dynamic. Static methods give the user a set of photos of a scene and dynamic ones show a “movie”. While this section describes static methods of exploration, the following one gives description for dynamic visualization techniques.

The interesting problem is to find a set of images, representing all the scene. Let us provide a strict formulation. There is a scene, i.e., a set of faces F and a set of vertices V are provided. Of course, a reduced version of the task could be considered as well as the complete exploration task. If it is necessary to find a set of images, completely representing an interesting *part* (parts) of a scene, then F and V represent sets of faces and vertices, respectively, of the part (parts). A surrounding discrete viewpoint sphere S is given. It is not hard to compute the undirected bipartite graph $G = (S \bigcup V, E)$ (refer to algorithm 1 for details). The task is to find a set of viewpoints representing all the scene:

$$M \subseteq S \,:\, V = \{v|(u, v) \in E, u \in M\}\,.$$

Unfortunately, the minimal set cannot be found in real time.

Lemma 4.3.1. *The minimization of $|M|$ is NP-complete.*

Proof. Let us remember the Feedback Vertex Set (FVS) problem: given an undirected graph, what is the smallest set of vertices whose deletion leaves an acyclic graph? Garey and Johnson [GJ79] have shown that the FVS $\in$ NP. Now we shall try to transform the FVS problem to the initial task to prove NP-completeness.

Let us consider the following transformation: V will have nodes corresponding to all cycles of the graph in the FVS. S corresponds to all the vertices of the graph,

$$E = \{(u, v)| \text{ the cycle corresponding to } v \text{ contains a vertex corresponding to } u\}.$$

Thus, the FVS can be converted to the initial problem, but there is a difficulty: the number of cycles in a graph can be exponential in the number of nodes of the graph. So, there is no direct polynomial algorithm, but, fortunately, we are not obliged to consider all the cycles in a graph.

Let us remember two definitions:

1. A *fundamental cycle* with respect to a spanning tree T of graph G consists of a non-tree edge (u, v) and a path in T between u and v.

2. A *fundamental set of cycles* with respect to a spanning tree T of graph G consists of all fundamental cycles with respect to T of G.

A fundamental set of cycles is a very important construction since it is a basis of all cycles of a graph: we can consider a cycle as a vector of m dimensions ($|E| = m$, where E is a set of arcs) with 0 or 1 as components, 1 if a cycle contains corresponding edge and 0 otherwise. Then $\oplus$ operation can be easily defined as addition modulo 2 (see [SR61]).

Therefore, every cycle in a graph can be obtained as a sum of some fundamental cycles with respect to a depth-first search tree, so, if we cut all cycles from a fundamental set, all cycles in a graph are cut automatically.

But the size of a basis is linear in number of edges. Moreover, the size of a basis is equal to cyclomatic number of graph, so, we can transform the FVS problem to our initial problem in polynomial time, that proves NP-completeness. $\square$

Thus, the minimization of $|M|$ is NP-complete. Moreover, Feige in [Fei98] has proved a very strong result showing that for every $\epsilon > 0$ there is no polynomial-time algorithm that can approximate the task within $(1 - \epsilon) \log |V|$ unless $NP \subseteq DTIME\left[n^{O(\log \log |V|)}\right]$.

However, there are some good news for us: such a simple heuristic as a greedy algorithm (Algorithm 2) has a good bound.

```
$G = (S \bigcup V, E)$; $M = \emptyset$
while $E \neq \emptyset$ do
    Select $s \in S$ with the maximum number of adjacent edges
    $M = M \bigcup \{s\}$
    Remove from $G$ $s$ with all adjacent vertices and edges
end while
```

Algorithm 2: The greedy algorithm approximating the set cover problem.

Lemma 4.3.2. *The greedy algorithm finds a solution with at most $c_{opt} \cdot \left(\log \frac{|V|}{c_{opt}} + 1\right)$ vertices, where c_{opt} is a number of vertices in optimal solution. So, it is an $O(\log |V|)$ approximation — or better if c_{opt} is large.*

Proof. Let us denote the initial number of vertices $|V|$ as n. Since the optimal solution uses c_{opt} vertices, there must be some set that covers at least a $\frac{1}{c_{opt}}$ fraction of the vertices. Therefore, after the first iteration of the algorithm, there are at most $n \cdot \left(1 - \frac{1}{c_{opt}}\right)$ vertices left. The optimal solution for the task at the second step could not be greater than c_{opt}, since the initial optimal solution satisfies the new task. After the second step, there are at most $n \cdot \left(1 - \frac{1}{c_{opt}}\right)^2$ vertices left, etc. After c_{opt} rounds, there are at most $n \cdot \left(1 - \frac{1}{c_{opt}}\right)^{c_{opt}} < n \cdot \frac{1}{e}$ vertex nodes left. After $c_{opt} \ln \frac{n}{c_{opt}}$ rounds there are at most c_{opt} points left. Thus, the number of iterations the algorithm needs is $It(n) = c_{opt} + It\left(\frac{n}{e}\right) = O(c_{opt} \cdot \ln n)$. □

This algorithm gives a good illustration of efficiency of the greedy search, however, to use it in the real life, the algorithm is to be improved. First of all, it misses a stopping condition. The algorithm produces a set of viewpoints which covers all given mesh. Often last iterations of the algorithm produce viewpoints which almost do not bring new information. In real life we seldom need all the covering set. Thus, a stopping condition, a threshold is to be defined.

The main step of the algorithm is the next point to be improved. At each step the algorithm selects a viewpoint with the maximum number of adjacent edges in the visibility graph. In fact, it means that the algorithm considers a viewpoint to be good, if it gives a high number of visible faces. However, in the previous chapter we have introduced the group of high-level methods of viewpoint quality estimation. Thus, it is reasonable to use equation (3.5) instead of the number of visible faces.

Algorithm 3 shows the improved version of the algorithm 2. More strictly: let us suppose that two sets are given for a scene: a set of faces $F = \left\{f_i, 1 \leq i \leq n_f\right\}$

and a set of vertices $V = \left\{v_j, 1 \leq j \leq n_v\right\}$. The scene disjunction into a set of objects is supplied: $\Omega = \{\omega_k, 1 \leq k \leq n_\omega\}$, $V = \bigcup\limits_{k=1}^{n_\omega} \omega_i$, $k \neq l \Rightarrow \omega_l \bigcap \omega_k = \emptyset$. For each viewpoint s of the discrete sphere S the set of visible vertices $V(s) \subseteq V$ is given.

```
$\Omega = \{\omega_k, 1 \leq k \leq n_\omega\}; V = \bigcup\limits_{k=1}^{n_\omega} \omega_i; S; \tau$
$M = \emptyset$
**while** $\frac{I(M)}{I(S)} < \tau$ **do**
    Select $s \in S$ such as $I(M \bigcup \{s\}) = \max\limits_{s_i \in S} I(M \bigcup \{s_i\})$
    $M = M \bigcup \{s\}$
**end while**
```

Algorithm 3: The algorithm of automatic camera placement.

Let us denote the curvature in a vertex $v \in V$ as $C(v)$ and the total curvature of a mesh $V_1 \subseteq V$ as $C(V_1) = \sum\limits_{v \in V_1} C(v)$. We suppose that all objects in Ω have non-zero curvatures. In addition to equation 3.5, let us introduce the quality of a set of viewpoints:

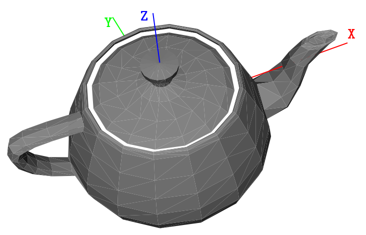
$$I(S_1 \subseteq S) = \sum_{\omega \in \Omega} q(\omega) \cdot \frac{\rho_\omega + 1}{\rho_\omega + \theta_{S_1,\omega}} \theta_{S_1,\omega},$$

where $\theta_{S_1,\omega} = \frac{C(V(S_1) \bigcap \omega)}{C(\omega)}$, $V(S_1) = \bigcup\limits_{s \in S_1} V(s)$. Since the camera remains outside the scene and always points to the center of the sphere, there is no need to define the view angle.

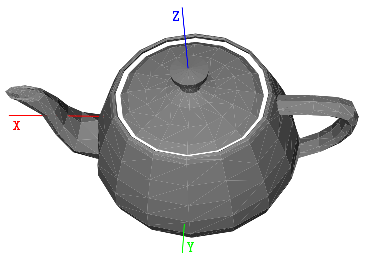
Now let us define a stopping condition: having given a threshold $0 \leq \tau \leq 1$, one should find a set of viewpoints $M_k \subseteq S$ such as $\frac{I(M_k)}{I(S)} \geq \tau$. At the beginning $M_0 = \emptyset$, each step i of the algorithm adds to the set the best viewpoint s_i: $I(M_{i-1} \bigcup \{s_i\}) = \max\limits_{s \in S} I(M_{i-1} \bigcup \{s\})$, $M_i = M_{i-1} \bigcup \{s_i\}$.

4.3.2 Camera placement examples

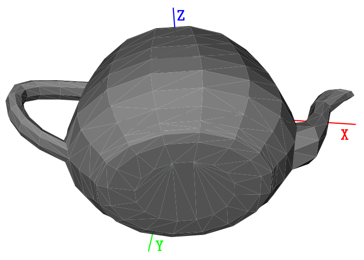
Now let us present several images illustrating the task of camera placement. Two well-known models will be shown here: the Utah teapot (figure 4.5) and the Stanford dragon (figure 4.6). The third model (figure 4.7) is very good for testing exploration techniques. It represents six objects embedded into holes on the sphere, and the explorer should not miss them. The views are selected using the algorithm 3. The images are sorted in descending order of amount of information they give. No additional information is provided, so, the bounding box sizes are taken as the importance function $q(\omega)$ and $\rho_\omega \equiv 0.1 \forall \omega \in \Omega$.



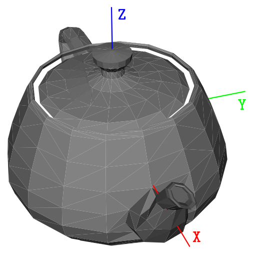
(a) First iteration (the best viewpoint): 40% of vertices are visible, 60% of surface is uncovered



(b) Second iteration: 43% of surface is uncovered

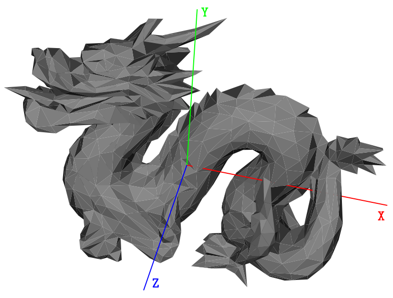


(c) Third iteration: 25% of surface is uncovered

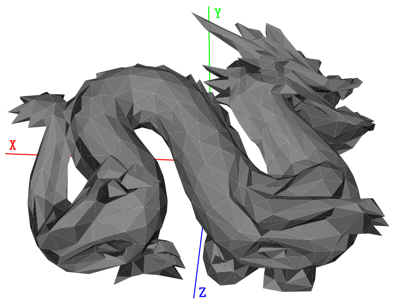


(d) Fourth iteration: 7% of surface is uncovered

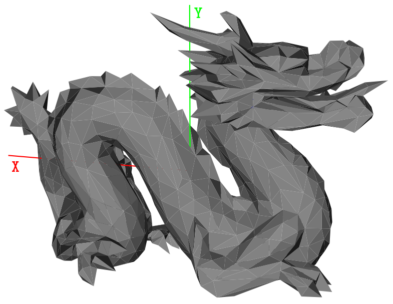
Figure 4.5: All vertices of the Utah teapot model are visible from 6 points of view, 4 points cover more than 93% of surface.



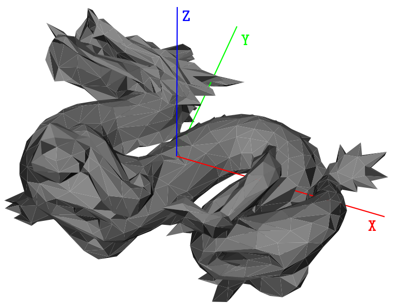
(a) First iteration (the best viewpoint): 38% of vertices are visible, 62% of surface is uncovered



(b) Second iteration: 37% of surface is uncovered

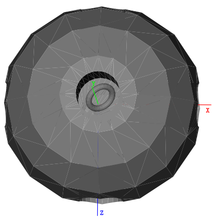


(c) Third iteration: 15% of surface is uncovered

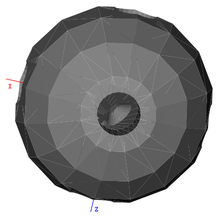


(d) Fourth iteration: 5% of surface is uncovered

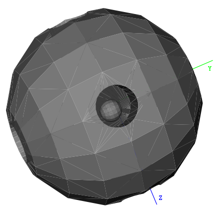
Figure 4.6: All vertices of the Stanford dragon model are visible from 8 points of view, 4 points cover more than 95% of surface.



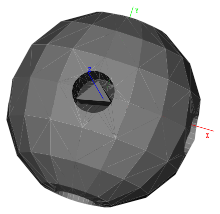
(a) First iteration (the best viewpoint): 71% of surface is uncovered



(b) Second iteration: 52% of surface is uncovered



(c) Third iteration: 24% of surface is uncovered



(d) Fourth iteration: 8% of surface is uncovered

Figure 4.7: This model is very good for testing exploration techniques, it represents six objects imposed into holes on the sphere, and the explorer should not miss them. 13 points of view, found by the algorithm, show all the surface of the object.

For the Utah teapot model the algorithm produces only 6 viewpoints even if we demand to observe all the surface (the stopping condition $\tau = 1$). First 4 iterations of the algorithm cover more than 93% of surface.

All vertices of the Stanford dragon model are visible from 8 points of view, 4 points cover more than 95% of surface.

The sphere with embedded objects is a good illustration why the stopping condition is to be introduced in the algorithm of the camera placement. 13 points of view are necessary to observe all the surface of the model, but even the first four iterations show more than 92% of the surface.

Figure 4.8 shows the amount of acquired information in dependence on the number of algorithm steps. It is interesting that for a simple scene consisting of one object even 4 points could show almost all its surface.

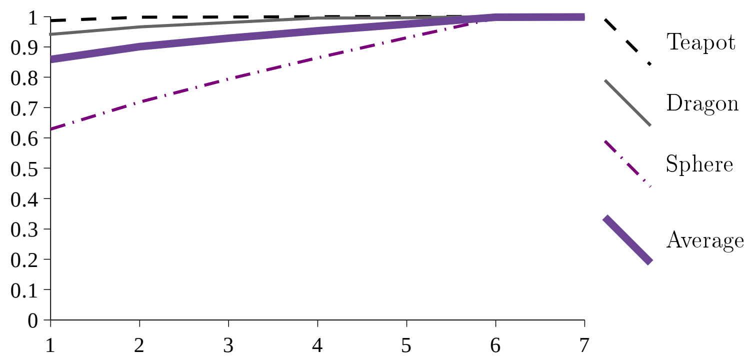


Figure 4.8: The amount of acquired information in dependence of number of viewpoints selected by greedy search for the Utah teapot model (figure 4.5), the sphere with embedded objects (figure 4.7) and the Stanford dragon (figure 4.6).

4.4 Dynamic understanding — making a movie

4.4.1 The criteria

A set of images is a good way to represent a not very complex scene. But sometimes it is not very easy to understand a scene if a set of viewpoints does not have a knowledge on how one can walk from one point to another. In such a case, the best solution is to give the user the ability to view a film made by a virtual camera, a film that gives the user a general knowledge of a scene.

The goal of this section is to develop a technique for a real-time global exploration. Moreover, we would like to develop a method to create films in real time with camera remaining outside the virtual world.

First of all, aesthetic criteria of the film quality must be defined:

1. The movie should not be very long, but it must show as much information as possible.
2. The operator (or an algorithm, if the movie is to be created automatically) should avoid fast returns to already visited points.
3. The camera path must be as smooth as possible, because a film with brusque direction changes is confusing for the user.
4. The operator should try to guide the camera via viewpoints as good as possible.

4.4.2 Incremental exploration

Algorithm

Here we propose an incremental construction method. The main idea is to determine where lie unexplored areas of a scene, then to create “magnets” in these areas. The camera is to be considered as a “magnet” of opposite polarity. Magnetic forces of large unexplored areas will attract the camera. In order to simplify the computations we use gravitational forces instead of magnetic ones.

The method is incremental, thus, having a trajectory line from the starting point to the current position of a camera, the camera is to be pushed towards unexplored areas. The aesthetic criteria could be satisfied with the following schema of exploration: at each step a mass is assigned to each point of the discrete sphere and to the current position of the camera. Then the camera point is put under the Newton's law of gravity. The superposition of the gravitational forces for the camera point is considered as the vector of movement.

Let us introduce a notation for the set of visible vertices with respect to the viewpoint p: $V(p) = \{v \in V | v \text{ is visible from } p\}$. $P_0^k = \{p_0, p_1, \ldots, p_k\}$ is the set of viewpoints (the camera trajectory), the set of explored vertices is $V\left(P_0^k\right) = \bigcup_{i=0}^{k} V(p_i)$. Exploration is started from the best viewpoint p_0, where the measure of viewpoint quality is expressed by equation (3.4).

Now, a new viewpoint p_{k+1} is to be appended to the trajectory P_0^k. Let us consider the point p_k, where camera stays. Since the camera cannot leave the surface of the sphere, the movement factors are represented by vectors lying in the tangent plane to the sphere in the point p_k. If a mass m is assigned to the point p, then in the tangent plane a force with the norm $||g(\vec{p})|| = \frac{m}{|(p_k,p)|^2}$ appears. The direction of $g(\vec{p})$ is the direction from p_k to p. It can be determined by taking the intersection line of the tangent plane and the plane, where the arc (p_k, p) of the sphere lies. Figure 4.9 shows an example of gravitational forces.

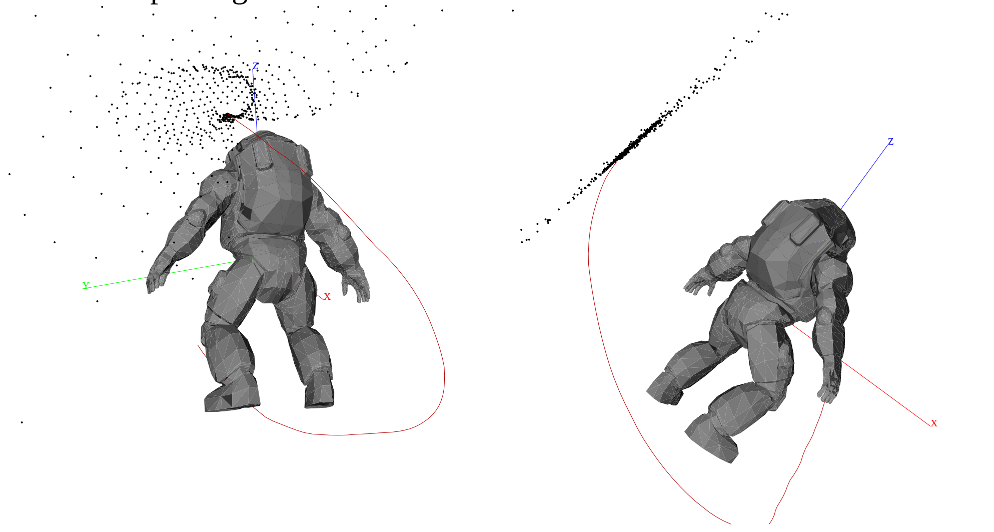


Figure 4.9: The black points indicate endpoints of the gravitational forces. They lie on the plane, tangent to the end of the trajectory. Note that the camera will not return to the visited areas since the forces in these directions are equal to zero.

Now the mass determination method is to be defined. In order to attract the camera to unexplored areas, we pose the following condition: more a viewpoint brings a new information, greater is its mass. Let us introduce a function evaluating the mass of the viewpoint $p \in S$ with respect to the camera trajectory P_0^k (of course, we suppose that $V(p) \neq \emptyset$):

$$m(p) = \frac{\left|N\left(p|P_0^k\right)\right|}{|V(p)|} \cdot I(p),$$

where:

1. $I(p)$ is computed according to equation (3.4)
2. $N\left(p|P_0^k\right) = V(p) \setminus \left(V(p) \bigcap V\left(P_0^k\right)\right)$ is the set of new vertices discovered from the viewpoint p.

Thus, $\left|N\left(p|P_0^k\right)\right|$ is the amount of the new information brought by the point p with respect to the trajectory P_0^k. Multiplier $I(p)$ forces the "operator" to guide the trajectory via good viewpoints. Algorithm 4 expresses the scheme.

```
Input: The set of vertices $V$, the discrete sphere $S$, the camera step size $c$
Output: The exploration trajectory $P$
$k \leftarrow 0$; $P \leftarrow \{p_0\} : I(p_0) = \max_{s \in S} I(s)$
while $V \setminus V(P_0^k) \neq \emptyset$ do
    $\vec{d_k} \leftarrow \sum_{s \in S \setminus \{p_k\}} g(\vec{s})$
    $\vec{d_k} \leftarrow \vec{d_k} \cdot \frac{c}{||\vec{d_k}||}$
    find $p_{k+1} \in S$ closest to $\vec{p_k} + \vec{d_k}$
    $P_0^{k+1} \leftarrow P_0^k \bigcup \{p_{k+1}\}$
    $k \leftarrow k + 1$
end while
```

Algorithm 4: The incremental algorithm computing the trajectory of the virtual camera.

