FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

3D CAPTCHA - Obfuscated Rendering

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Alex Druml
Matrikelnummer 0625181

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Thomas Auzinger, MSc.

Wien, 13.03.2013

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

3D CAPTCHA - Obfuscated Rendering

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Alex Druml
Registration Number 0625181

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Thomas Auzinger, MSc.

Vienna, 13.03.2013

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Alex Druml
Martinstraße 8/9, 1180 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

First of all, I want to thank Michael Wimmer and Thomas Auzinger for giving me the opportunity to write this thesis. Their support and constructive criticism helped me to improve it and to get a different perspective on my work. In addition, I want to thank Matthias Bernhard for the comprehensive support during the design and realization of the user study.

I owe special thanks to my friends Florian, Lukas and Markus for the great years of study and hours of discussions.

Last but not least, I want to thank Simone for the support in so many ways and years, without her, this would not have been possible at all.

Abstract

Motivated by the increasing demand of 3D models for interactive applications, 3D model databases require high quality metadata to provide effective retrieval and exploration possibilities. The generation of such metadata is still an open problem for automated systems, whereas humans are able to process this task with minimal cognitive effort. In this thesis, we present a framework to utilize a human workforce to generate meaningful annotations. Inspired by the general purpose aspect of the reCAPTCHA approach, the concept of outsourcing is achieved by integrating the task into a CAPTCHA challenge, i.e. a challenge for telling humans and automated systems apart. To protect the system against automated attacks, the 3D models are presented in an obfuscated manner such that they are only recognizable for humans.

In this work a new rendering approach is proposed that provides an obfuscated representation of an animated scene containing 3D objects. The basic principle of the obfuscation approach is to capture the salient geometry features of the objects. New methods are introduced to reveal additional features of the object surface texture, allowing the user to provide more specific annotations. A user study was conducted to evaluate the approach presented with regard to the limits of human object recognition abilities. A test procedure is presented that allows to determine the degree of difficulty and, consequently, the definition of a difficulty threshold to implement human perceptual limits. To avoid testing all possible parameter combinations, two perception-related dimensions are introduced in the user evaluation to select relevant technical parameters.

Therefore, a method that is specialized on the obfuscation of animated scenes containing static 3D models is shown, and, in addition, a method to determine the perceptual limits of object recognition in this context is proposed.

Kurzfassung

Aufgrund des steigenden Bedarfs an 3D Modellen für interaktive Anwendungen, benötigen 3D Modelldatenbanken hochqualitative Metadaten um Usern effektive Explorations- und Retrievalfunktionen zu ermöglichen. Das damit verbundene Annotierungsproblem, welches dem Generieren von jenen Metadaten entspricht, stellt für automatische Systeme weiterhin ein offenes Problem dar. Menschen hingegen sind in der Lage diese Aufgabe mit minimalem kognitivem Aufwand zu bewältigen. Im Zuge dieser Arbeit wird ein Framework präsentiert, das diese menschlichen Fähigkeiten zur Annotierung von 3D Modellen nützt. Inspiriert durch den reCAPTCHA Ansatz wird dabei der Annotierungsprozess durch Integration in eine CAPTCHA Challenge ausgelagert um damit gleichzeitig automatische Systeme von Menschen zu unterscheiden. Automatisierte Angriffe auf das System werden verhindert, indem die Objekte in einer verschleierten Form dargestellt werden, sodass sie lediglich für Menschen erkennbar sind.

In dieser Arbeit wird eine neue Renderingmethode vorgestellt, die eine verschleierte Repräsentation einer animierten Szene mit 3D Objekten ermöglicht. Die Grundidee dieses Ansatzes ist die auffälligen Geometriemerkmale der Objekte darzustellen. Diese Idee wird durch lokales Anzeigen der Objektoberfläche erweitert um genauere Annotationen zu ermöglichen. Zur Evaluierung des Ansatzes wurde eine Anwenderstudie durchgeführt um die Grenzen der menschlichen Objektwahrnehmung zu testen. Damit wird gleichzeitig eine Testmethodik vorgestellt, die es erlaubt den Schwierigkeitsgrad der Verschleierung (obfuscation) zu bestimmen und ermöglicht somit auch die Definition eines Grenzwertes in Abhängigkeit der Wahrnehmungsgrenzen. Da eine Erhebung über alle möglichen Kombinationen von technischen Parametern mit den verfügbaren Ressourcen unvereinbar ist, werden relevante Parameter auf Basis von wahrnehmungsbezogene Parameter ausgewählt, die gleichzeitig als Dimensionen für die Evaluierung herangezogen werden.