It is obvious that the camera tries to move as quickly as possible to large uncovered areas, and does not return to visited areas, because the mass $m(p)$ of a visited point p is zero. Also, the camera does not have any brusque trajectory changes except in the case, where the camera finds a very interesting object suddenly. For example, as soon as the camera sees the interior of a building through a window, a

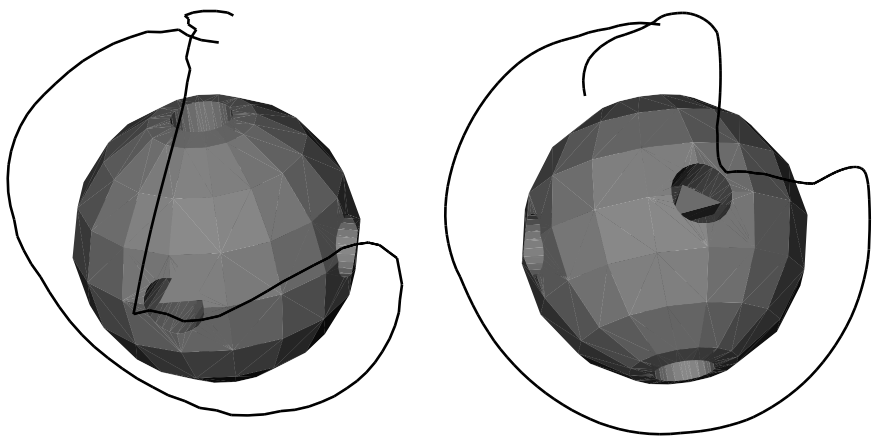

(a) The sharp edge in trajectory (b) The smooth trajectory

Figure 4.10: **(a)** The interesting object appeared suddenly and the big attractor disappeared. So, the camera wants to see another interesting object immediately. **(b)** Introducing the inertia factor a smooth trajectory is obtained.

big attractor disappears. So, the camera moves to see another uncovered area. Figure 4.10(a) shows an example of such a situation. If an inertia factor is introduced, then the trajectory becomes smoother (Figure 4.10(b)). More influence the inertia has on the trajectory, smoother path is obtained.

The simplest way to introduce the "inertia" is to take into account previous direction of move while computing the next one. For example, algorithm 4 at step *k* defines camera moving direction d_k as follows:

$$\vec{d_k} = \sum_{s\in S\setminus\{p_k\}} \vec{g_s} \cdot \frac{c}{\left\|\sum\limits_{s\in S\setminus\{p_k\}} g(\vec{s})\right\|}.$$

To introduce the inertia the step could be redefined:

$$\vec{d_k} = \sum_{s\in S\setminus\{p_k\}} g(\vec{s}) \cdot \frac{c}{\left\|\sum\limits_{s\in S\setminus\{p_k\}} g(\vec{s})\right\|} + c_i \cdot \vec{d_{k-1}},$$

where c_i is the inertance constant.

Thus, we have shown that the trajectory computed with the proposed method passes through the good viewpoints, it is rather short, it does not quickly return to the visited areas and it does not contain any sharp edges — all the criteria are satisfied.

Once one have seen all the scene, what next?

Sometimes it is not sufficient to view an object once, for example, if the user has passed some parts of the scene quickly, he (she), probably, did not understand it properly. So, it is better to show the parts of the scene several times. The method could be easily adapted to this task. If it is possible to "forget" parts already seen a long time ago, there will be always some regions of the object to explore.

The "forgetting" could be done in different manners, the simplest one is to determine the "size of memory", the time L, during which the algorithm keeps in mind a visited vertex. Then the amount of new information brought by the point p_{k+1} is:

$$N\left(p_{k+1}|P_0^k\right) = V(p_{k+1}) \setminus \left(V(p_{k+1}) \bigcap V\left(P^k_{\max(k-L,0)}\right)\right).$$

A bit more sophisticated method keeps in mind not a binary information about was a vertex visited, or not, but about the time of the last visit. More precisely, having a trajectory P_k^0, we define the time T of a vertex $v \in V$ visit as follows:

$$T\left(v|P_0^k\right) = \begin{cases} \max\limits_{i=0\dots k}\{i|v \in V(p_i)\} & \text{if } v \in V\left(P_0^k\right) \\ -1 & \text{if } v \notin V\left(P_0^k\right) \end{cases}$$

Then $m(p)$ is transformed to

$$\hat{m}(p) = \sum_{v \in V} \left(k - T\left(v|P_0^k\right)\right) \cdot I(p)$$

and a vertex not visited during a long time attracts the camera with more and more power until the camera sees it.

Trajectory dithering or how to avoid local minimums of energy

Another situation deserves special attention: sometimes, emulating gravity laws on high symmetry objects, two opposite factors will annulate each other. For example, let us look at figure 4.11. The model is shown from the start point (the best viewpoint). A big attractor (a large uncovered area) is situated at the opposite point of the viewpoint sphere.

The camera passes below the model, despite that a trajectory passing "equator"

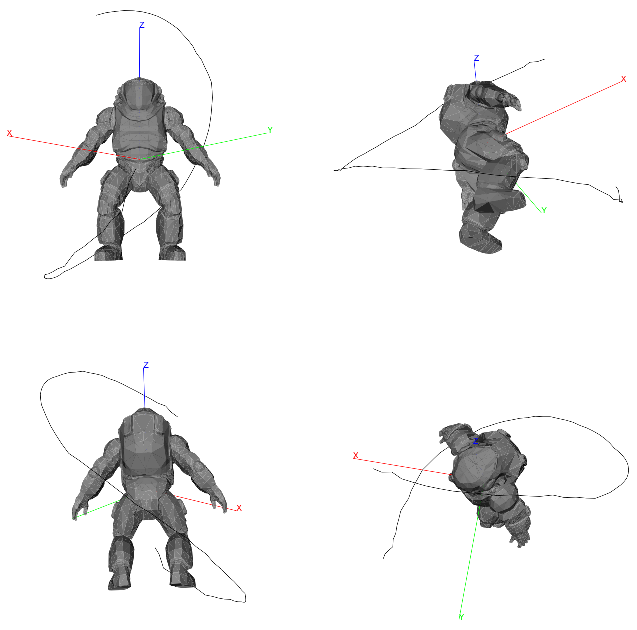


Figure 4.11: Trajectory computed without dithering. The images represent 4 different snapshots from the movie.

(supposing that a point above head is the “north pole”) is more interesting. Due to high symmetry of the object the “western” and the “eastern” hemispheres attract the camera equally. The opposite factors annulate each other and the camera passes near the “south pole”.

To avoid this effect we propose to slightly dither the calculated trajectory and see if we can obtain more interesting results. Figure 4.12 shows an illustration for the technique. Starting from the best view we slightly shake the end of the trajectory, and this allows to avoid local minimums of the gravity forces. The result, obtained by applying the method, can be found at figure 4.13.

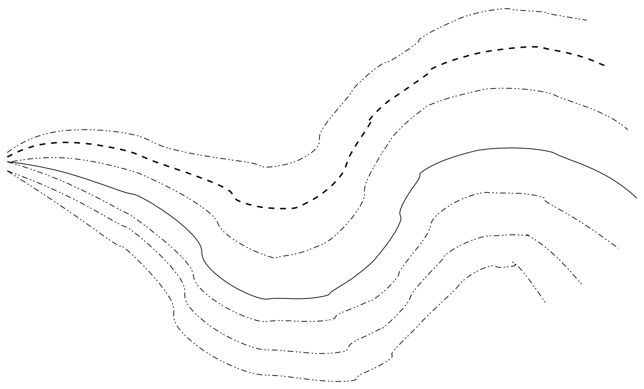


Figure 4.12: Every time we have calculated a trajectory section, we shall slightly dither it and see if we can obtain a better line. Here the initial trajectory is shown by solid line, curves obtained by dithering are drawn by dashed lines and the best is marked thick.

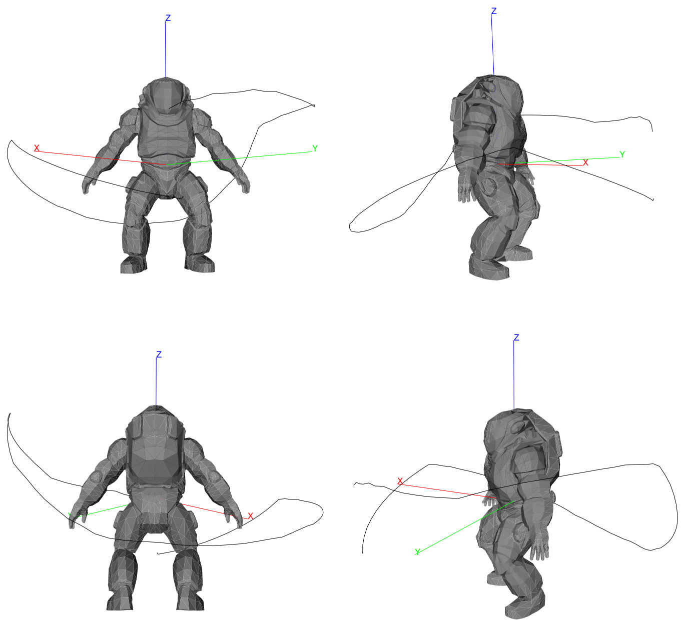


Figure 4.13: Trajectory computed applying dithering technique. The images represent 4 different snapshots from the movie.

Examples of trajectories

This section presents several exploration trajectories computed for different 3D scenes.

Figure 4.14 shows a trajectory obtained with the “gravitational” method of global scene exploration. The images are the snapshots taken consequently from the “movie”, the first one corresponds to the start of the movement, i.e., to the best viewpoint. Let us compare it with the trajectory, shown at figure 2.6. It is presented in [PSF04] article and is computed with the method introduced in [BDP99].

The trajectories are smooth and they have approximatively the same length. However, it is easy to see that the new method gives a better path. Figure 2.6 shows that the camera passes under the floor of the virtual office, and during this time the user does not see the scene. The new trajectory is free of this disadvantage.

The next example of the new method application is shown at figure 4.15. This model is very good for testing exploration techniques, it represents six objects imposed into holes on the sphere, and the explorer should not miss them.

None of the previously proposed methods can properly observe this model. All of them, having found an embedded object, are confused in choosing the next direction of movement. This happens because of missing information about unexplored areas. On the contrary, the new method operates with the analytic visibility graph, which allows to determine where some unexplored areas rest.

The exploration trajectory for the Utah Teapot is shown at figure 4.16.

Additional examples of trajectories (including protracted trajectories) can be found in appendix A.

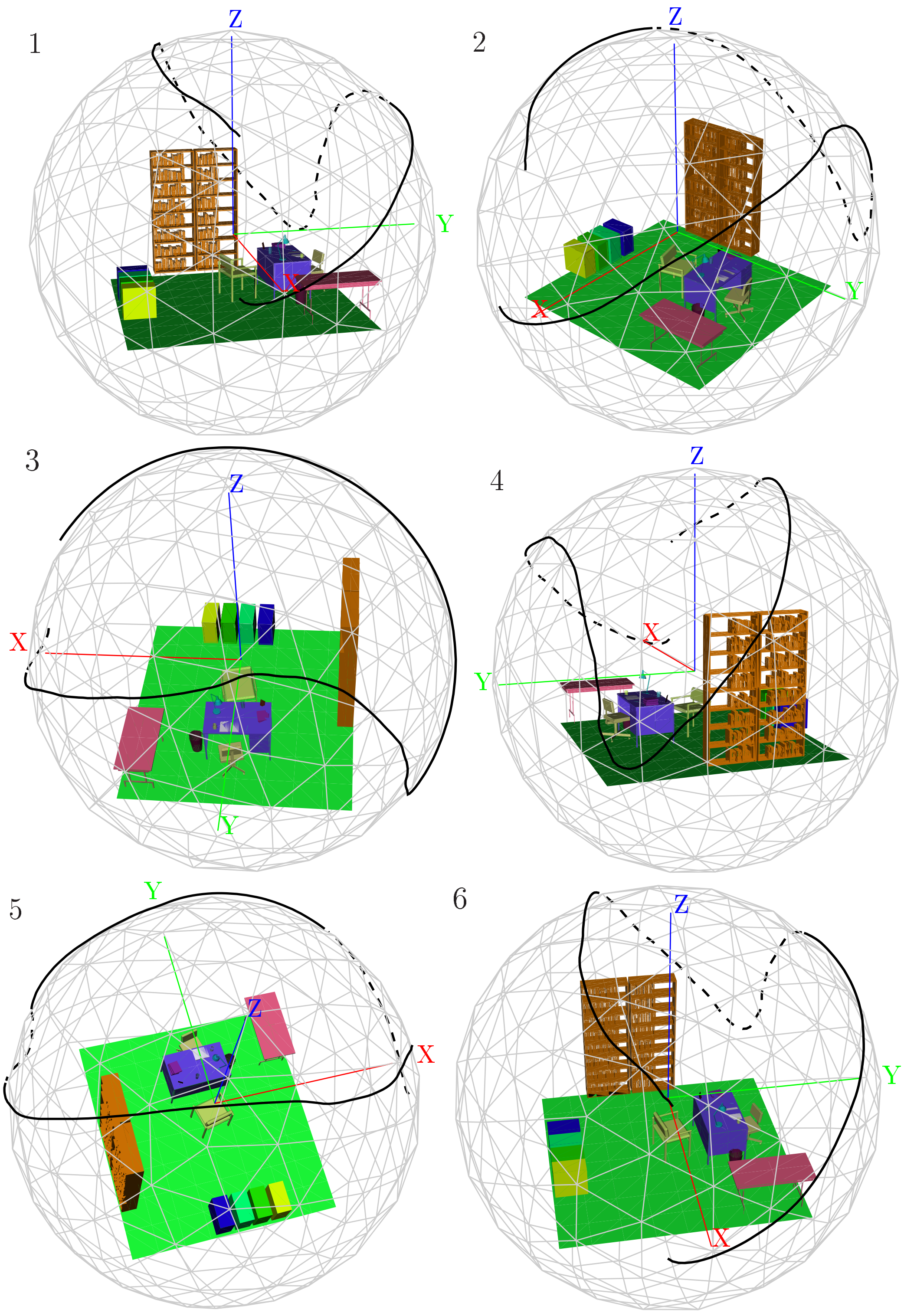


Figure 4.14: The exploration trajectory for the virtual office model computed with the new method. The images are the snapshots taken consequently from the "movie", the first one corresponds to the start of the movement (the best viewpoint).

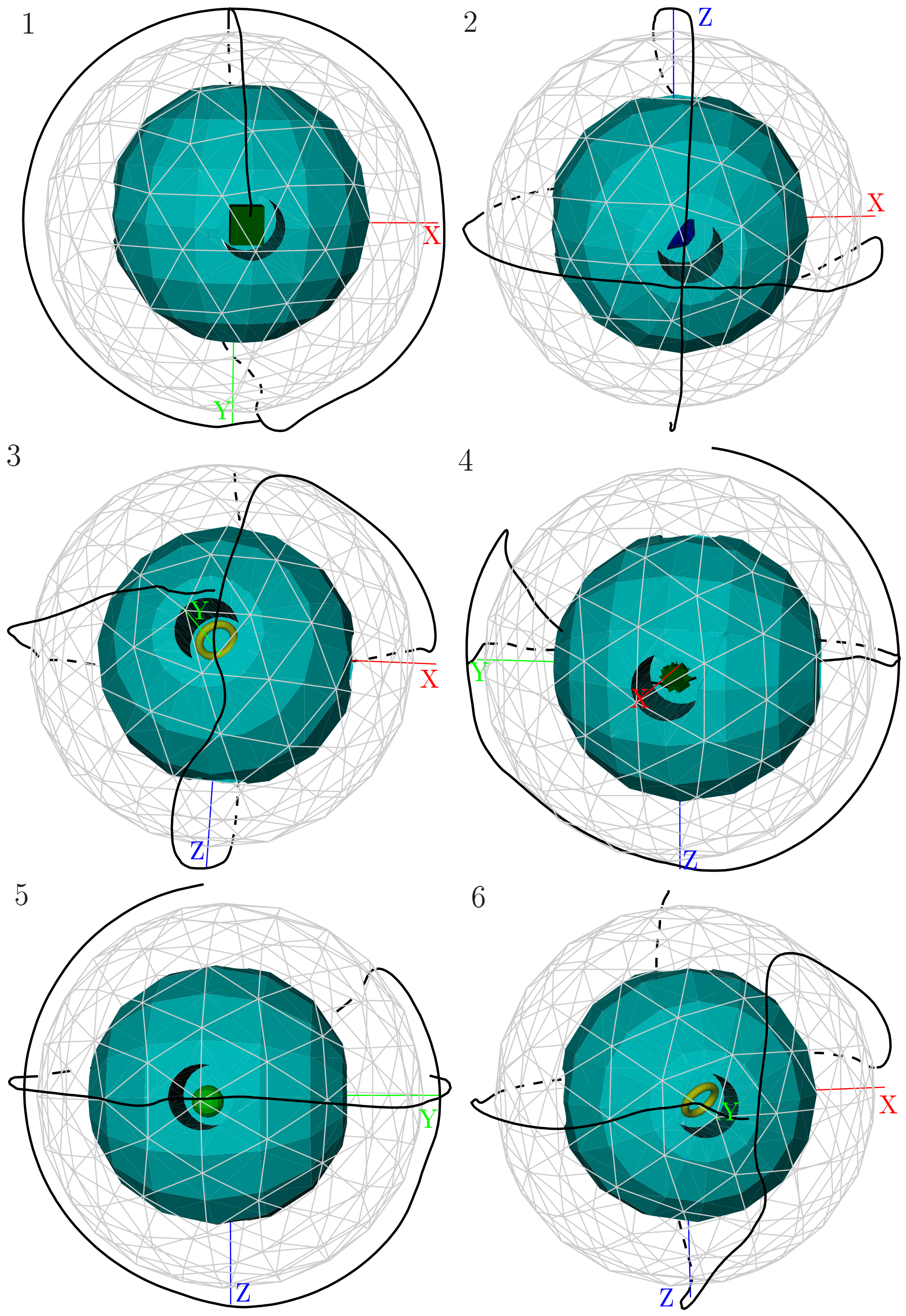


Figure 4.15: The exploration trajectory for the sphere with six embedded objects. The images are the snapshots taken consequently from the “movie”, the first one corresponds to the start of the movement (the best viewpoint).

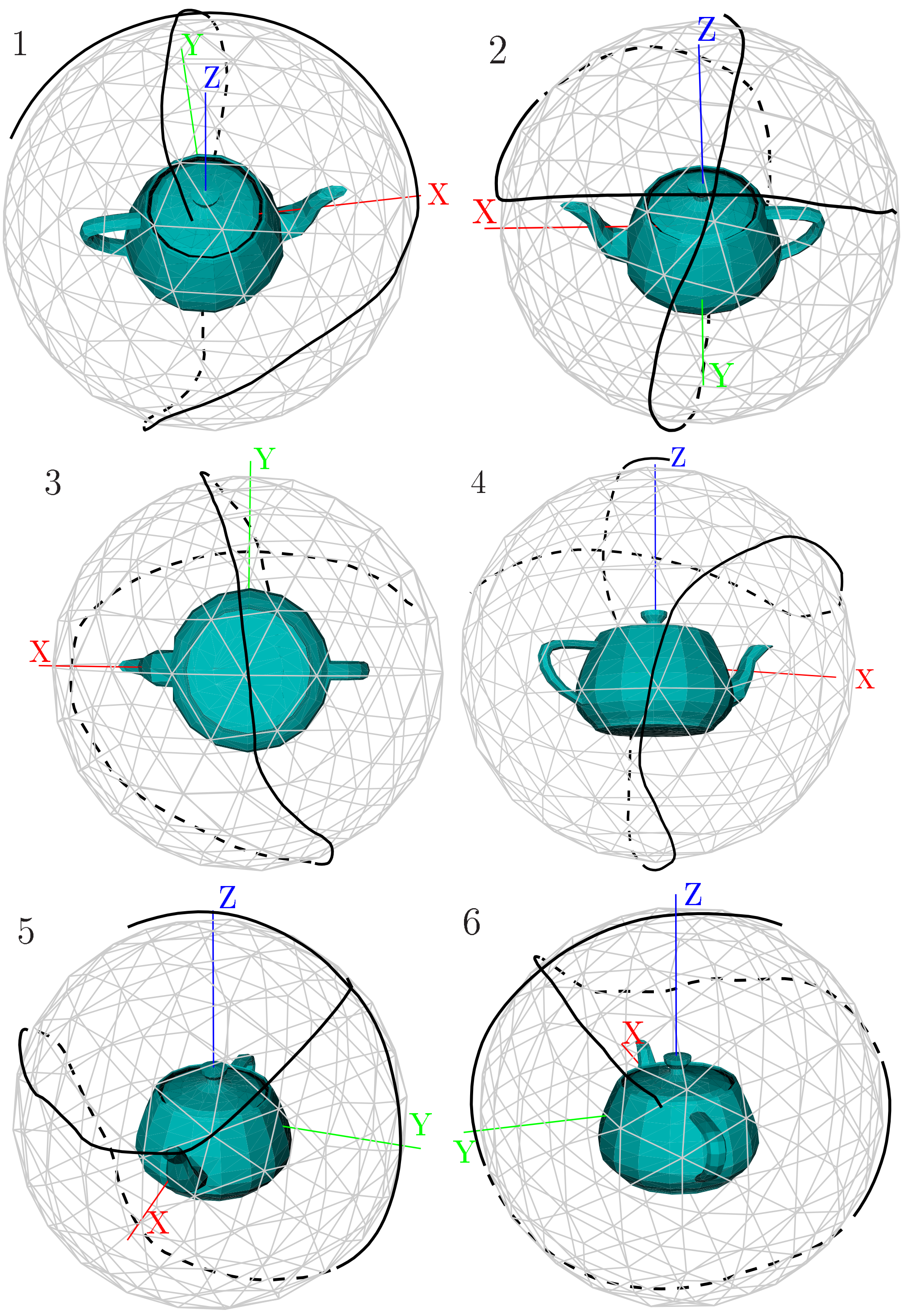


Figure 4.16: The exploration trajectory for the Utah teapot model. The trajectory is computed by the incremental method using the viewpoint entropy as the quality heuristic. Images are taken consequently from the "movie", the first one is the best viewpoint.

4.4.3 Non-incremental exploration

Now let us remember the algorithm of automatic camera placement from section 4.3. The main idea of the non-incremental algorithm of exploration is to connect the set of viewpoints with a trajectory.

Thus, the question is: if the camera has to move from one viewpoint to another, what path on the sphere is to be chosen? A naive answer is to connect the viewpoints with a geodesic line, the shortest one. This preserves the camera from brusque changes of trajectory during traversal from one point to another and gives the shortest solution, but acute angles still could appear in control points of trajectory. Such connection does not guarantee that the path consists of good viewpoints. This drawback is serious, and, in order to avoid it, we have to introduce additional costs and discounts.

Algorithm

The main idea is to make the distances vary inversely to the viewpoint qualities. It means the augmentation of path's length if it contains bad viewpoints and the reduction of the length otherwise. For example, it can be done in the following way: if two vertices s_1 and s_2 are adjacent in a sphere tessellation, then the new distance between s_1 and s_2 is calculated with the formula:

$$\hat{d}_{s_1,s_2} = ||\vec{s_1} - \vec{s_2}|| \cdot c_q(s_1, s_2), \tag{4.1}$$

where $||\vec{s_1} - \vec{s_2}||$ is the Euclidean distance between points s_1 and s_2 and $c_q(s_1, s_2) = 1 - \frac{I(s_1)+I(s_2)}{2\max\limits_{s \in S} I(s)}$ is the discount that forces the camera to pass via "good" viewpoints.

This empiric formula augments distances near "bad" viewpoints and reduces near "good" ones. Figures 4.17 and 4.18 illustrate the reason to introduce the discount c_q. It is easy to see that the camera trajectory presented at figure 4.18(b) brings to the user more information than the shortest one (figure 4.18(a)).

Now, having defined the metric and having found the set of viewpoints, we would like to determine a trajectory of the camera. It is not hard to construct a complete graph of distances $G = (M_k, E)$, where the weight of an arc $(v_1, v_2) \in E$ is equal to the metric between the viewpoints v_1 and v_2 (equation (4.1)).

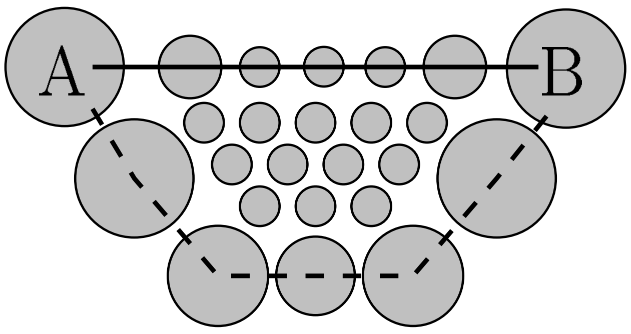
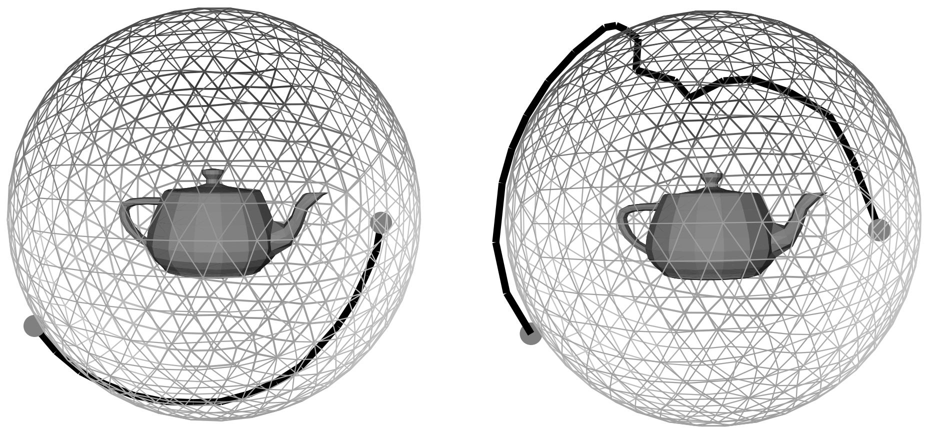


Figure 4.17: The reason to change the metric. The circles represent viewpoints: larger circles denote better viewpoints. The solid line shows the geodesic line between viewpoints A and B, the dashed line shows the shortest path according to the new metric. It is clear that sometimes it would be better to increase the length of the walk-trough in order to better inspect certain places.



(a) Shortest line connecting two viewpoints. (b) Shortest line with respect to the viewpoint qualities.

Figure 4.18: The trajectories between two selected points on the surface of the surrounding sphere.

Now the trajectory could be computed as the shortest Hamiltonian path (or circuit, if we would like to return the camera to initial point). The problem is also known as the *travelling salesman problem* (TSP). Unfortunately, the TSP problem is NP-complete even if we require that the cost function satisfies the triangle inequality. But there exist good approximation algorithms to solve the problem. Moreover, often $|M_k|$ is rather small, and the problem in such a case could be solved even by the brute-force algorithm in real-time.

Examples of trajectories

Figures 4.16 and 4.19 show camera trajectories for the Utah teapot model. The first one is obtained by applying the incremental technique with the viewpoint entropy as the quality heuristic, and the second one is obtained by non-incremental method. Both of them show 100% of the surface of the teapot model. The new method could give brusque changes of the trajectory, and the old one is free of this disadvantage. A simple way to smooth the trajectory is to construct a NURBS curve. Control points for the curve are to be taken from the approximation of the minimal set of viewpoints, and its order is to be defined by solving the TSP task. The new technique gives significantly shorter trajectories, and this advantage is very important.

One more example of the new method application is shown at figure 4.20. As we have already said above, this model is very good for exploration technique tests, it represents six objects imposed into the sphere with holes, and the explorer should not miss them. Compare it with the trajectory shown at figure 4.15. The trajectories are quite similar, but the new one is shorter almost two times than the old one.

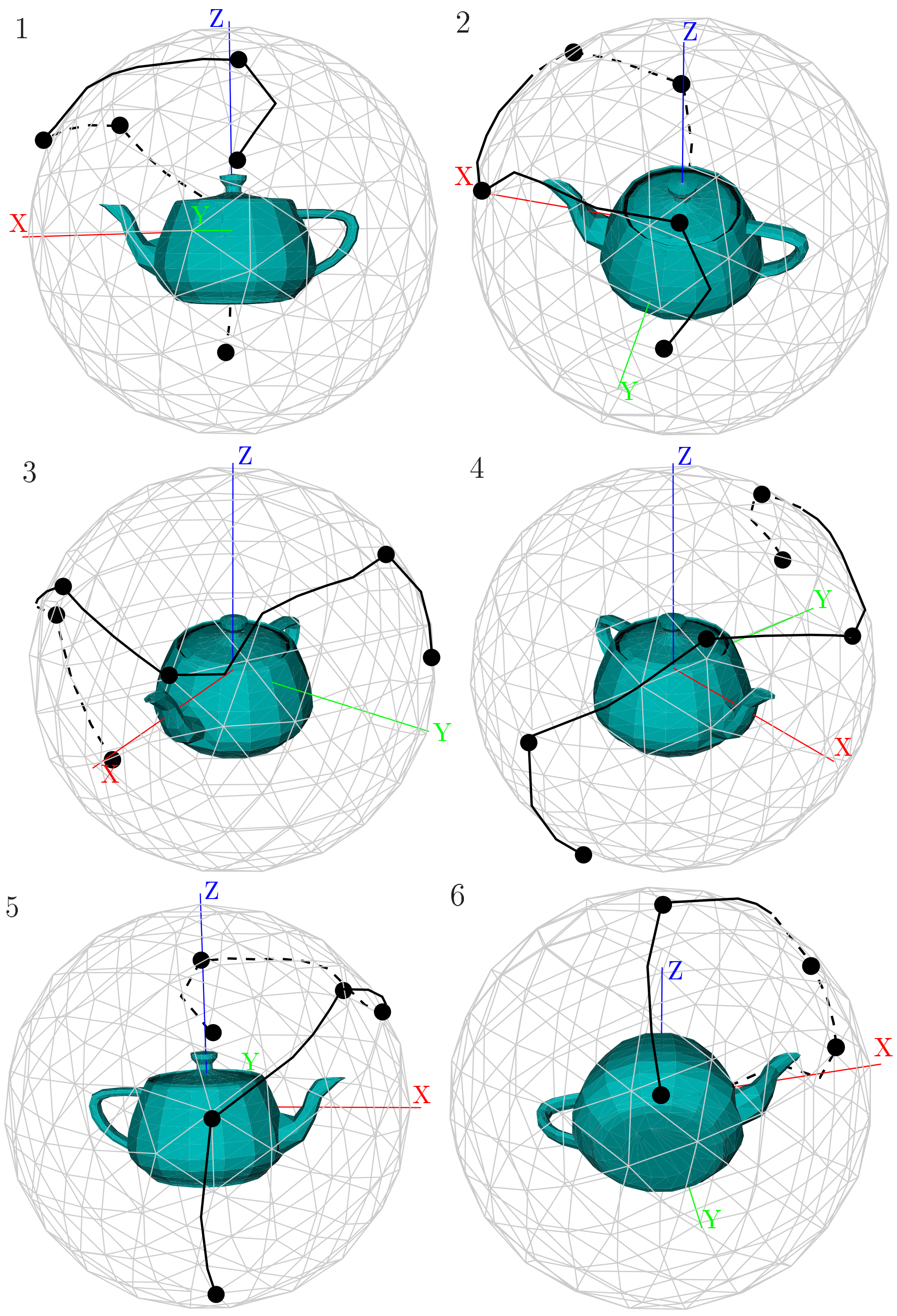


Figure 4.19: The exploration trajectory for the Utah teapot model, obtained with the new technique. Images are taken consequently from the "movie". Black knots are the control points of the trajectory, i.e. an approximation of the minimal set of viewpoints sufficient to see all the surface of the teapot model.

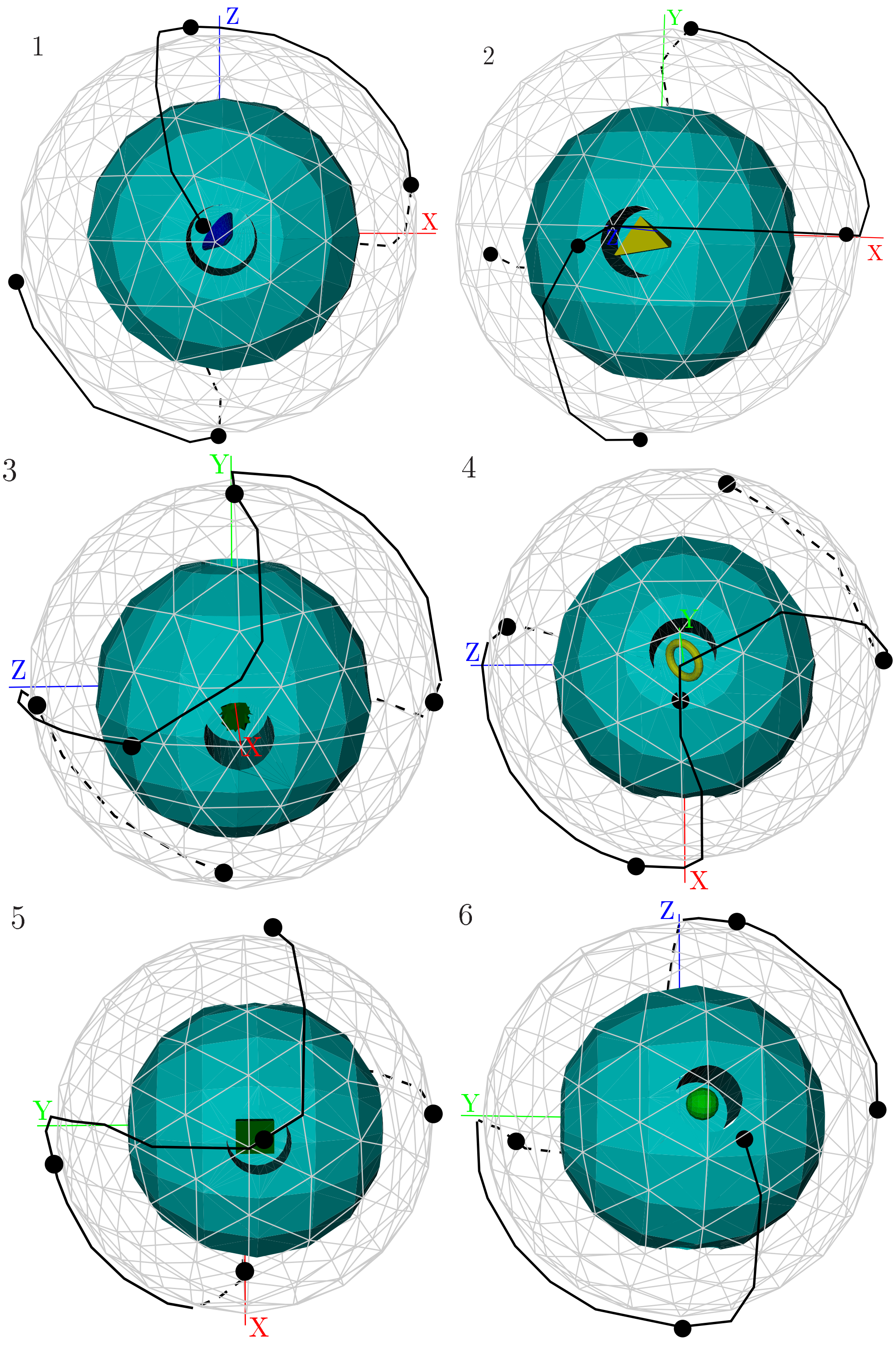


Figure 4.20: The exploration trajectory for the sphere with several embedded objects. Images are taken consequently from the “movie”, black knots are the control points of the trajectory.

4.5 Conclusion

In this chapter several approaches of virtual worlds exploration are described and discussed. In the beginning we have presented an approximate method that finds a minimal set of viewpoints sufficient to see all the scene. Then we show a very rapid method of viewpoint quality estimation.

Several methods of virtual “film making” (automatic scene exploration) are introduced too. The proposed techniques allow to get a good comprehension of a single virtual artefact or a general comprehension of a scene.

Note also that previously we have fixed an *offline* exploration of a virtual world as our purpose, but in the majority of cases the techniques, proposed in this chapter may work in *real-time*.

Chapter 5

Local exploration

5.1 Introduction

The previous chapter is dedicated to global scene exploration techniques. It is very helpful to give the user a general knowledge on a scene, but is not sufficient in cases, where it is necessary to explore some parts of a scene in detail. In this chapter we propose methods to automatically explore indoor scenes. Their objective is to navigate through the virtual environment to progressively show all the available information to the user.

A set of “photos” of a virtual world often gives a good knowledge of its structure. Section 5.2 describes a method of automatic camera placement for indoor scenes. When the photos are not elucidative, a dynamic method of exploration, proposed in section 5.3, could be used. It uses the set of “photos” as the control set and creates walk-throughs simulating human paths. Thus, in section 5.3 the camera is restricted to be at a constant height from the floor. Finally, section 5.4 introduces a technique of exploration, where the camera can freely navigate in 3D space.

5.2 Automatic camera placement

As we have said before, in this section the camera is supposed to be at a constant height from the floor to simulate a human path. The methods are designed for environments with flat (or almost flat) terrains. In case when the ground is strongly deformed, the scene could be decomposed. In order to reduce the amount of computations, the set of viewpoints is discretized.

In section 4.2 we have introduced a fast algorithm for visibility computations. It can be easily adapted to calculate the visibility graph for an indoor scene. In the adaptation of the algorithm the set of viewpoints is considered as a display to draw scene triangles. The algorithm iterates through each vertex of the scene and finds all pixels of the plane which are visible from the given vertex. Figure 5.1 gives an illustration.

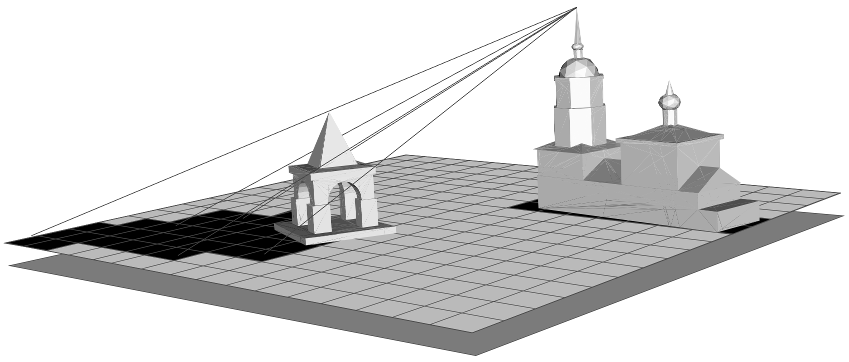


Figure 5.1: The solid bottom plane is the ground plane, the rasterized one represents a set of viewpoints to be processed. The church spire is visible from gray pixels and is not visible from the black ones.

Having the viewpoint quality criterion (equation (3.5)) and the fast algorithm for visibility computations, we are ready to choose good views. The "photos" should be as good as possible (provide as much information as possible) and the number of photos should not be very great. These criteria are satisfied with a greedy search scheme. Let us give a more strict formulation.

Let us suppose that there is a scene, i.e., that two sets are given: a set of faces $F = \left\{f_i, 1 \le i \le n_f\right\}$ and a set of vertices $V = \left\{v_j, 1 \le j \le n_v\right\}$. The scene disjunction into a set of objects is provided: $\Omega = \{\omega_k, 1 \le k \le n_\omega\}$, $V = \bigcup\limits_{k=1}^{n_\omega} \omega_i$, $k \neq l \Rightarrow \omega_l \bigcap \omega_k = \emptyset$. A set of viewpoints S is provided, a bipartite analytic visibility graph $G = (S \bigcup V, E)$ is computed.

Nobody has eyes in the back of his head, so, the angle of view and a view direction should be provided with a viewpoint in order to get a photograph. In order to simplify calculations, we use a set D of predefined view directions (refer

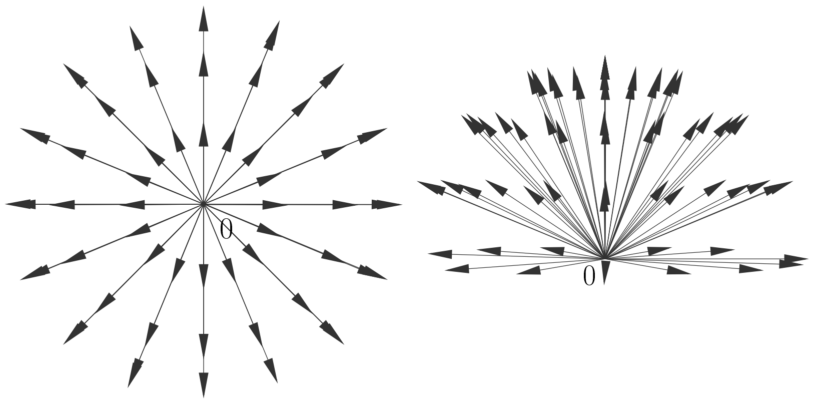


Figure 5.2: 65 predefined directions of view.

to figure 5.2 for illustration). The set D consists of 65 vectors with the start points in the origin and the endpoints lying in the unit hemisphere.

Thus, a set of photos is determined by the view angle α and a set of pairs $\{s, d\}$, $s \in S$, $d \in D$. Let us introduce the notation of a vector: if there is a point p, then $\vec{p}$ is the vector from the origin to p. The vertex $v \in V$ is visible from the viewpoint s in the view direction d, if the arc (s, v) exists in the graph G and the vector $\vec{v} - \vec{s}$ stays in the cone with the axis $\vec{d}$ and the angle α:

$$V_\alpha(\{s, d\}) = \left\{ v \in V | (s, v) \in E, \cos\frac{\alpha}{2} \leq \frac{\vec{d} \cdot (\vec{v} - \vec{s})}{||\vec{d}|| \cdot ||\vec{v} - \vec{s}||} \right\}.$$

Let us denote the curvature in a vertex $v \in V$ as $C(v)$ and the total curvature of a mesh $V_1 \subseteq V$ as $C(V_1) = \sum_{v \in V_1} C(v)$. We suppose that all the objects in Ω have non-zero curvatures. In addition to equation (3.5), let us introduce the quality of a set of viewpoints:

$$I_\alpha(X \subseteq S \otimes D) = \sum_{\omega \in \Omega} q(\omega) \cdot \frac{\rho_\omega + 1}{\rho_\omega + \theta_{\alpha,X,\omega}} \theta_{\alpha,X,\omega},$$

where $\theta_{\alpha,X,\omega} = \frac{C(V_\alpha(X) \bigcap \omega)}{C(\omega)}$, $V_\alpha(X) = \bigcup_{x \in X} V_\alpha(x)$.

A set of viewpoints, giving a good representation of a scene, could be obtained by a greedy search. The greediness means choosing a best viewpoint at each step of the algorithm. More strictly: having given a threshold $0 \leq \tau \leq 1$, one should find a set of viewpoints $M_k \subseteq S \otimes D$ such as $\frac{I_\alpha(M_k)}{I_\alpha(S \otimes D)} \geq \tau$. At the beginning $M_0 = \emptyset$, each step i of the algorithm adds to the set the best view (s_i, d_i). $M_i = M_{i-1} \bigcup \{(s_i, d_i)\}$:

$$I_\alpha \left(M_{i-1} \bigcup \{(s_i, d_i)\} \right) = \max_{x \in S \otimes D} I(M_{i-1} \bigcup \{x\}).$$

Figure 5.3 shows a set of “photos” of a virtual town model made by the algorithm. A map of the town is shown at figure 5.4. The viewpoints are indicated on the map too.

In this scene each building represents a single object. The importance contribution $q(\omega)$ is taken equal to 10 for the churches and the chapels, and equal to 2 for the ordinary buildings. Predictability parameter ρ_ω is equal to 1 for the churches and chapels and equal to 0.1 for other buildings. The stopping condition τ was taken equal to 0.95 and the view angle is equal to $\frac{\pi}{3}$.