Somit wird mit dieser Diplomarbeit ein Verfahren vorgestellt, das auf die Verschleierung (obfuscation) von animierten Szenen mit statischen 3D Objekten spezialisiert ist. In diesem Kontext wird eine Evaluierungsmethode zur Bestimmung der Wahrnehmungsgrenzen präsentiert.

Contents

CHAPTER 1

Introduction

1.1 Motivation

The increasing popularity of interactive 3D applications, supported by recently established interaction concepts, is directly coupled with the demand for high quality 3D models. Examples of such applications are map-visualizations (e.g. Google Earth), the gaming segment or archaeological reconstructions of cultural heritage. In the course of the application development process, the creation and design of adequate models is a time-consuming task that requires specialized skills and qualifications. Motivated by these workflow-issues, well known services like Google 3D Warehouse [24] or TurboSquid [59] provide collections containing multiple hundred thousands of 3D models.

Similar to image collections, exploring and searching for specific models in databases with large scale dimensions is enabled by an appropriate retrieval technique. Users usually access the databases by providing a high-level description of the desired objects, formulated in a text-query. To process such queries, additional metadata associated with each model is needed. Automatic extraction of new metadata from object features is a common problem in the field of image and object retrieval; the missing link between the low- and high-level features is known as the semantic gap [44]. Therefore, the textual annotation of models with semantically relevant keywords is in general done in a manual way. Depending on the guidelines of the service, the quality and the number of resulting annotations can vary significantly. The problem of poor annotation quality becomes more evident when performing a practical search for the term *eagle* on Google Warehouse – examples of retrieved objects are shown in Figure 1.1 and sorted according to their ranking in the results.

Motivating humans to participate the process of annotation can be accomplished in different ways. Ahn and Dabbish [1], for example, introduced with the *ESP game* a method where the annotation task is encapsulated by a multiplayer online gaming concept. In contrast to the entertainment based approach, paid solutions are realized using crowdsourcing services like the Amazon Mechanical Turk [34] where the annotation task is broken down into a set of short Human Intelligence Tasks which can then be performed by the crowd, consisting of several hun-

Figure 1.1: Ordered exemplary results for the term *eagle* on Google 3D Warehouse, where the first occurrence of the animal was on page 3. (figures from [24])

dred thousand workers. With the reCAPTCHA approach Ahn et al. [4] introduced a method to integrate human-computation-tasks into a web-security processing context. A common problem for online services such as free email providers, public user polls, messages boards etc. is their abuse through automated processes and bot systems. To cope with this problem, a puzzle, which has to be solved by the user, is included in the process (e.g. registration). Those puzzles are designed in a way such that humans can solve the given problem with minimal effort, while it remains a hard or unsolvable problem for computers. This concept was first described by Ahn et al. [2] as CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) and allows the service to consequently validate a user as human. The reCAPTCHA concept utilizes the effort of solving CAPTCHAs for a useful purpose, more precisely, the user solves the CAPTCHA by transcribing words into written form that can not be recognized by state-of-the-art Optical Character Recognition (OCR) algorithms.

1.2 Problem Statement and Aim of this Work

Inspired by the general purpose aspect of the reCAPTCHA approach, this work aims at an analogous integration of the 3D model annotation task in terms of a CAPTCHA challenge. reCAPTCHA uses words scanned from books which can not be interpreted reliably by current OCR algorithms as a transcription problem for the CAPTCHA challenge. In detail, it uses two words, the control word, which is *known*, and an *unknown* word. If the user transcribes the control word correctly, it is assumed that the unknown word is also transcribed correctly.

In this work the control and transcription principle introduced with the reCAPTCHA approach is adapted from a text-based method to a CAPTCHA challenge defined by the usage of

3D models. Instead of presenting the user two words, which have to be transcribed, two objects from a given model database have to be textually annotated by the user to pass the challenge. Therefore, the formalism of the human-computation-tasks switches from a transcription problem to an annotation problem. Visualizing the 3D objects using standard rendering techniques creates potential security issues that could invalidate the CAPTCHA capabilities of the system. A specific example for an automatic way to solve this kind of challenges could exploit Google Image Search [25] by using rendered images from a challenge as query image. Results from the image search and the corresponding meta information is utilized as solution for the challenge. To protect the system against automated attacks, the idea is to present the 3D models in an obfuscated manner such that they are only recognizable for humans. Besides this aspect, the usage of 3D models introduces an additional dependency on the object orientation, because relevant features for the object recognition could be distributed across the object geometry. As static images do not provide the required flexibility, a successive change of the orientation over time can overcome this issue. Moreover, this kind of animation has a supporting effect regarding security aspects.