5.3 Creating a walk-through

5.3.1 A single trajectory

A set of images is a good way to represent a simple scene. But sometimes it is not very easy to understand a scene if the set of viewpoints does not supply information on how one can walk from one point to another. Refer to figure 5.3 for an example. The scene is not complex, but the set of photos confuses the user. In such a case, the best solution is to give the user the ability to view a film made by a virtual camera, a film that gives the user a general knowledge of a scene.

The goal of this section is to develop a local exploration technique. We would like to develop a method to create films in real time with the camera remaining inside the virtual world.

Let us remember the aesthetic criteria of the film quality which we defined in section 4.4.1:

1. The movie should not be very long, but it must show as much information as possible.
2. The operator (or an algorithm, if the movie is to be created automatically) should avoid fast returns to already visited points.

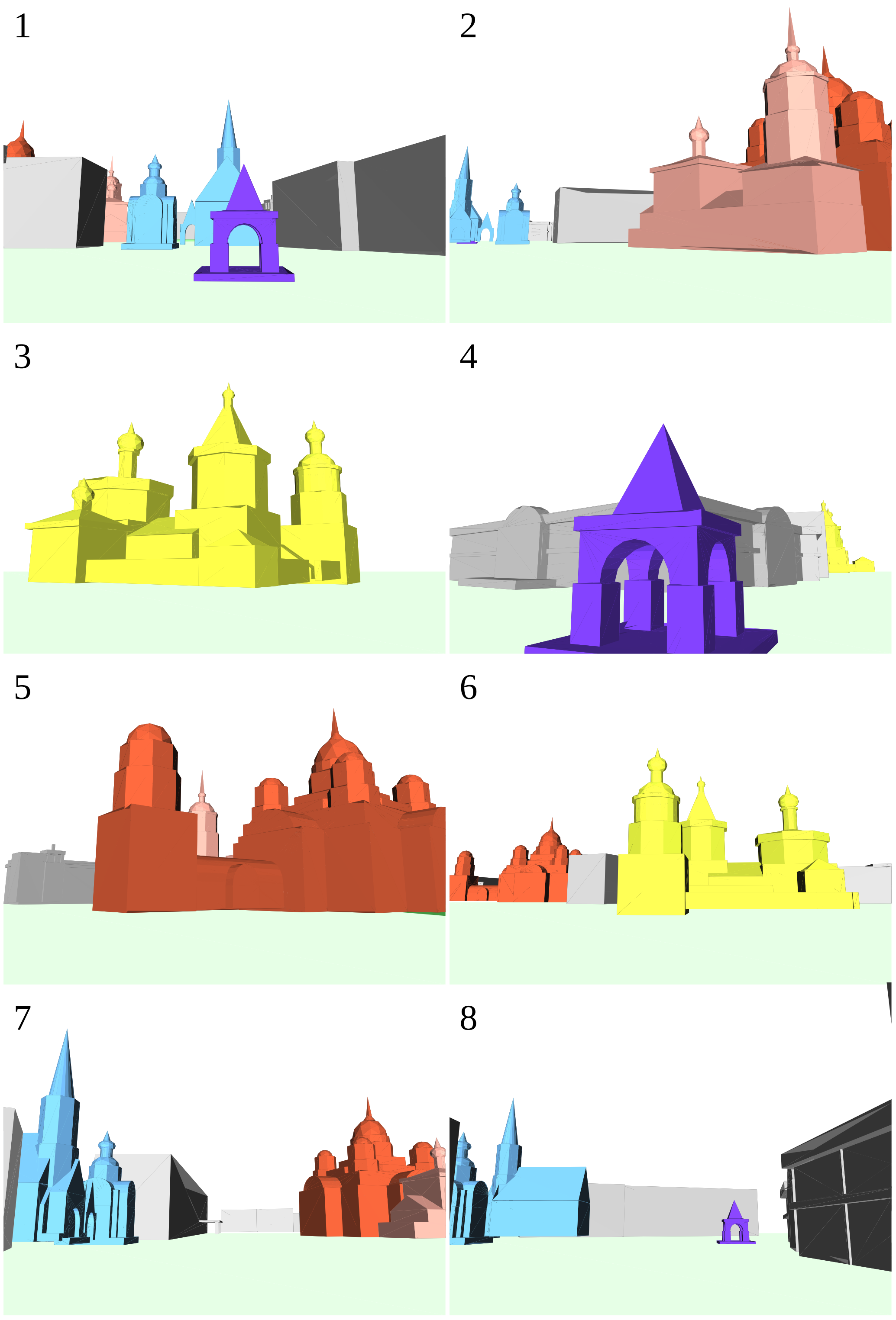


Figure 5.3: The figure shows eight views in the order they were selected by the greedy search algorithm. Refer to figure 5.4 for a map of the virtual town.

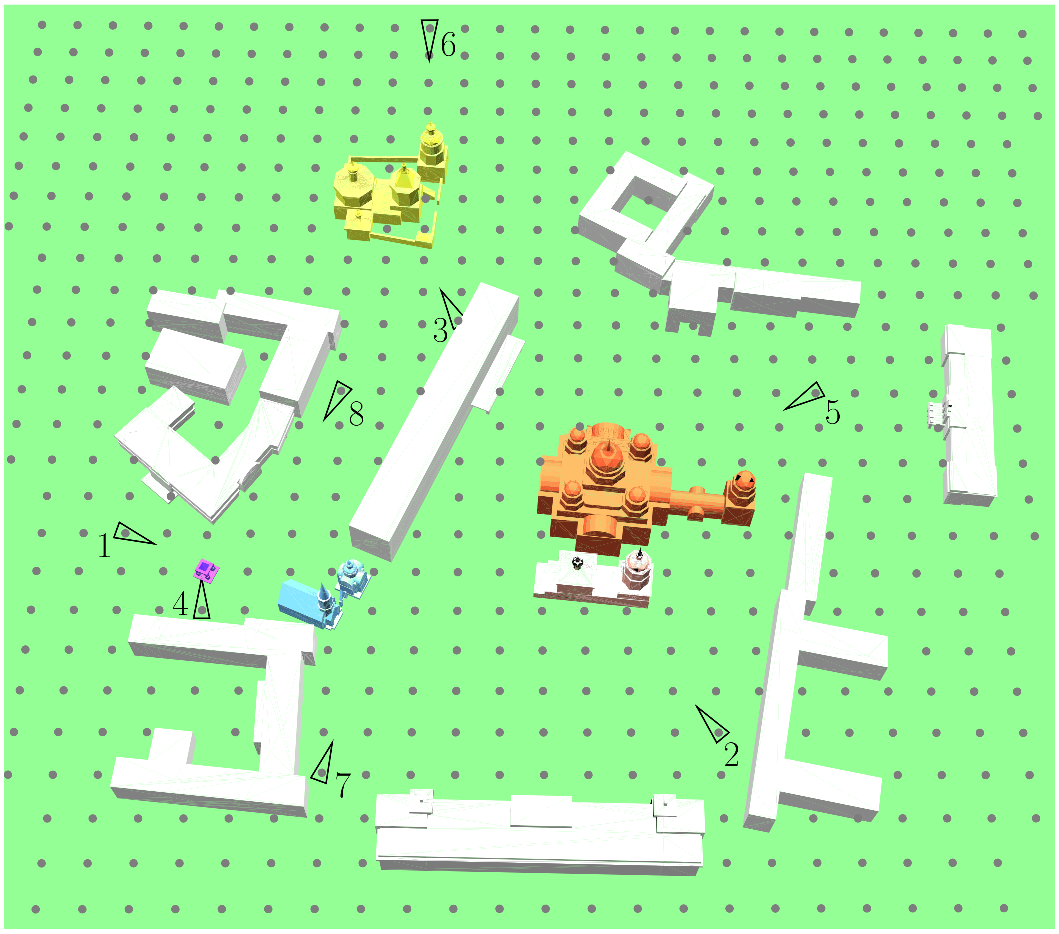


Figure 5.4: The virtual town map. The dark dots represent the set of viewpoints. Black triangles indicate eight points and directions of view selected by the greedy search algorithm. The viewpoints are numbered in the order they were selected by the greedy search algorithm. Refer to figure 5.3 for the "photos".

3. The camera path must be as smooth as possible, because a film with brusque direction changes is confusing for the user.

4. The operator should try to guide the camera via as good viewpoints as possible.

In addition to these criteria there is also the problem of *collision detection* — the camera should not pass through objects.

The last condition could be satisfied with the following schema of exploration: an analytic visibility graph $\hat{G} = (S, \hat{E})$ is to be calculated, where an arc $(s_1, s_2) \in \hat{E}$ if and only if $s_1 \in S$ is visible from $s_2 \in S$ (there are no obstacles between the viewpoints). Then the weight function $w(e) : \hat{E} \to \mathbb{R}^+$ is to be specified for each arc of $\hat{E}$. For example, the Euclidean distance could be considered as such a weight function. Then if there are two arbitrary viewpoints a and b and there is a linked component of the graph which contains both a and b, the shortest path between the viewpoints can be found using the Dijkstra's algorithm. There will be no collision between the broken line (the shortest path) $P = (a = p_1, p_2, \dots, p_n = b)$ and objects of the scene, due to consequent visibility between viewpoints p_i and p_{i+1}, $1 \le i \le n - 1$.

The graph $\hat{G}$ can be quickly computed using a slight adaptation of algorithm 1. The visibility is to be computed between two viewpoints instead of visibility between viewpoints and vertices of a scene. All triangles that do not intersect the viewpoint plane are to be eliminated. The rest is to be projected as it is shown at figure 5.5. The total running time of the algorithm is $O\left(n_f \cdot n_s^{3/2}\right)$ operations in the worst case.

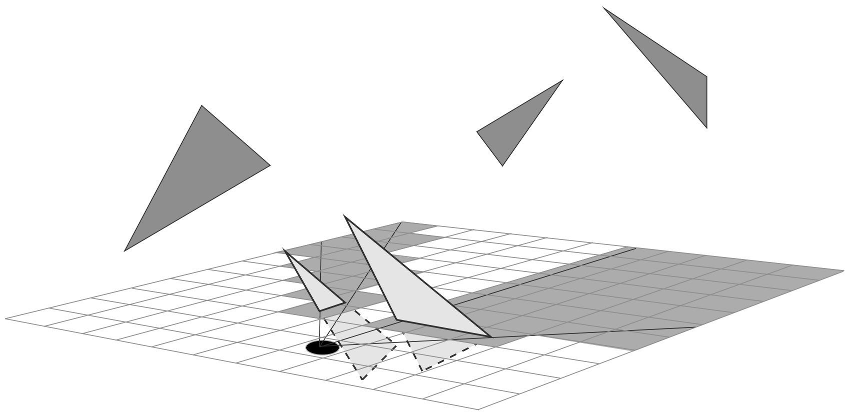


Figure 5.5: The rasterized plane represents a set of viewpoints to be processed. The dark triangles are eliminated due to Z-coordinates. The viewpoint marked by the black circle is visible from white pixels and is not visible from the gray ones.

There is no need to calculate the visibility between all pairs of viewpoints. In order to increase performance, it is better to define a radius r and to calculate for each viewpoint $s \in S$ the visibility between it and its neighbours: $S(s, r) = \{s_i \in S \setminus \{s\} : \|\vec{s_i} - \vec{s}\| \le r\}$. In such a case, the method is less sensible to perturbations in the set of viewpoints.

Now we can choose paths with respect to obstacles. The next question is: if there are two viewpoints and the camera has to move from one to another, which path is to be chosen? A naive answer is to connect the viewpoints with the shortest path ($w(e) : \hat{E} \to \mathbb{R}^+$ is the Euclidean distance). But it does not guarantee that the path consists of good viewpoints. This drawback is serious, and, in order to avoid it, we have to introduce additional costs and discounts.

Let us suppose that there are two views (s_1, d_1) and (s_2, d_2), where each one is visible from the other: $(s_1, s_2) \in \hat{E}$. Note that we do not consider the view directions d_1 and d_2 in this definition. We just suppose that there are no obstacles between points s_1 and s_2.

Let us consider factors influencing the quality of a film. We suppose that the resulting weight function is the superposition of different costs and discounts. First of all, the "operator" (or an algorithm) should take into account the cost of moving the camera from one point to another:

$$c_m\left((s_1, d_1), (s_2, d_2)\right) = \|\vec{s_2} - \vec{s_1}\|.$$

The cost of turning the camera is to be considered also:

$$c_t\left((s_1, d_1), (s_2, d_2)\right) = \arccos \frac{\vec{d_1} \cdot \vec{d_2}}{\|\vec{d_1}\| \cdot \|\vec{d_2}\|}.$$

Now let us introduce the discount c_q that forces the camera to pass via "good" viewpoints. The following empiric formula augments displacement costs between "bad" viewpoints and reduces near "good" ones.

$$c_q\left((s_1, d_1), (s_2, d_2)\right) = 1 - \frac{I_\alpha(s_1, d_1) + I_\alpha(s_2, d_2)}{2 \max\limits_{x \in S \otimes D} I_\alpha(x)}.$$

Figure 5.6 shows an illustration for the costs. There are three different paths from the left viewpoint to the right one. Paths 1 and 3 have the same length and their intermediate points show equal information (the church is symmetric). But one can conclude that 3 is worse, because it requires more efforts for camera turns. Path 2 is shorter than 1, but in spite of this 1 is better, because its intermediate viewpoint shows the church entirely, while only a small part of the wall is visible from the intermediate viewpoint of path 2. Thus, the first path is the best one.

Thus, the weight function for an arc between two viewpoints could be expressed

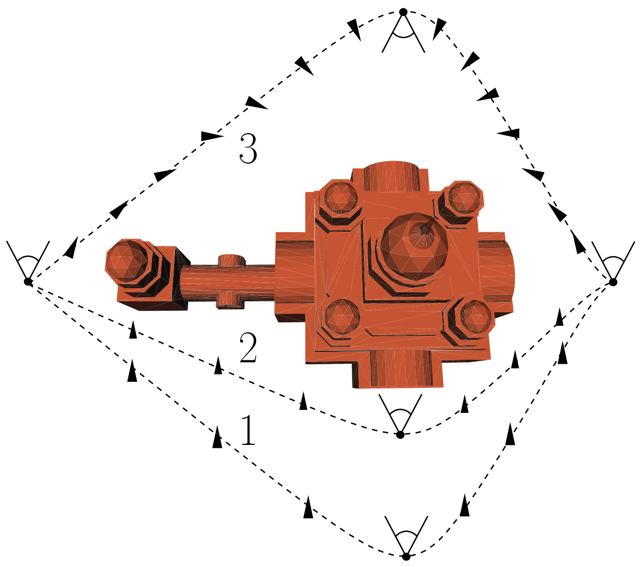


Figure 5.6: Three different paths from the left viewpoint to the right one are shown in dashed lines. Paths 1 and 3 have the same length, but 3 is worse, because it requires more efforts for camera turns. Path 2 is shorter than 1, but in spite of this 1 is better, because its intermediate viewpoint shows the church entirely, while only a small part of the wall is visible from the intermediate viewpoint of path 2.

as follows:

$$\begin{aligned} w\left((s_1,d_1),(s_2,d_2)\right) &= \alpha_m \cdot c_m\left((s_1,d_1),(s_2,d_2)\right) + \\ &+ \alpha_t \cdot c_t\left((s_1,d_1),(s_2,d_2)\right) + \\ &+ \alpha_q \cdot c_q\left((s_1,d_1),(s_2,d_2)\right), \end{aligned}$$

where α_m, α_t and α_q are the coefficients of relative importance of the different costs. For example, if the wage of viewpoint quality is more important than the cost of camera displacement, α_q is to be chosen greater than α_m. Having defined the weight function, a trajectory between two views is to be obtained by the Dijkstra's algorithm.

Now let us remember the set of "photos" from the previous section. So, one has found the set $M_k \subseteq S \otimes D$ of viewpoints, and the trajectory of the camera is to be determined. We solve the problem by converting it to the Travelling Salesman Problem (TSP). The control set of "photos" is the set of cities to visit. The travelling costs between pairs of cities are to be determined as the length of the shortest paths between them in the graph $\hat{G}$.

The resulting trajectory could be computed as the shortest Hamiltonian path (or circuit, if we would like to return the camera to initial point). Note that this instance of TSP is natural and satisfies the triangle inequality constraint. That is, c_q breaks the triangle inequality in the graph $\hat{G}$, but the TSP instance satisfies it. Indeed, the travelling costs are defined as lengths of *shortest* paths in $\hat{G}$, thus, for any two cities A and B there is no circuit shorter than the shortest path between A and B. Unfortunately, the TSP problem is NP-complete even if we require that the cost function satisfies the triangle inequality. But there exist good approximation algorithms to solve the problem. For example, Christofides in [Chr76] has proposed a constant-factor approximation algorithm which always finds a tour of length at most 1.5 times the shortest tour.

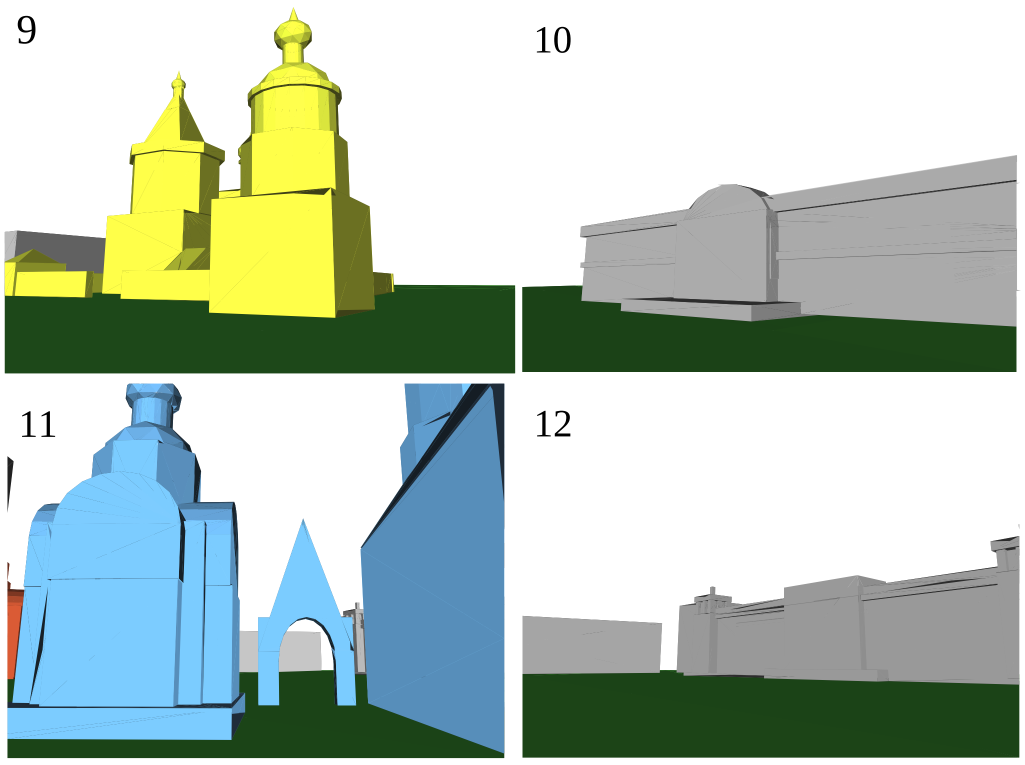


Figure 5.7: The figure shows four additional "linking" viewpoints added during solving the TSP. Refer to figure 5.8 for a map of the virtual town.

Figures 5.7 and 5.8 show the final exploration trajectory for the virtual town we have already met. The viewpoint visiting order was determined by the TSP solution. Note also the solid black triangles which were added during the resolving process of the problem. These triangles indicate additional viewpoints to visit in

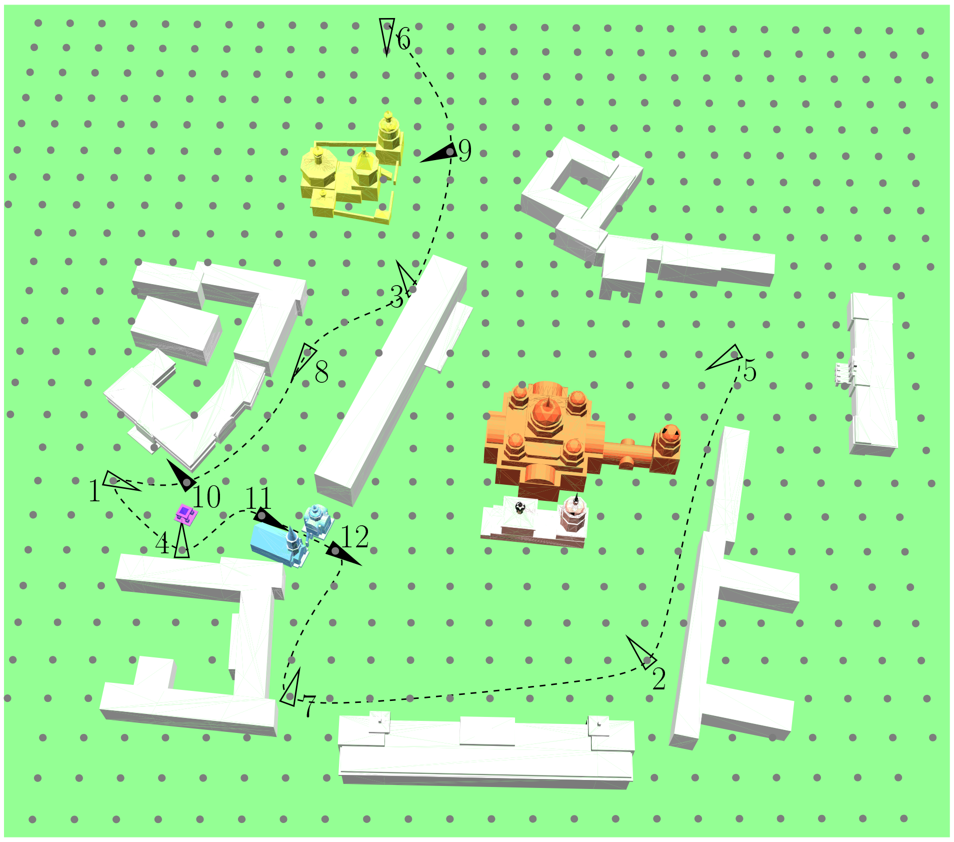


Figure 5.8: The filled triangles indicate additional “linking” views, the dashed line shows the final camera trajectory. The viewpoints are numbered in order they were found. Refer to figures 5.3 and 5.7 for the “photos”.

order to avoid collisions with objects. Figure 5.7 shows the “photos”. Note that the additional viewpoints 11 and 12 make the archway accessible during traversal from the point 4 to 7. When the additional viewpoints were found and the visiting order was determined, the trajectory is obtained by creating an open interpolated spline curve. The cost function c_q forces the trajectory to pass through good viewpoints, the TSP solution gives short trajectory without quick returns to visited areas. Thus, all the aesthetic criteria are satisfied.