From the demand of an obfuscated object visualisation follows the main task of this work, which is the design of a rendering pipeline that uses 3D models from a given database as input and synthesizes an animated sequence of the objects by introducing synthetic manipulations. Based on the approach proposed by Mitra et al. [47], which aims at the obfuscation of 3D models in static images, further extensions are presented to reveal additional object features and to meet challenges regarding security aspects when using animated scenes.

The intended use of our system in a CAPTCHA context implies that the generated scenes should only be recognizable for humans – therefore the obfuscation process directly addresses the abilities of the human perception. Hence, the second goal of this work is to establish a method for determining the link between the degree of obfuscation and the degree of difficulty.

The main research question of this work is how to construct a processing pipeline that obfuscates 3D models in a way such that it is still recognizable for humans while it is unrecognisable for automatic systems. This implies a second question on how to evaluate the presented obfuscation approach regarding the level of difficulty. As concrete computation or analytical models of complex object perception are not available, a user based evaluation has to be conducted to determine the performance and limitation of human object recognition in our context.

1.3 Contributions

In the following the main contributions of this thesis are outlined, the scope of these contributions include a wide range of topics, starting from computer vision to cognitive sciences:

- **Object obfuscation rendering pipeline**
  An algorithm is proposed that is specialized on the generation of an animated and obfuscated visualization of 3D scenes containing static 3D models. The generated visualizations of this pipeline contain the obfuscated object and, in addition, a synthesized background. To prevent automatic detection of the obfuscated object, we present a method to guarantee a certain degree of temporal coherence in both the object and the background

region of the animation. By including additional object features, future extensions are outlined that enable more detailed user annotations or annotations on a higher semantic level.

- **Perceptually related parameters**
  In general, an obfuscation can be achieved in a variety of ways due to the high number of degrees of freedoms of this problem. Our presented method represents a subset of the possible design space, which is characterized by a number of technical parameters. Due to the missing link between the technical parameters and their effect from a perceptual point of view, the number of parameters are abstracted and represented by a set of perceptually motivated parameters.

- **Perceptual user study**
  A novel method for testing the interaction between technical parameters of the rendering prototype is proposed by measuring the degree of difficulty at the same time. In addition to that, a new approach for testing the usability across different settings is presented.

Beside these contributions, the rendering technique presented in this work could be incorporated with the crowdsourcing game from the concurrently written thesis of Felberbauer [19]. The main idea of the game presented is to generate annotations in a competitive gameplay, where new keywords are generated and simultaneously rated by the users. The obfuscated visualization could be used in the existing game concept to increase the difficulty of the standard 3D model annotation levels.

1.4 Structure

This thesis is structured into the following six main chapters:

1. **Introduction**
   With the first chapter the main motivation, the problem statement and aim of this work were presented. Moreover the contributions of the thesis were outlined and it contains the current section which provides a structural overview of the thesis.

2. **Background & Related Work**
   The basic principle of the thesis is related to multiple and different research fields, from computer vision to topics focused on the basics of human visual perception. In this chapter the fundamentals for the obfuscation method described in the next chapter are explained and corresponding state-of-the-art approaches are presented.

3. **Overview**
   Starting with a brief overview on the big picture of the CAPTCHA system, the specific role of the rendering engine is pointed out.

4. **Obfuscated Rendering**
   In this chapter, the main idea of the obfuscated rendering approach is outlined, followed by detailed explanations of the several stages of the proposed obfuscation pipeline.

5. **Results**
   The main topic of this chapter is the evaluation of the proposed method in terms of a user study. The conception, realization and results of the user based evaluation approach developed in this thesis are discussed. An additional test using the state-of-the-art optical flow approach is performed and corresponding findings are presented.

6. **Conclusion and Future Work**
   The final chapter briefly summarizes the highlights of this thesis and lists the most important findings as well as some implications for future work.

CHAPTER 2

Background & Related Work

As this work is strongly related to human perceptual aspects, the first part of this chapter explains the fundamentals and principles of the human visual perception. Next, details about CAPTCHA systems and the state-of-the-art are presented. Due to the fact that the main idea and the work itself includes concepts of crowdsourcing, this topic represents the last part of this chapter.