5.3.2 Splitting a trajectory

A solution of the TSP instance produces a trajectory which traverses all control views. But nothing forces us to produce a *single* trajectory. Suppose, one should observe a city. Cities contain a lot of touristic places, often they are far from each other. And it is quite convenient to take a subway to reach consequently the sights to be observed. Therefore, we may produce *a set* of walk-throughs. Let us note that a photo is a degenerated walk-through, a trajectory of zero length.

One can remove long paths from the final trajectory, but it is more logical (and less expensive) to split the initial instance of the TSP. This could be done in different ways. The simplest one is to define a threshold value and to eliminate all edges longer than the threshold. Another way is to consider the travelling costs between the control views as a random variable, and then to use a method of gross error detection. All long paths will be removed. Often it is sufficient to eliminate all paths longer than the mean value plus the quadratic error for the random variable.

Figure 5.9 shows the final set of exploration trajectories for a virtual town. The problem was splitted into three TSP instances by removing all paths longer than the mean value plus the quadratic error.

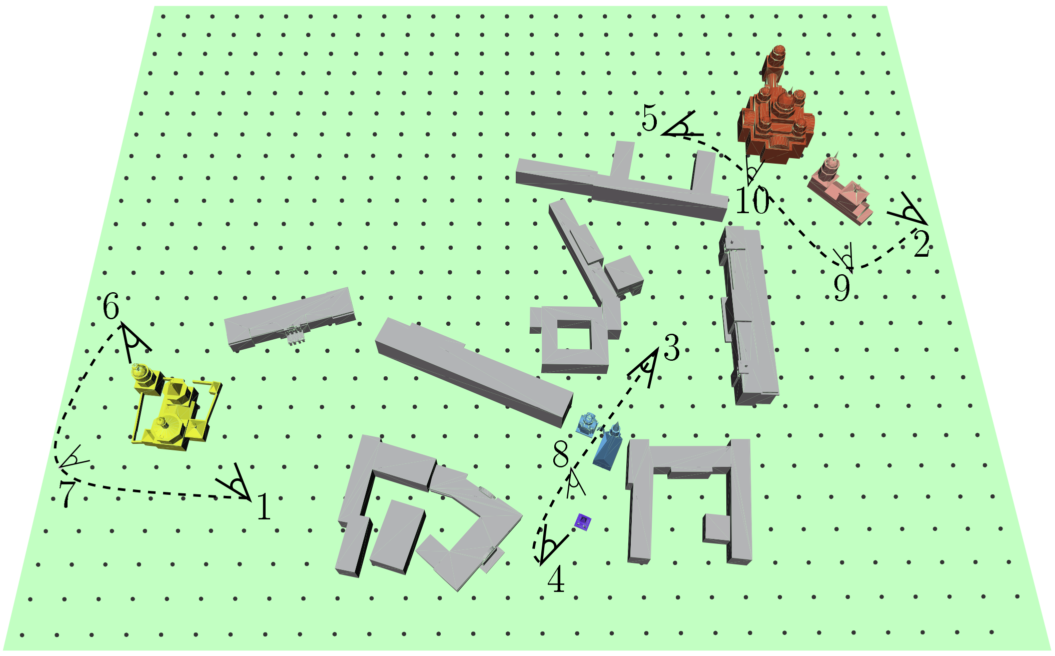


Figure 5.9: The viewpoints 1–6 are found by the automatic camera placement algorithm. The viewpoints 7–10 indicate additional “linking” views. The dashed lines show three camera trajectories. The viewpoints are numbered in order they were found.

5.4 Flying like a bird

The previous sections describe a method of creating walk-throughs that simulate human paths. In this section we show how the technique could be adapted, in order to obtain the camera freely navigating in 3D space.

Thus, in the previous sections the camera is restricted to be at a constant height from the floor (figure 5.1). The rasterized plane in figure 5.1 represents a set of viewpoints to be processed. If we would like to use all three dimensions, the viewpoint set is to be extended. Figure 5.10 shows extended set of viewpoints. The set of viewpoints is to be the set of plane layers, where the viewpoints fill all the scenic volume. The visibility graph for each layer may be computed in $O\left(\sqrt{n_s} \cdot n_f \cdot n_v\right)$, where n_s is the number of viewpoints in the layer, n_f is the number of faces in the scene and n_v is the number of vertices in the scene. If there are l layers, then the task of the visibility graph computing for all the set of viewpoints may be performed as l independent tasks for each layer. Otherwords, if the set of viewpoints is equidistant, and there are n viewpoints in the set, then the computing time is $O\left(\sqrt[3]{n^2} \cdot n_f \cdot n_v\right)$. Unfortunately, in this case the algorithm is quite time consuming in the visibility computation phase. However, note that it is still better than $O(n \cdot n_v \cdot Z)$, the time bound of the point-to-region computations, where Z is the resolution of the Z-buffer.

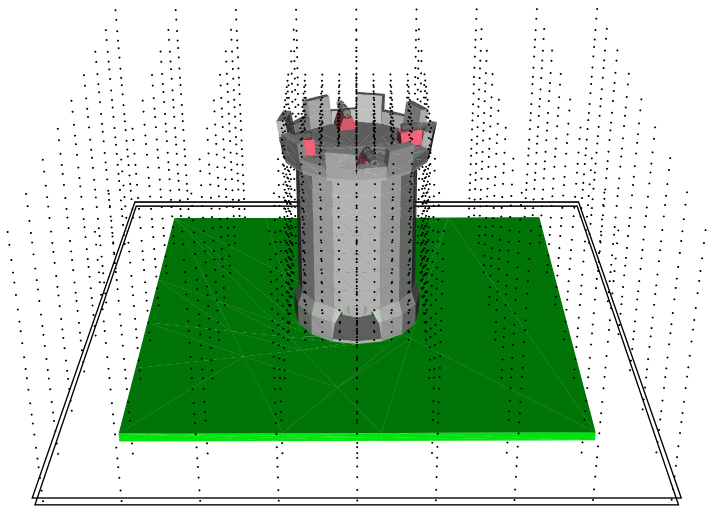


Figure 5.10: The set of viewpoints is the set of plane layers. The viewpoints fill all the scenic volume.

Let us present the set of control views for the scene (figure 5.11). The figure presents 6 control views in the order they were selected by the camera placement algorithm. Figures 5.12(a) and 5.12(b) show the resulting trajectory generated from the above controls views. The resulting trajectory is shown from two points of view: from the side 5.12(a) and from the top 5.12(b). The hidden part of the trajectory is shown in the white dashed line. In order to improve the presentation, we have drawn in white dashed line the treasure chest and the sword (frames 3–5 of figure 5.11). The full sequence of resulting control frames can be seen in appendix B. Refer to figures B.1, B.2 and B.3 to see 18 additional "linking" views added during solving the travelling salesman problem.

Note that neither the treasure chest, nor the sword are visible from the exterior of the scene, but the camera, guided by our method, easily finds them!

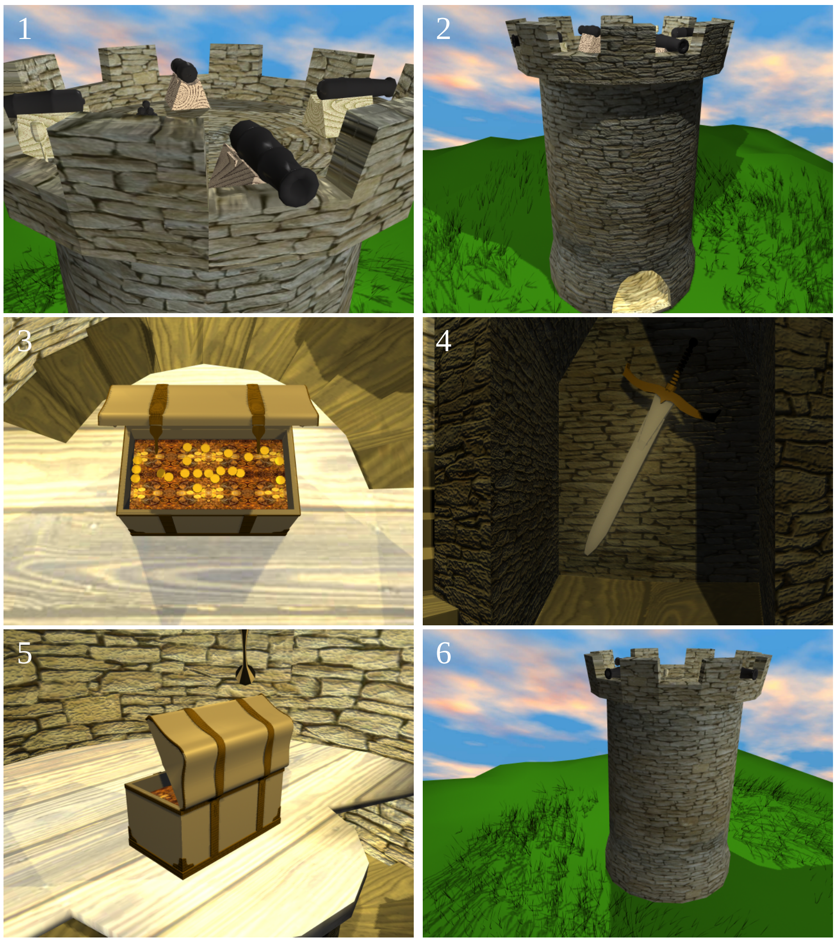
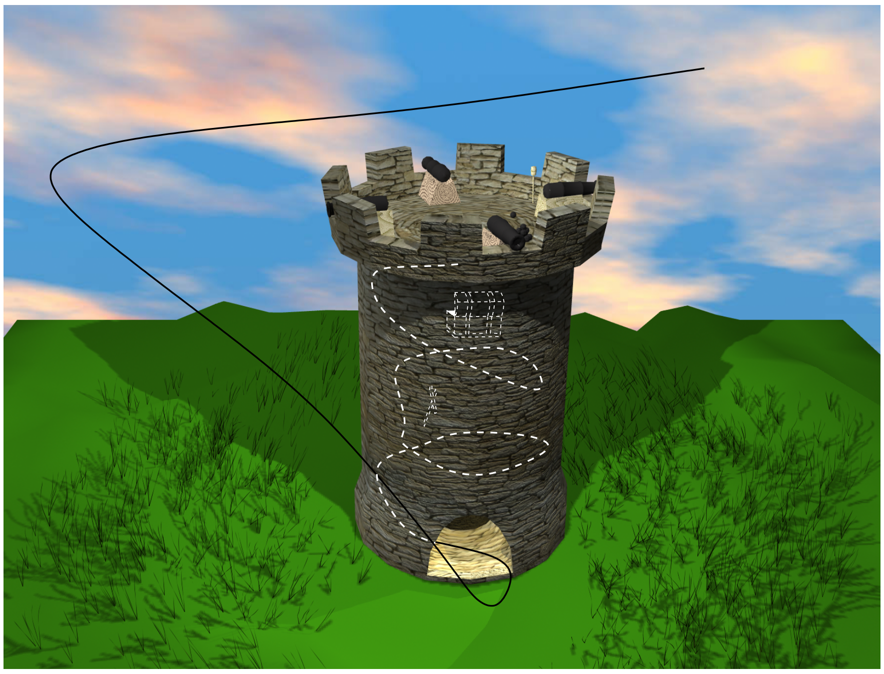
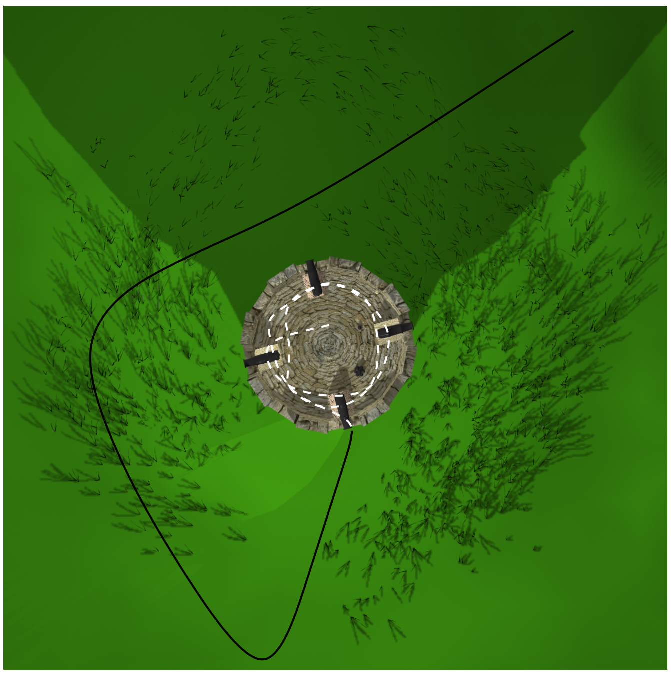


Figure 5.11: The set of 6 control views. The images are presented in the order they were selected by the camera placement algorithm.



(a)



(b)

Figure 5.12: The resulting trajectory is shown from two points of view: from the side **(a)** and from the top **(b)**. The hidden part of the trajectory is shown in the white dashed line. In order to improve the presentation, we have drawn by white dashed line the treasure chest and the sword (refer to frames 3–5 of figure 5.11).

5.5 Textures

Up to this moment we discarded all information other than scene geometry. However, textures may bring a lot of information to the user. For example, an exibition hall may be modelled as a simple cube with paintings applied as textures. In such a case the above methods will fail in exploration. Fortunately, there are simple ways to take into account textures. The general idea is to transform the information stored in *images* to *geometry* that the above techniques can handle.

Figure 5.13 shows an example. A very simple wall in a street passway, textured by graffiti, can be transformed to a mesh without textures. The domain of displacement mapping is well studied and there are displacement methods supported by majority of GPUs.

Thus, having transformed images into meshes, the methods of scene exploration we propose show better results. Figure 5.14 shows a trajectory in a passway. While trajectory 5.14(a) skips the graffiti, trajectory 5.14(b) does not forget to show it.

However, some problems appear in the transformations. For example, a texture of asphalt will produce tons of small triangles. First of all, it increases scene complexity, and, on the other hand, the produced mesh will have high curvature values. One solution is to use mesh saliency measure [LVJ05] instead of mean curvature in viewpoint quality estimations. Since saliency negates repeated textures, the method based on saliency will select good viewpoints better than the method based on mean curvature.

However, the problem of complexity is more important than the problem of repeated textures. The solution is to apply filters to textures before displacement mapping. First of all, it is neccessary to apply filters to eliminate noise and then edge detection filters help to reduce size of produced meshes.

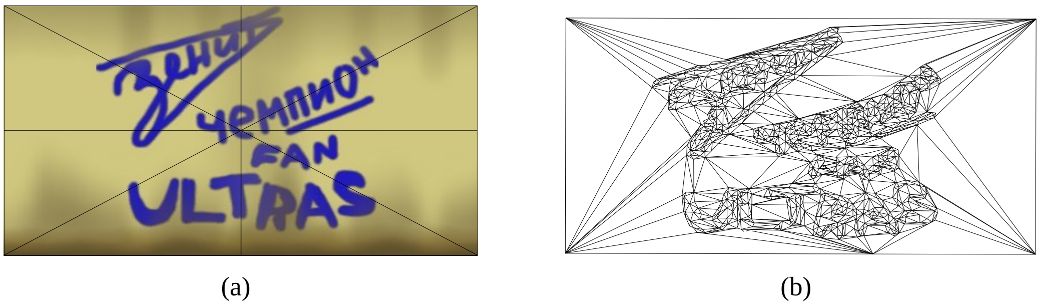


Figure 5.13: Transformation of textures into geometry. **(a)** The original wall contains 8 triangles and 1 texture. **(b)** The transformed wall contains 1504 triangles and no textures.

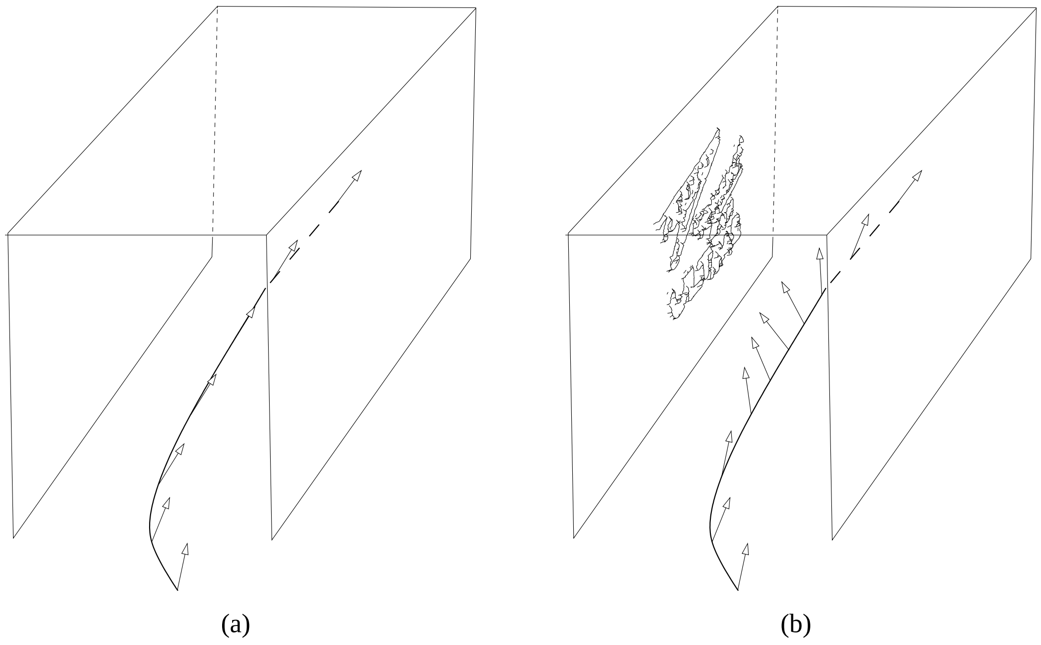


Figure 5.14: Two different trajectories for the same passway. Small arrows indicate camera directions. **(a)** Only original scene geometry is taken into account. **(b)** After transforming textures into geometry.

5.6 Speed-up

Unfortunately, the proposed techniques of local scene exploration may be time consuming for complex scenes. The visibility graph computing phase is the most expensive task. How could we speed up the calculations?

The most logical answer is to simplify the scene. The domain of scene simplification is well studied. Refer to figure 5.15 for an example. The figure shows a glove model and its simplification. As we can see, the simplified version contains about 50% of its original size. Thus, the visibility computation routine will require about 4 times less memory and time. Our tests show that the viewpoint quality estimation routine produces equal results for models and their simplified versions.

However, if we are afraid to loose the accuracy, then hybrid computations may be used. Remember that we use the visibility of vertices to compute viewpoint qualities. In order to get cheaper, but still accurate results, one could take the original set of vertices and the set of faces from the simplification. Refer to figure 5.15 for an example. One can take the set of vertices from the original version of the glove (figure 5.15(a)). The set of faces, or the set of occluders is to be taken from the simplification (figure 5.15(b)). Since the topology almost does not change in the simplification process, the shape of the occluders does not change neither. So, the accuracy of the visibility computations will not be affected by the simplification process.

The next point to optimize is the shape of the viewpoint set. The most simple

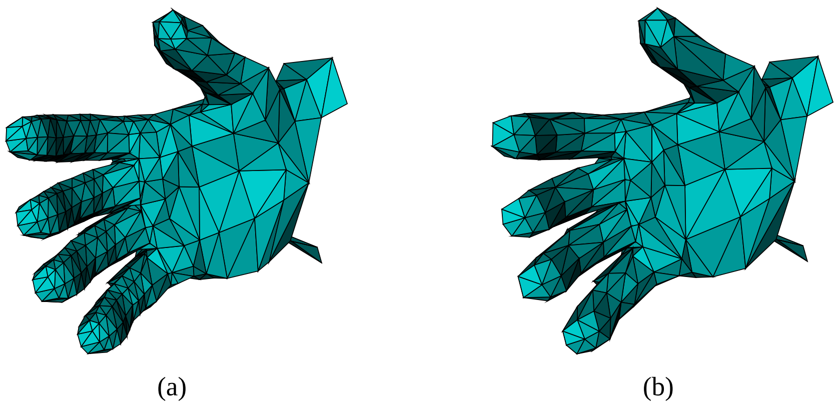


Figure 5.15: A glove and its simplification. **(a)** The original contains 478 vertices and 943 faces. **(b)** The simplified version 260 vertices and 503 faces.

way is to make it equidistant, when all viewpoints form a regular grid. However, it forces to make excessive calculations. Refer to figure 5.9. The viewpoints are placed uniformly, but it is useless to make the grid so dense in the upper left corner, since it is empty.

Often there is given some scene division into a set of objects. Thus, the solution is to start with some density of the grid and to make the grid more dense around objects of the scene. Figure 5.10 illustrates the technique. It is easy to see that the viewpoint set is not crowded in the "street", but becomes much more dense in the bounding box of the tower.

Fortunately, the visibility computations phase is to be performed only once, and then the visibility graph is to be used in multiple explorations without recomputing.

5.7 Conclusion

In this chapter a non-incremental method of local scene exploration is demonstrated. It allows a camera to navigate inside a model until most of interesting reachable places are visited. In distinction from [VS03, Dor01], it guarantees that all interesting places in a scene will be observed. Results produced by the method satisfy the aesthetic criteria defined in section 4.4.1. The techniques, presented in this chapter, may be easily generalized in order to be used in global exploration too.

Chapter 6

Semantic distance

6.1 Introduction

As we have mentioned above, Computer Graphics suffers from weak representation of 3D data. It would be very helpful if a semantic 3D model could be used instead of structureless collection of polygons. Let us consider an example, refer to figure 6.1.

Figure 6.1(a) shows a scene with two churches. There are three manually fixed viewpoints. The dashed line is the trajectory, obtained by the above method of indoor exploration. The exploration is started at the viewpoint 1. It is easy to see that the camera turns aside from the big church in the viewpoint 2 and then reverts back again. The trajectory is very short and smooth, but the exploration is confusing for the user.

Figure 6.1(b) shows another trajectory, where the exploration starts in the viewpoint 1, passes via 3 and terminates in the point 2. The camera turns aside from the big church only when its exploration is complete. Despite the increased length of the trajectory, the exploration is more clear.

Thus, we postulate that the above method of indoor exploration could be improved by regrouping the control frames in dependence on *relativeness* of information they show. In other words, if two frames show the same object, they are consequent.

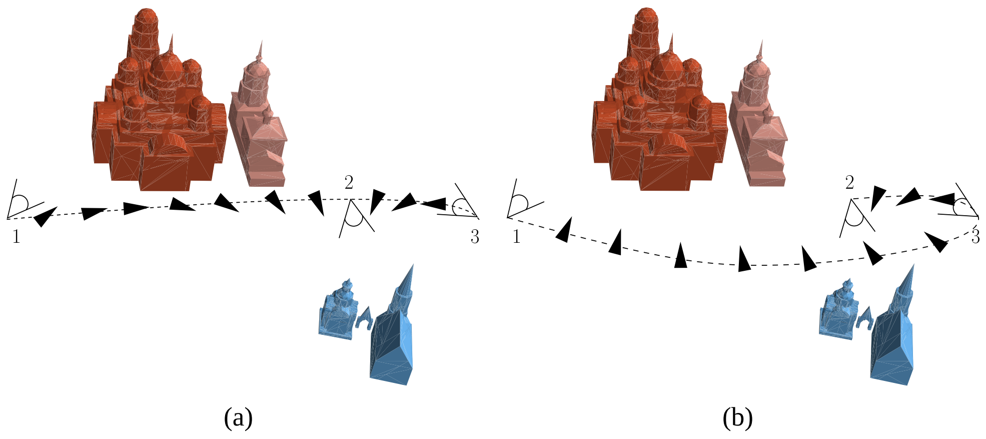


Figure 6.1: Two different trajectories. **(a)** During traversal from the point 1 to 3 the camera turns aside from the big church and then reverts back again. **(b)** The camera turns aside from the big church only when its exploration is complete.

6.2 Excursus

In the following sections we shall use some facts of network flows, so let us briefly remind the notations.

Definition 6.2.1. *Given a connected undirected graph $G = (V, E)$ and a pair of nodes $s, t \in V$, called respectively* source *and* sink, *let $c : E \to \mathbb{R}_+$ be the capacity of the edges in G. A* flow *in G is a function $f : VxV \to \mathbb{R}_+$ such that:*

$$\forall u, v \in V, f(u, v) \leq c(\{u, v\}) \tag{6.1}$$

$$\forall u, v \in V f(u, v) = -f(u, v) \tag{6.2}$$

$$\forall u \in V \setminus \{s, t\}, \sum_{v \in V} f(u, v) = 0 \tag{6.3}$$

Definition 6.2.2. *Given a connected undirected graph $G = (V, E)$. Let f be a flow in G. The* value *of the flow f is defined as:*

$$|f| = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t)$$

Definition 6.2.3. *Given a connected undirected graph $G = (V, E)$ and considering a pair of nodes $u, v \in V$. Let f be a flow in G. The* residual *capacity of the pair*

(u, v)*, or* $c_f(u, v)$ *is given by:*

$$c_f(u, v) = c(u, v) - f(u, v)$$

Definition 6.2.4. *Given a connected undirected graph* $G = (V, E)$*. Let* f *be a flow in* G*. The* residual network *of* G *induced by* f *is digraph* $G_f = (V, E_f)$*, where:*

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

The residual network is always directed, either for directed or undirected graphs.

Problem 6.2.1. *Given a connected undirected graph* $G = (V, E)$*, where* V *is a set of nodes and* E *is a set of edges between those nodes (*$E \subseteq \{\{u, v\} : u, v \in V\}$*), a capacity* $c : E \longrightarrow \mathbb{R}_+$ *and two nodes* s*,* t*, find a maximum flow (a flow whose value is maximal) from source* s *to sink* t *in* G*. Let* f *be this flow.*

Definition 6.2.5. *Given a connected graph* $G = (V, E)$*, directed or undirected, and a pair of nodes* $s, t \in V$*. A st-cut is a partition of* V *into* S *and* $T = V \setminus S$*, such that* $s \in S$ *and* $t \in T$*. The* capacity *of the cut, represented as* $c(S, T)$*, is given by:*

$$\sum_{u \in S} \sum_{v \in T} c(u, v)$$

Definition 6.2.6. *A* minimum cut *(*mincut*) is a cut whose capacity is minimal.*

Theorem 6.2.1. *The following three conditions are equivalent:*

1. f *is a maximum flow in* G
2. *The residual network* G_f *contains no augmenting paths.*
3. $|f| = c(S, T)$ *for some st-cut.*

Proof sketch: If there is an augmenting path, we can send flow along it, and get a greater flow, hence it cannot be maximal, and vice versa. If there is no augmenting path, divide the graph into S, the nodes reachable from s in the residual network, and T, those not reachable. Then $c(S, T)$ must be 0. If it is not, there is an edge (u, v) with $c(u, v) > 0$. But then v is reachable from s, and cannot be in T.

The classic book on network flows is written by Ford and Fulkerson [FF62]. The authors are introduced the theory of flows in directed networks. Undirected networks case study is available in [KL98].

6.3 Regrouping frames

The relativeness (or semantic distance) can be defined in many ways. For example, Foo et al. in [FGRT92] define a semantic distance in conceptual graphs based on Sowa's definition (refer to [Sow79]). Zhong et al. in [ZZLY02] propose an algorithm to measure the similarity between two conceptual graphs.

In our work we use semantic networks. An example of such a network is given at figure 6.2. The figure shows a semantic network for a small virtual museum,

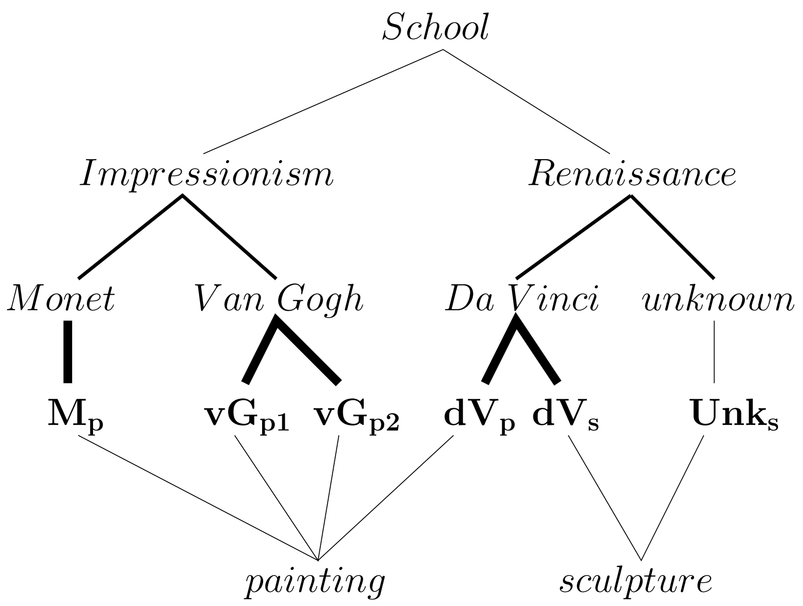


Figure 6.2: The semantic network example for a virtual museum. Importances of relationships are denoted by thickness of lines.

which contains paintings by Van Gogh ($\mathbf{vG_{p1}}$, $\mathbf{vG_{p2}}$), Monet ($\mathbf{M_p}$), Da Vinci ($\mathbf{dV_p}$) and two sculptures by Da Vinci ($\mathbf{dV_s}$) and unknown sculptor ($\mathbf{Unk_s}$). Importances of relationships are denoted by thickness of lines. Thus, the information that $\mathbf{M_p}$ is created by Claude Monet is more important than the information that $\mathbf{M_p}$ is a painting.

In order to measure semantic distance between two objects s and t, we transform the semantic network to an undirected flow network $G = (C, E_C)$. Edges E_C correspond to relationships between concepts C of the semantic network, capacities $b(i, j) > 0 \; \forall (i, j) \in E_C$ are to be set according to the relationship importances. s and t are the source and the sink with unbounded supply and demand, respectively.

Let us denote the capacity of a xy-cut ($x, y \in C$) as $cut(x, y)$ and the capacity of xy-mincut as $\min cut(x, y)$. Then the semantic distance can be defined as follows:

$$D(s,t) = \frac{1}{\min cut(s,t)}. \tag{6.4}$$

For continuity reasons we define $D(x, x) \equiv 0 \; \forall x \in C$.

Figure 6.3 shows the above network with the capacities assotiated.

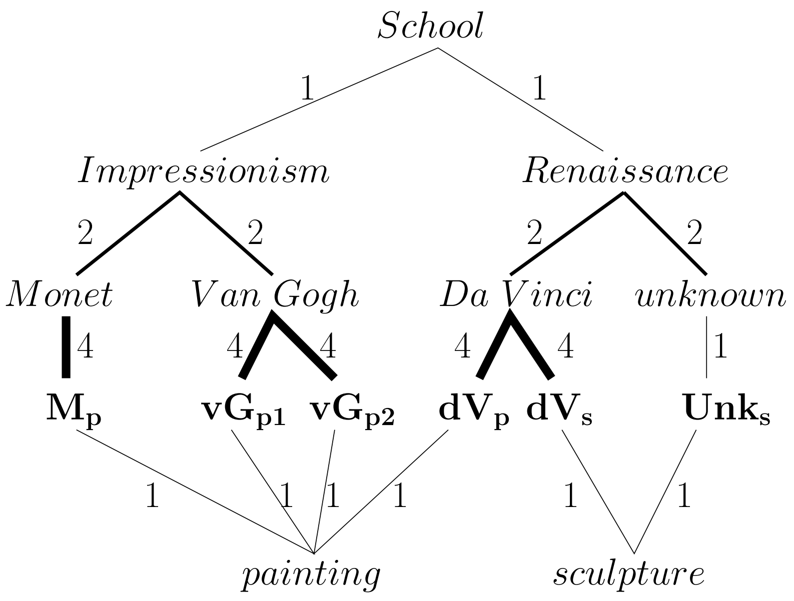


Figure 6.3: The semantic network example for a virtual museum. The numbers indicate the capacities for the network $G = (C, E_C)$.

Lemma 6.3.1. *Equation* (6.4) *defines a metric* $D : C \times C \longrightarrow \mathbb{R}^+$.

Proof. It is easy to see that all the demands are satisfied. Indeed,

1. capacities are non-negative, $D(x, y) \geq 0 \; \forall x, y \in C$,
2. the identity of indiscernibles is satisfied:
   $D(x, y) = 0$ if and only if $x = y$,
3. the network is undirected, the symmetry is satisfied, $D(x, y) = D(y, x)$,
4. the triangle inequality is satisfied also:
   $D(a, b) \leq D(b, c) + D(c, a)$.

Let us suppose that the triangle inequality is not satisfied, then there exists a network and three nodes a, b, c such as $D(a,b) > D(b,c) + D(c,a)$. Then

$$\frac{1}{\min cut(a,b)} > \frac{1}{\min cut(b,c)} + \frac{1}{\min cut(c,a)}. \tag{6.5}$$

The ab-mincut is also bc-cut or ca-cut. Without loss of generality, let us suppose that ab-mincut is bc-cut. Then $\min cut(b,c) \le \min cut(a,b) \Rightarrow \frac{1}{\min cut(b,c)} \ge \frac{1}{\min cut(a,b)}$. Our assumption (refer to inequality (6.5)) implies a contradiction. Therefore, equation (6.4) defines a metric. □

Now, having defined the semantic distance between two objects, we can introduce the semantic distance between two "photos". More strictly, if one "photo" shows a set of objects $A \subseteq C$ and another shows a set of objects $B \subseteq C$, one can define the similarity distance between the sets as follows:

$$D(A,B) = \frac{\sum\limits_{(a_i,b_j)\in A\times B} D(a_i,b_j)}{|A|\cdot|B|}.$$

Lemma 6.3.2. *The distance between sets is also a metric.*

Proof. Obviously, the non-negativity and the identity of indiscernibles and the symmetry are satisfied. Let us suppose that the triangle inequality is not satisfied, then there exists a network and three subsets A, B, C such as $D(A,B) > D(B,C) + D(C,A)$.

$$\frac{\sum\limits_{A\times B} D(a_i,b_j)}{|A|\cdot|B|} > \frac{\sum\limits_{A\times C} D(a_i,c_k)}{|A|\cdot|C|} + \frac{\sum\limits_{C\times B} D(c_k,b_j)}{|C|\cdot|B|}. \tag{6.6}$$

$|A|\cdot|B|\cdot|C| > 0 \Rightarrow$

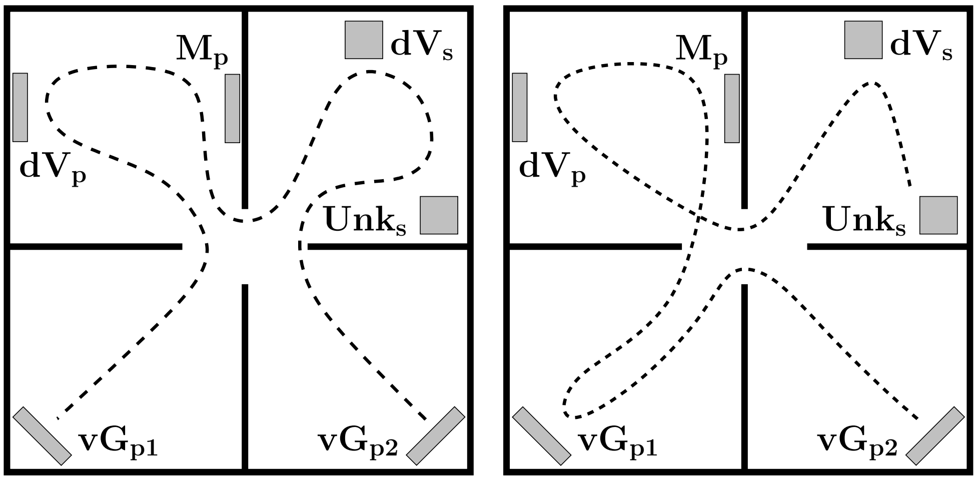
$$|C|\cdot\sum_{A\times B} D(a_i,b_j) > |B|\cdot\sum_{A\times C} D(a_i,c_k) + |A|\cdot\sum_{C\times B} D(c_k,b_j).$$

Let us enumerate all the members of the inequality:

$$\sum_{1\le k\le|C|} \overbrace{\sum_{\substack{1\le i\le|A|\\1\le j\le|B|}} D(a_i,b_j)}^{k\text{–}th\ member} > \sum_{1\le j\le|B|} \overbrace{\sum_{\substack{1\le i\le|A|\\1\le k\le|C|}} D(a_i,c_k)}^{j\text{–}th\ member} + \sum_{1\le i\le|A|} \overbrace{\sum_{\substack{1\le k\le|C|\\1\le j\le|B|}} D(c_k,b_j)}^{i\text{–}th\ member}$$

For each $(i, j, k) \in [1 \dots |A|] \times [1 \dots |B|] \times [1 \dots |C|]$ there is exactly one inequality $D(a_i, b_j) \leq D(a_i, c_k) + D(b_j, c_k)$. But the mapping is also surjective, i.e. all the members are numbered with the triples (i, j, k). Therefore, our assumption (refer to inequality (6.6)) implies a contradiction. The similarity distance between "photos" satisfies the triangle inequality. $\square$

Now let us show how the metric could be used in scene exploration. In the previous chapter we have defined the travelling costs between pairs of cities as the length of the shortest paths between them in the graph $\hat{G}$. Now let us redefine the travelling cost between two "photos" as the length of the shortest path between them *plus the semantic distance between the "photos"*.



(a) (b)

Figure 6.4: The virtual museum exploration trajectories. **(a)** The shortest trajectory. **(b)** The shortest trajectory with respect to the semantic network shown at figure 6.2.

Figure 6.4(a) presents the shortest exploration trajectory for the small virtual museum we presented above. Refer to figure 6.2 for the semantic network. The exploration starts from the Van Gogh painting ($\mathbf{vG_{p1}}$), then comes Da Vinci painting $\mathbf{dV_p}$, Monet ($\mathbf{M_p}$) etc. The exploration terminates after visiting the second painting of Van Gogh ($\mathbf{vG_{p1}}$).

It is clear that the order is not good. For example, it is better to observe creations of one author consequently. A trajectory, obtained with respect to the semantic network is shown at figure 6.4(b). The camera starts with two paintings by Van Gogh ($\mathbf{vG_{p1}}$, $\mathbf{vG_{p2}}$), continues with Monet ($\mathbf{M_p}$), then shows the painting by Da

Vinci (**dV$_{\mathbf{p}}$**), passes to the sculpture by Da Vinci (**dV$_{\mathbf{s}}$**) and terminates with the sculpture by unknown author (**Unk$_{\mathbf{s}}$**).

It is easy to see that the second trajectory is more logical than the first one. The trajectory does not interrupt exploration of items by the same author and the camera passes to the renaissance items only when all the impressionists paintings are explored.

6.4 Conclusion

In this chapter we have introduced a new measure of similarity between objects. It is useful when some additional knowledge of scene structure could be provided. This measure, so called semantic distance, evaluates relationships in the scene to improve the exploration methods. The technique allows to put together control frames in dependence on *relativeness* of information they show.

Semantic networks promise to be a rich area for further research. We are currently defining similarity measure between objects, but it should be possible to extend the definitions taking into account user preferences. It will also be an interesting exercise to use machine learning techniques to take into account implicit user preferences. Different people have different tastes, and artificial intelligence techniques could help to handle some uncertainties. Probably, it would be possible to create for each user a database of preferences to improve exploration of further scenes.

Chapter 7

Conclusion

Le but du travail entrepris dans cette thèse était la mise en œuvre de nouvelles techniques pour l'exploration automatique différée de mondes virtuels. L'objectif de l'exploration différée est d'analyser la scène afin de permettre de contrôler la caméra virtuelle (c'est-à-dire donner des bons points de vue, créer des trajectoires etc) en temps réel après l'analyse. Au moment de la rédaction de ce mémoire, il est possible de conclure que cet objectif a été rempli de manière significative.

7.1 Améliorations apportées

Dans le cadre de ce travail nous avons apporté les améliorations suivantes.

- (Chapitre 3) D'abord, nous avons présenté une fonction de *bas niveau* pour évaluer la qualité d'un point de vue. La fonction ne dépend pas des changements d'une scène qui garde sa topologie originale. Elle est également extensible dans la classe des surfaces continues telles que les NURBS. Cette méthode d'évaluation d'un bon point de vue apporte une amélioration considérable par rapport aux techniques existantes et permet de disposer d'un critère beaucoup plus précis de la qualité de la position de la caméra.
- Ensuite, nous avons introduit une heuristique de *haut niveau* qui permet d'évaluer la qualité d'un point de vue en fonction de *la familiarité* de l'utilisateur avec les objets d'une scène et de *la fonctionnalité* des objets. Par exemple, il est évident que dans un musée virtuel les différents objets ne devraient

pas avoir tous la même importance. Les objets d'art sont sensiblement plus importants que les murs, les chaises etc. Si nous disposons d'une bonne division d'un modèle virtuel en un ensemble d'objets, nous obtenons une bonne heuristique pour le contrôle de la caméra.

- (Chapitre 4) L'amélioration suivante concerne l'exploration externe des mondes virtuels. L'exploration locale peut être utile, et même nécessaire dans certains cas, mais seulement l'exploration globale pourrait donner à l'utilisateur une connaissance générale sur une scène. Ainsi nous avons présenté une méthode très rapide de recherche des bons points de vue. Puisque la fonction de la qualité est basée sur la courbure intrinsèque de surface, le problème de la visibilité est réduit à un problème de visibilité entre points. Ceci nous permet de réduire sensiblement le temps de calculs. Bien que notre objectif de départ était l'exploration différée de mondes virtuels, les résultats que nous obtenons dans le calcul de la trajectoire de la caméra sont si rapides que les techniques proposées peuvent être utilisées pour une exploration directe avec la majorité des scènes.

- Nous avons introduit deux techniques d'exploration *externe* de mondes virtuels. En créant des “films”, ces techniques donnent une bonne compréhension d'un objet simple ou une compréhension générale d'une scène complexe. La première technique est une technique d'exploration incrémentale. La caméra part d'un bon point de vue, choisi grâce à la méthode citée ci-dessus. À chaque étape, la caméra est attirée par des parties encore inconnues de la scène. La direction de déplacement est déterminée par la superposition des forces d'attraction. La deuxième technique est une technique d"exploration non-incrémentale. Elle prends l'ensemble de bons points de vue et les relie par une trajectoire en fonction de la distance entre eux et de la qualité de chaque segment.

- (Chapitre 5) Nous avons également présenté de nouvelles techniques pour l'exploration *locale*. Ces techniques permettent à la caméra de se diriger à l'intérieur (ou bien à l'extérieur) d'une scène jusqu' à ce que la plupart de ses parties intéressantes soit visitée. À la différence d'autres méthodes déjà existantes (par exemple, [VS03, Dor01]), elles garantissent qu'on observera tous les endroits intéressants d'une scène. En utilisant l'heuristique de haut niveau, citée plus haut, pour estimer la qualité d'un point de vue, la technique d'exploration locale proposée donne des résultats très précis.

- (Chapitre 6) Une nouvelle mesure de similitude entre les objets est également présentée dans ce mémoire. Elle est utile quand une connaissance additionnelle de la structure de la scène pourrait être fournie. Cette mesure, dite distance sémantique, évalue des relations dans la scène pour améliorer la méthode d'exploration. La technique permet de regrouper des vues en fonction de la similarité d'information qu'ils montrent.

Les résultats obtenus permettent de répondre aux objectifs fixés au départ. Ils permettent même d'aller au delà dans la mesure où les techniques proposées, de par leur efficacité, peuvent être utilisées non seulement dans le cadre d'une exploration différée mais aussi dans le cadre d'une exploration directe, pour des mondes virtuels relativement complexes.

7.2 Perspectives

Les résultats obtenus sont prometteurs mais peuvent être complétés par d'autres méthodes. Il reste par conséquent plusieurs axes de recherche à poursuivre dans ce domaine. Les axes peuvent être distingués en trois groupes de problématiques :

1. Planification et création d'une exploration ;
2. Évaluation de la qualité d'une exploration ;
3. Optimisation des calculs.

Discutons d'abord des possibilités dans le premier domaine. Les réseaux sémantiques semblent être un secteur riche permettant d'améliorer la qualité des explorations. Nous définissons actuellement la mesure de similitude entre les objets, mais il devrait être possible d'étendre les définitions en tenant compte des préférences de l'utilisateur. Il serait possible d'envisager une notion de qualité d'un point de vue liée aux desideratas de l'utilisateur, qui pourrait affecter une valeur spécifique à certains objets qu'il souhaiterait mettre en valeur. Si l'utilisateur préférait, par exemple, voir des peintures dans un musée virtuel, il serait possible d’introduire directement dans le réseau sémantique des nœuds fictifs. Ces nœuds permettraient de regrouper toutes les peintures afin d’obtenir des relations additionnelles.

En plus, il serait intéressant d’employer des techniques d’apprentissage pour tenir compte des préférences *implicites* de l’utilisateur. Les personnes différentes ont des goûts différents, et les techniques d’intelligence artificielle pourraient aider à lever un certain nombre d’incertitudes. Dans la pratique, l’utilisateur ne peut pas exprimer toutes ses préférences de manière précise. Par exemple, si l’utilisateur dit que il voudrait s’approcher d’un objet, quelle distance doit-on maintenir entre l’objet et la caméra ? Donc, il est nécessaire d’enregistrer tous les cas quand il est d’accord avec une exploration proposée, et, surtout, quand il n’est pas d’accord. Probablement, il serait possible de créer pour chaque utilisateur une base de données des préférences pour améliorer l’exploration des scènes suivantes. Nous pensons que les techniques d’apprentissage peuvent aider à lever une partie de ces incertitudes.

Qu’inclut le deuxième axe de recherche ? Soit nous implémentons d’autres méthodes déjà existantes, soit nous cherchons de nouvelles approches, mais, de toutes façons, nous devons savoir les comparer. Dans ce travail, nous prenons les méthodes existantes et les comparons *manuellement* avec celles proposées ici. Mais comment peut-on comparer des résultats *automatiquement* ? Y a-t-il des critères objectifs de qualité d’une trajectoire ? Ces questions ouvrent des perspectives de recherche importantes.

Le problème principal est qu’il n’y a pas de critères assez strictes de la qualité de l’exploration. En plus, il faut gérer les incertitudes de goût. Par exemple, si on a deux explorations, où la première est plus longue que la deuxième, mais la deuxième montre moins d’information, laquelle est la meilleure ? Et si l’exploration montre beaucoup d’information, mais fait des changements brusques dans le mouvement de la caméra, est-ce qu’elle est bonne ? Peut-être, pour comparer automatiquement deux explorations, il faudrait faire de la recherche assez profonde dans le domaine de la psychologie et de la perception humaine.

Enfin, il paraît impératif d’optimiser les calculs afin d’obtenir des temps de recherche satisfaisants sur des scènes plus vastes. Les méthodes proposées devraient pallier ce problème avec des rendements corrects.

Les technologies informatiques ne cessant d'évoluer, de nouvelles méthodes devraient pouvoir voir le jour bientôt (calculs parallèles processeur central — carte graphique, calculs partagés sur plusieurs cartes graphiques intégrées au même ordinateur, etc.). Grâce à tout cela, la taille des scènes virtuelles ne cessant d'augmenter, il est intéressant de continuer d'étudier les techniques d'explorations différées afin de garantir à l'utilisateur une connaissance optimale de la scène et ce quels que soient sa taille et son nombre d'objets.

Avec l'importance grandissante que prennent les univers virtuels en informatique, il restera constamment nécessaire d'innover dans les techniques permettant de les modéliser d'une part, mais aussi de les explorer le plus efficacement possible.

Appendix A

“Gravitational” global exploration examples

Further, we give some more examples of exploration trajectories for different scenes.

Figures A.1, A.2 and A.3 give exploration trajectories for the martini glass, the candlestick and the mug from the SGI Open Inventor toolkit. Let us compare it with the trajectories, shown at figures 2.7 and 2.8. It is easy to see, that the new trajectories are better than the old ones. They are more smooth and they are shorter. In the old examples the exploration stops when a certain percentage of the faces (90%-99%) has been visited. In the new explorations we require the visibility 100% of surface. One more moment deserves special attention: all three models have similar shapes, they all are vessels. Thus, the exploration trajectories should be similar. Contrary to the explorations, proposed by Vazquez et al, our trajectories respond to this natural demand.

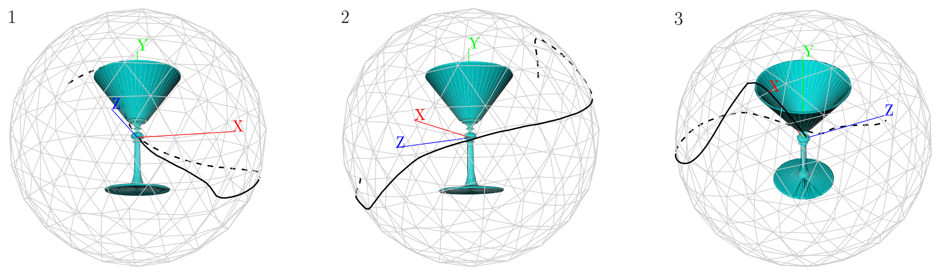


Figure A.1: Exploration trajectory for the SGI martini cup. The images are the snapshots taken consequently from the “movie”, the first one corresponds to the start of the movement (the best viewpoint).

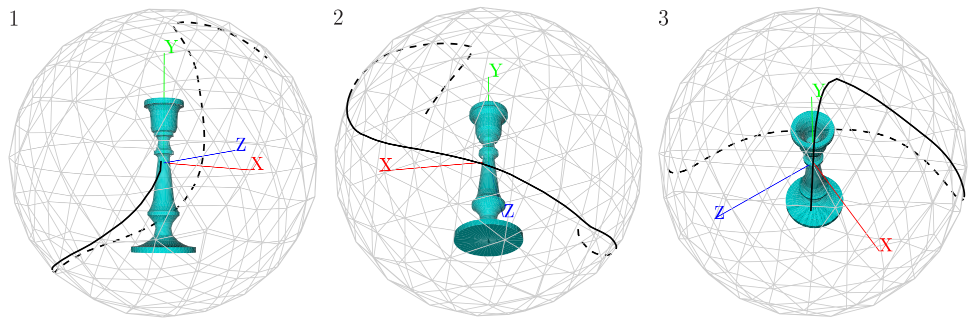


Figure A.2: Exploration trajectory for the SGI candlestick. The images are the snapshots taken consequently from the “movie”, the first one corresponds to the start of the movement (the best viewpoint).

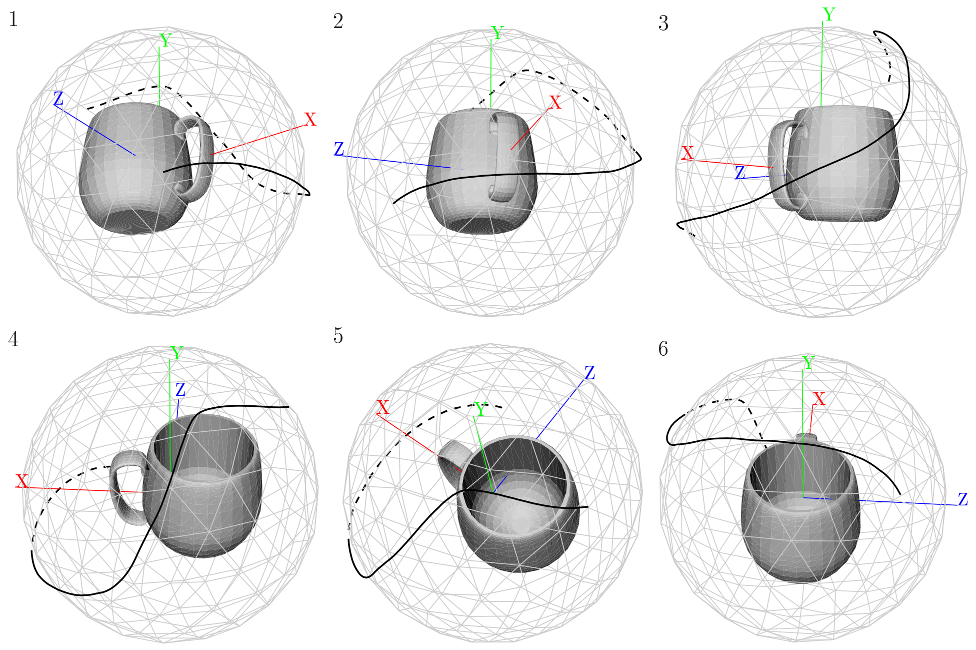


Figure A.3: Exploration trajectory for the SGI mug exploration. The images are the snapshots taken consequently from the “movie”, the first one corresponds to the start of the movement (the best viewpoint).

Figure A.4 shows protracted trajectories made for the Stanford Bunny and for the ionic temple model. Figure A.5 shows the best viewpoint a virtual residential quarter scene. The exploration trajectory for the quarter is given at figure A.6.

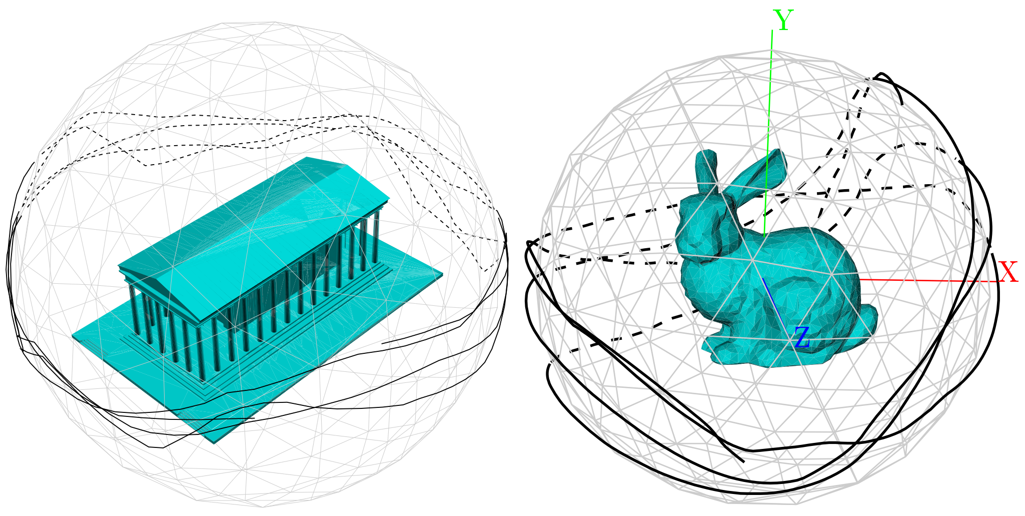


Figure A.4: The protracted exploration trajectories for the ionic temple model and the Stanford bunny.

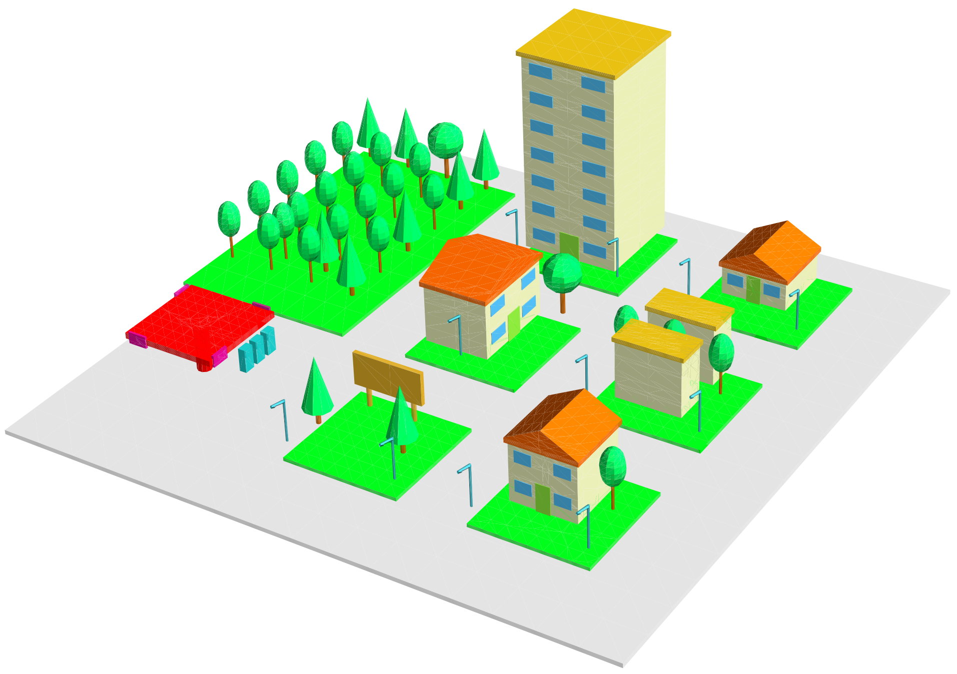


Figure A.5: The first frame from the exploration movie — the best viewpoint.

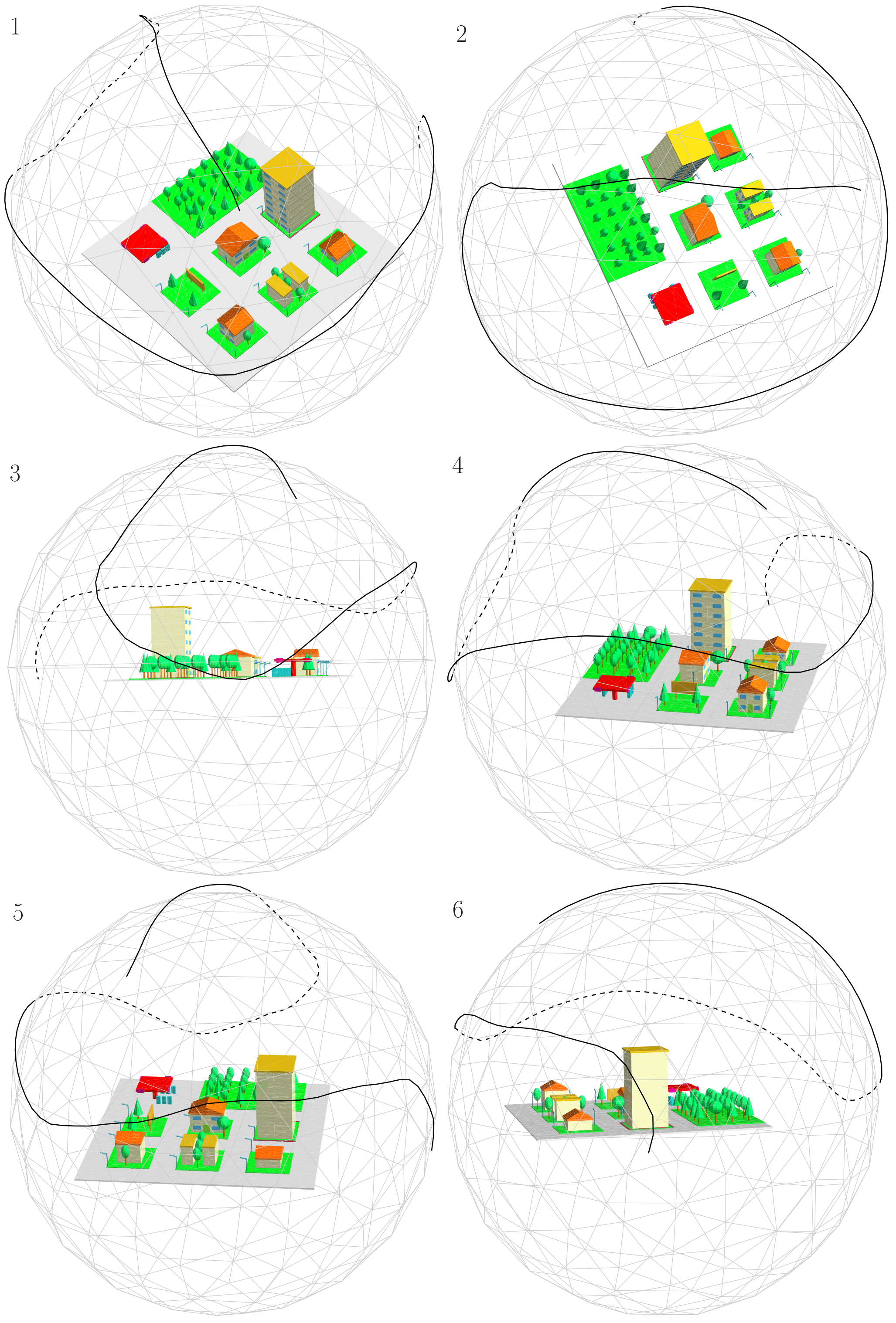


Figure A.6: Residential quarter exploration, refer to figure A.5 for a more detailed picture.

Appendix B

Free fly exploration snapshots

The following figures B.1, B.2 and B.3 show all the sequence of resulting 24 control frames for the example of free navigation in 3D space (refer to section 5.4). The figures show 6 initial control views as well as 18 additional “linking” views added during solving the TSP. Initial views are outlined by black border. Refer to figure 5.11 for the only initial views selected. The number in the upper left corner indicates the number of the frame. Images are to be read from left to right and from top to bottom. Figure 5.12 shows the trajectory line from different views.

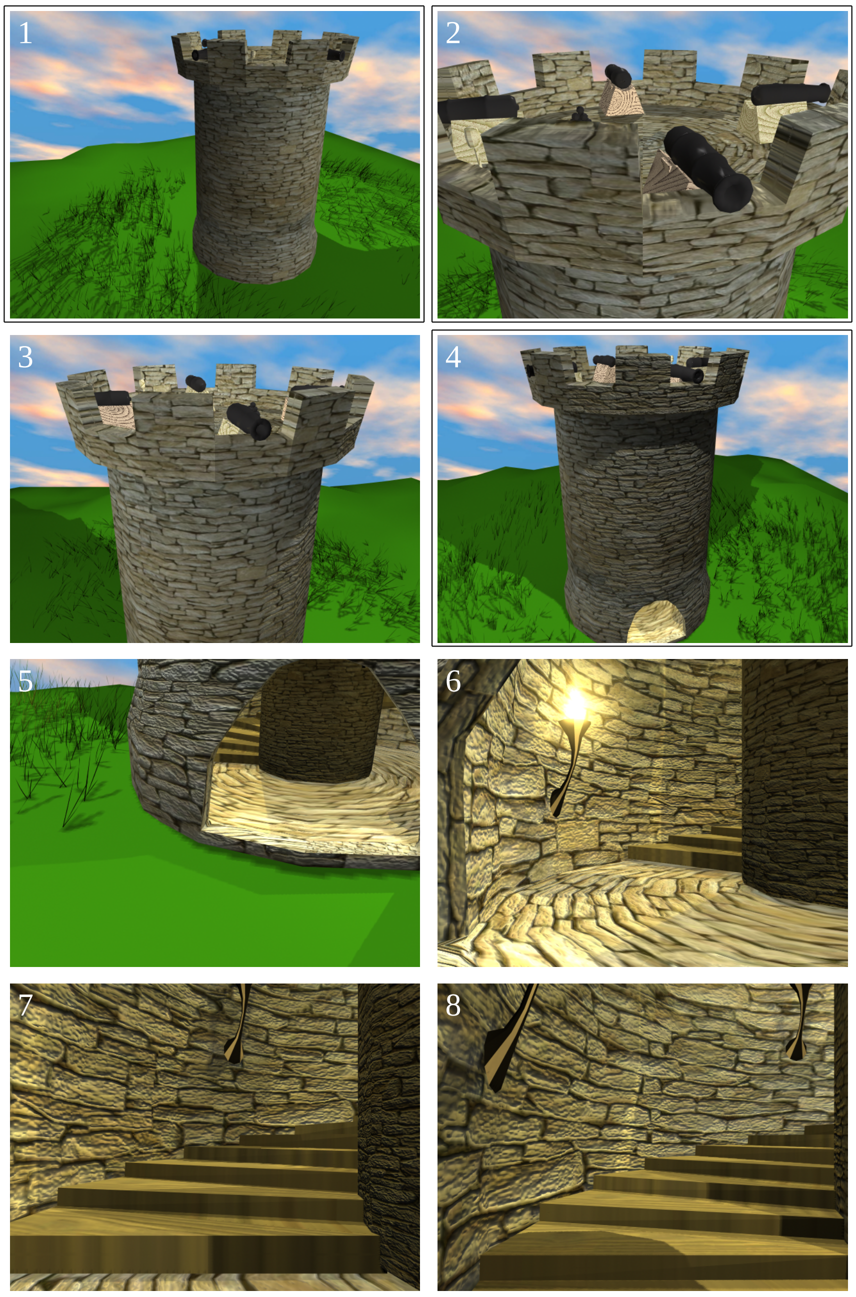


Figure B.1: Navigating freely in 3D space: control views 1–8. Initial views are outlined by black border. The number in the upper left corner indicates the number of the frame. Images are to be read from left to right and from top to bottom.

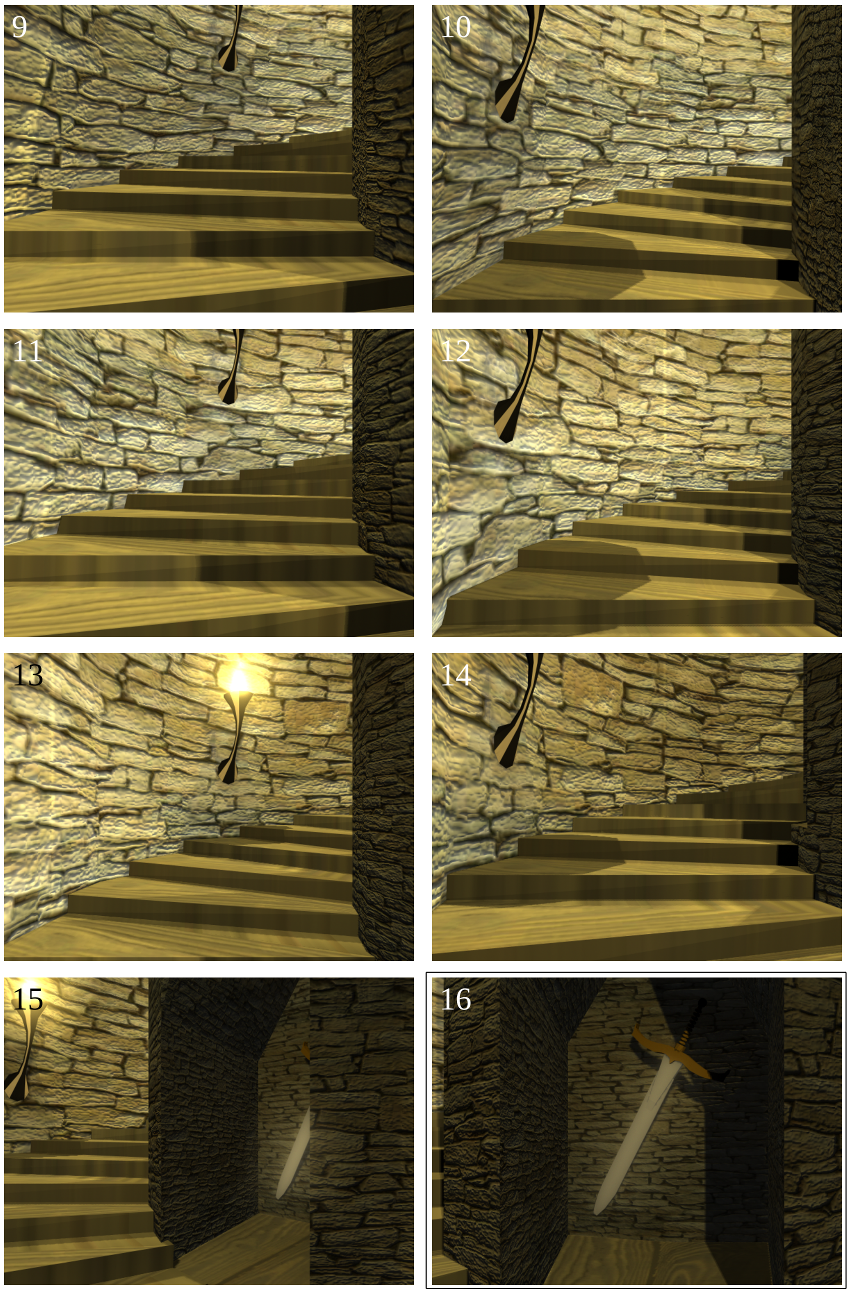


Figure B.2: Navigating freely in 3D space: control views 9–16. Initial views are outlined by black border. The number in the upper left corner indicates the number of the frame. Images are to be read from left to right and from top to bottom.

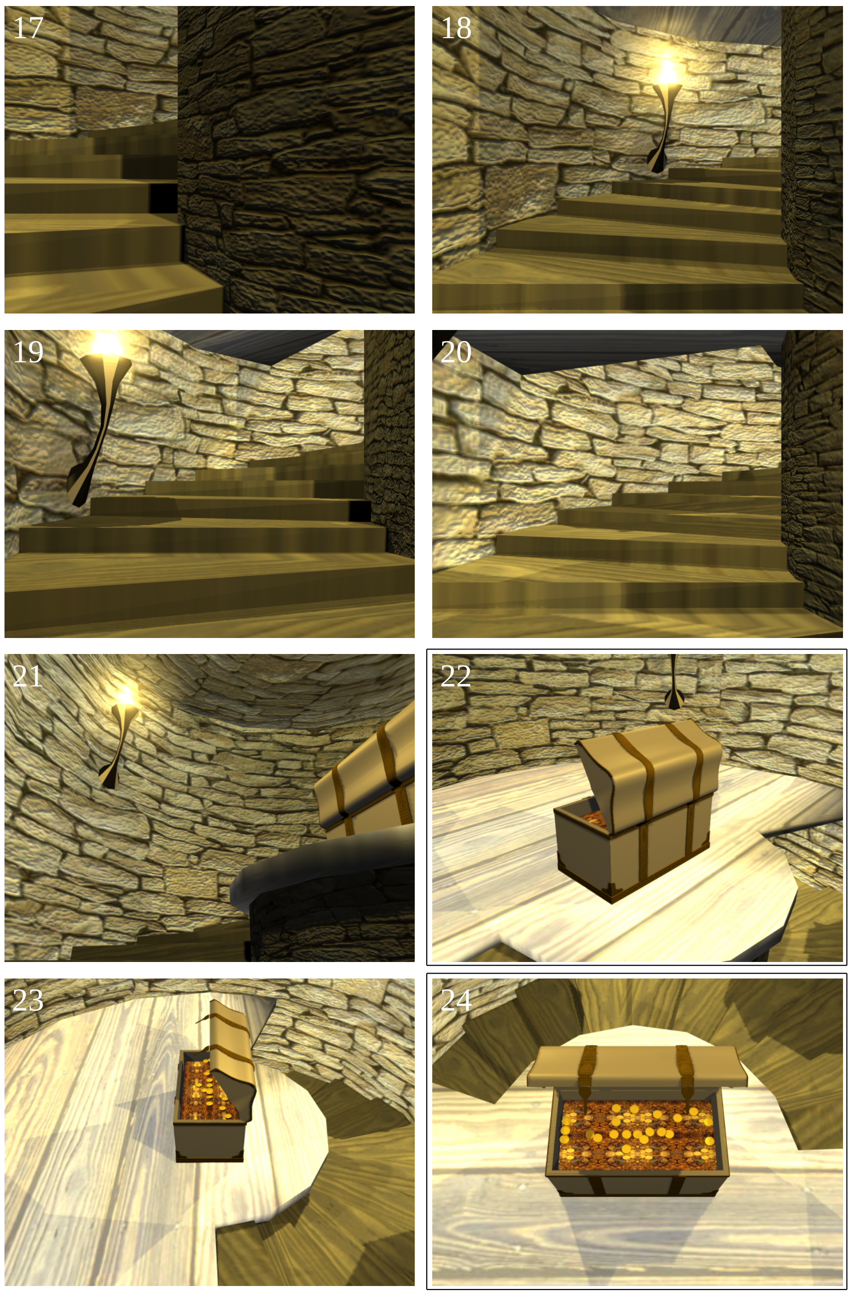


Figure B.3: Navigating freely in 3D space: control views 17–24. Initial views are outlined by black border. The number in the upper left corner indicates the number of the frame. Images are to be read from left to right and from top to bottom.

Appendix C

Memory and CPU usage

Table C.1 presents cost of visibility graph calculations for 4 models we have already met. The times are given for 4 viewpoint sets of different sizes: 400, 1600, 6400 and 25600 viewpoints. Figure C.1 shows a graph version of the table. It is remarkable that 64-times increased viewpoint set requires only doubled processing time.

The results are obtained using inexpensive laptop *Toshiba Satellite A60*, the configuration is: *CPU Intel Celeron D 335 (2.8 GHz), 190Mb RAM*. Note that no hardware acceleration was used, and the implementation of the algorithm can be improved. For example, we use floating point calculations, while it is possible to use fixed point or even integer calculations.

Scene	# Faces	# Vertices	Time (sec) for different number of viewpoints			
			# 400	# 1600	# 6400	# 25600
The Utah teapot (figure 4.16)	1012	522	1.32	2.06	2.46	3.37
The suit (figure 4.13)	6452	3215	41.27	48.01	56.64	84.53
The town (figure 5.3)	7083	3895	67.13	87.23	130.11	217.5
The office (figure 4.14)	6674	4724	77.38	97.14	164.28	288.57

Table C.1: Visibility graph computation phase times for different models.

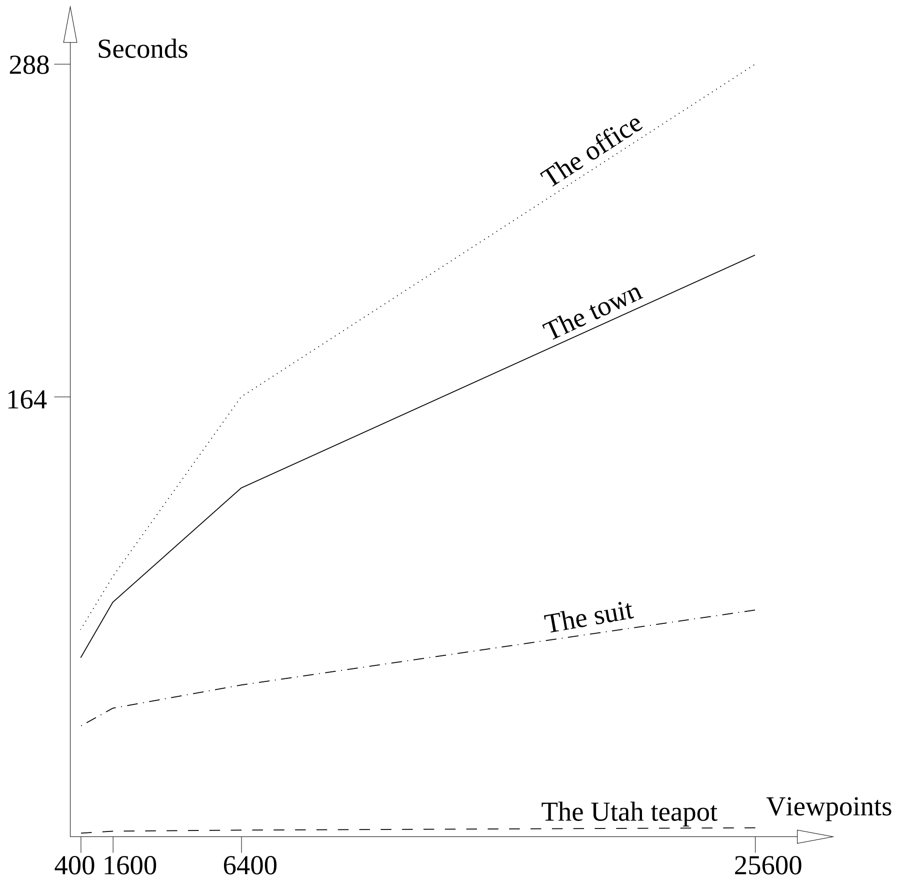


Figure C.1: Visibility graph computation phase times for different models.

Table C.2 shows times of trajectory choosing phase of the algorithms. It is easy to see that the visibility computations are more expensive than the path planning. Fortunately, the visibility computations phase is to be performed only once, and then the visibility graph is to be used in multiple explorations without recomputing. Note that techniques for the town scene trajectory choosing phase is more time consuming, because it requires the Travelling Salesman Problem solving.

Scene	Time (sec) for different number of viewpoints			
	# 400	# 1600	# 6400	# 25600
The Utah teapot	0.11	0.32	1.02	1.37
The suit	2.32	2.86	3.41	5.17
The town	10.88	13.23	24.19	32.21
The office	4.26	6.15	9.68	14.97

Table C.2: Trajectory choosing phase times for different models.

The simpliest way to store visibility information is to keep in memory a matrix of bits. Therefore, its size is the number of viewpoints multiplied by the number of vertices. Table C.3 shows memory usage for the scenes.

Scene	# Vertices	Matrix size (Mb) for different number of viewpoints			
		# 400	# 1600	# 6400	# 25600
The Utah teapot	522	0.025	0.10	0.40	1.59
The suit	3215	0.153	0.61	2.45	9.81
The town	3895	0.186	0.74	2.97	11.88
The office	4724	0.225	0.90	3.60	14.42

Table C.3: Memory usage for different models.

Bibliography

[BDP99] Pierre Barral, Guillaume Dorme, and Dimitri Plemenos. Visual understanding of a scene by automatic movement of a camera. In *GraphiCon'99*, Moscow (Russia), September 1999.

[BDP00a] Pierre Barral, Guillaume Dorme, and Dimitri Plemenos. Intelligent scene exploration with a camera. In *International Conference 3IA'2000*, Limoges (France), May 2000.

[BDP00b] Pierre Barral, Guillaume Dorme, and Dimitri Plemenos. Scene understanding techniques using a virtual camera. In *Eurographics'2000*, Interlaken (Switzerland), August 2000.

[BL97] W. H. Bares and J. C. Lester. Realtime generation of customized 3d animated explanations for knowledge-based learning environments. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI97/IAAI-97)*, pages 347–354. AAAI Press, 1997.

[Bre65] Jack E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[BTB99] Volker Blanz, Michael J. Tarr, and Heinrich H. Bulthoff. What object attributes determine canonical views? *Perception*, 28:575–599, 1999.

[BZRL98] W. H. Bares, L. S. Zettlemoyer, D. W. Rodriguez, and J. C. Lester. Task-sensitive cinematography interfaces for interactive 3d learning environments. In *Proceedings of the 1998 International Conference on Intelligent User Interfaces*, pages 81–88. Tasks and Usage, 1998.

[Cal86] Christopher R. Calladine. Gaussian curvature and shell structures. In *The Mathematics of Surfaces*, pages 179–196, Oxford, England, 1986. Clarendon Press.

[CDST95] Bernard Chazelle, David P. Dobkin, Nadia Shouraboura, and Ayellet Tal. Strategies for polyhedral surface decomposition: an experimental study. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 297–305, New York, NY, USA, 1995. ACM Press.

[Cha91] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.

[Che95] Shenchang Eric Chen. QuickTime VR: an image-based approach to virtual environment navigation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 29–38, New York, NY, USA, 1995. ACM Press.

[Chr76] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

[Col88] Christian Colin. A system for exploring the universe of polyhedral shapes. In *Eurographics'88*, Nice (France), September 1988.

[Dor01] Guillaume Dorme. *Study and implementation of 3D scene understanding techniques.* PhD thesis, University of Limoges (France), June 2001. (In French).

[FCOL99] Shachar Fleishman, Daniel Cohen-Or, and Dani Lischinski. Automatic camera placement for image-based modeling. In *PG '99: Proceedings of the 7th Pacific Conference on Computer Graphics and Applications*, page 12, Washington, DC, USA, 1999. IEEE Computer Society.

[FdABS99] Miquel Feixas, Esteve del Acebo, Philippe Bekaert, and Mateu Sbert. An information theory framework for the analysis of scene complexity. *Comput. Graph. Forum*, 18(3):95–106, 1999.

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[Fei02] Miquel Feixas. *An information theory framework for the study of the complexity of visibility and radiosity in a scene.* PhD thesis, University of Catalonia, 2002.

[FF62] Lester R. Ford and Delbert R. Fulkerson. *Flows in Networks* . Princeton University Press, 1962.

[FGRT92] Norman Foo, Brian J. Garner, Anand Rao, and Eric Tsui. Semantic distance in conceptual graphs. pages 149–154, 1992.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* W. H. Freeman And Company, New York, 1979.

[GRMS01] Bruce Gooch, Erik Reinhard, Chris Moulding, and Peter Shirley. Artistic composition for image creation. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 83–88, London, UK, 2001. Springer-Verlag.

[Ham93] Bernd Hamann. Curvature approximation for triangulated surfaces. pages 139–153, 1993.

[Jau04] Benoit Jaubert. Off-line automatic exploration of virtual worlds (in french). Master's thesis, University of Limoges (France), July 2004.

[JTP06] Benoit Jaubert, Karim Tamine, and Dimitri Plemenos. Techniques for off-line scene exploration using a virtual camera. In *International Conference 3IA'06*, Limoges (France, May 2006.

[KK88] Tomihisa Kamada and Satoru Kawai. A simple method for computing general position in displaying three-dimensional objects. *Comput. Vision Graph. Image Process.*, 41(1):43–56, 1988.

[KL98] David R. Karger and Matthew S. Levine. Finding maximum flows in undirected graphs seems easier than bipartite matching. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 69–78, New York, NY, USA, 1998. ACM Press.

[LVJ05] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. *ACM Trans. Graph.*, 24(3):659–666, 2005.

[MC00] Eric Marchand and Nicolas Courty. Image-based virtual camera motion strategies. In S. Fels and P. Poulin, editors, *Graphics Interface Conference, GI'00*, pages 69–76, Montreal, Quebec, May 2000. Morgan Kaufmann.

[NRTT95] Hansrudi Noser, Olivier Renault, Daniel Thalmann, and Nadia Magnenat Thalmann. Navigation for digital actors based on synthetic vision, memory, and learning. *Computers & Graphics*, 19(1):7–19, 1995.

[PB96] Dimitri Plemenos and Madjid Benayada. Intelligent display in scene modelling. new techniques to automatically compute good views. In *GraphiCon'96*, Saint Petersburg (Russia), July 1996.

[PGJT05] Dimitri Plemenos, Jerome Grasset, Benoit Jaubert, and Karim Tamine. Intelligent visibility-based 3D scene processing techniques for computer games. In *GraphiCon'05*, Novosibirsk (Russia), June 2005.

[Ple91] Dimitri Plemenos. A contribution to the study and development of scene modelling, generation and visualisation techniques. *The MultiFormes project., Professorial dissertation*, November 1991.

[Ple95] Dimitri Plemenos. Declarative modeling by hierarchical decomposition. the actual state of the MultiFormes project. In *GraphiCon'95*, Saint Petersbourg (Russia), July 1995.

[Ple03] Dimitri Plemenos. Exploring virtual worlds: Current techniques and future issues. In *International Conference GraphiCon'2003*, Moscow (Russia), September 2003.

[PSF04] Dimitri Plemenos, M. Sbert, and M. Feixas. On viewpoint complexity of 3D scenes. In *International Conference GraphiCon'2004*, Moscow (Russia), September 2004.

[SFR$^+$02] Mateu Sbert, Miquel Feixas, Jaume Rigau, Francesc Castro, and Pere Pau Vázquez. Applications of the information theory to computer graphics. In *International Conference 3IA'2002*, Limoges (France), May 2002.

[Sho92] K. Shoemake. Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Proceedings of Graphics Interface'92*, pages 151–156, 1992.

[Sow79] John F. Sowa. Semantics of conceptual graphs. In *Proceedings of the 17th annual meeting on Association for Computational Linguistics*, pages 39–44, Morristown, NJ, USA, 1979. Association for Computational Linguistics.

[SP05] Dmitry Sokolov and Dimitri Plemenos. Viewpoint quality and scene understanding. In Mark Mudge, Nick Ryan, and Roberto Scopigno, editors, *VAST 2005: Eurographics Symposium Proceedings*, pages 67–73, ISTI-CNR Pisa, Italy, November 2005. Eurographics Association.

[SPFG05] Mateu Sbert, Dimitri Plemenos, Miquel Feixas, and Francisco Gonzalez. Viewpoint quality: Measures and applications. In *Computational Aesthetics in Graphics, Visualization and Imaging*, 2005.

[SPT06] Dmitry Sokolov, Dimitri Plemenos, and Karim Tamine. Viewpoint quality and global scene exploration strategies. In *International Conference on Computer Graphics Theory and Applications (GRAPP 2006)*, Setúbal, Portugal, February 2006.

[SR61] Sundaram Seshu and Myril B. Reed. *Linear Graphs and Electrical Networks*. Addison Wesley, London, 1961.

[SS02] Stanislav L. Stoev and Wolfgang Straßer. A case study on automatic camera placement and motion for visualizing historical data. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 545–548, Washington, DC, USA, 2002. IEEE Computer Society.

[Váz03] Pere Pau Vázquez. *On the selection of good views and its application to computer graphics*. PhD thesis, Barcelona (Spain), May 2003.

[VFSH01] Pere Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. Viewpoint selection using viewpoint entropy. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 273–280. Aka GmbH, 2001.

[VFSH02] Pere Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. Image-based modeling using viewpoint entropy. In *Computer Graphics International*, pages 267–279, 2002.

[VFSL02] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Antoni Llobet. Viewpoint entropy: a new tool for obtaining good views of molecules. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, pages 183–188, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[VS03] Pere Pau Vázquez and Mateu Sbert. Automatic indoor scene exploration. In *International Conference 3IA'2003*, Limoges (France), May 2003.

[WW97] Daphna Weinshall and Michael Werman. On view likelihood and stability. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):97–108, 1997.

[YZW03] Ge Yunfang, Pan Zhigeng, and Xu Weiwei. Vision-based path planning in intelligent virtual environment. In *International conference 3IA'03*, pages 25–33, Limoges (France), May 2003.

[ZTS02] Emanoil Zuckerberger, Ayellet Tal, and Shymon Shlafman. Polyhedral surface decomposition with applications. *Computers & Graphics*, 26(5):733–743, 2002.

[ZZLY02] Jiwei Zhong, Haiping Zhu, Jianming Li, and Yong Yu. Conceptual graph matching for semantic search. In *ICCS '02: Proceedings of the 10th International Conference on Conceptual Structures*, pages 92–196, London, UK, 2002. Springer-Verlag.

List of Figures