2.1 Visual Perception

Bruce et al. [5] define perception as the ability to detect structures and events of the environment based on information derived from energies. These energies are produced by the surroundings and can be perceived as long as humans are sensitive enough to them.

2.1.1 The Visual System - a Physiological Overview

The type of energy can be radically different: possibilities range from kinetic energy in form of air pressure waves, impact pressure or thermal energy over chemical energy for olfaction to electromagnetic radiation. Especially the energy in form of light allows humans to perceive their environment through vision. From a more physical perspective, light is an electromagnetic radiation with a specific wavelength range of approximately 400 to 700 nm [5, 30]. In terms of the practically relevant spectrum which is defined from radio waves with wavelengths of 10^{11} nm to X-rays with a wavelength of 10^{-1} nm, visible light represents only a small part and is located between the near infra-red and the ultraviolet parts. For the transportation of energy, the model of photons is used. Photons are discrete particles consisting of a quantum of energy and moves at the speed of light in empty space. [5, 30]

The process of human vision starts where a photon is absorbed and converted into electrical signals by the eye [30]. Figure 2.1a illustrates the most important physiological parts of the human eye. The light traverses the optical apparatus (defined by cornea, intra-ocular fluid, lens and the vitreous body) and the refractive lens system and finally meets the photosensitive area of the eye, the retina [12, 22]. Hence, the eye shows a high similarity to a simplified camera model

with a lens, a changeable aperture (the iris) and a sensor (the retina). By adjusting the curvature of the lens using the ciliary muscles, its refractivity is changed, which is used to focus the image on the retina.

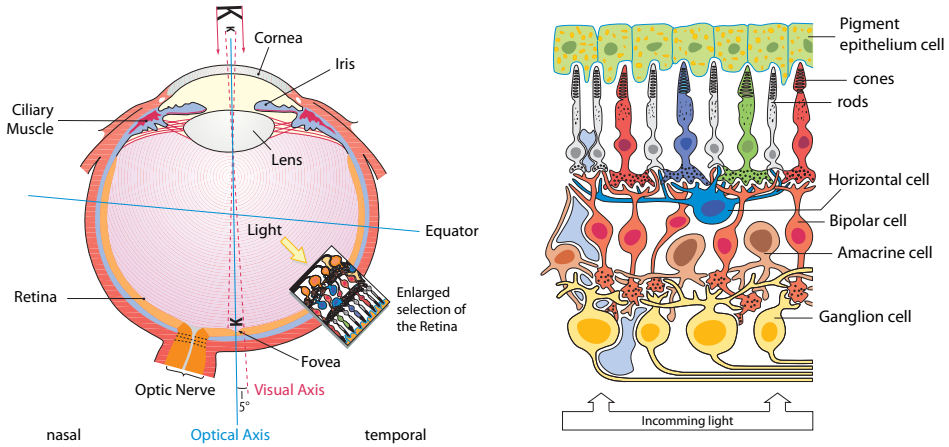


Figure 2.1: Illustration of the most important parts of the eye (a) and the schematic construction of the retinal cell layers (b). (figures adapted from [40])

Different layers of nerve cells (see Figure 2.1b) enable the retina to convert light into nerve impulses and do an early preprocessing of those signals. Photoreceptor cells provide the required sensitivity and sensorial function to absorb the incoming light. This incoming light causes a photochemical reaction involving rhodopsin pigments which are responsible for transforming the light stimulus into electrochemical signals (photo-electrical transduction) [5, 12]. In case of the human duplex retina, two main types of photoreceptors can be distinguished, namely rods and cones [30]. Rod receptors show, compared to cones, a remarkably lower level of activation, which enables monochromatic vision under low-light-level conditions *(scotopic vision)*. In contrast to the high sensitivity (magnitudes of 100), the resolution and visual acuity are degraded compared to cones [12]. Under standard daylight situations the cone receptors become active. This enables the perception of color information with a high degree of details *(photopic vision)*. Cones can further be classified into **S**hort, **M**edium and **L**ong type, representing the wavelength of their highest corresponding response ($\lambda_{max} = \{420nm, 535nm, 565nm\}$) regarding the visible light [12].

The ~$6 \cdot 10^6$ cones and ~$120 \cdot 10^6$ rods are not equally distributed over the retinal surface [22]. The fovea centralis is located $5\,^\circ$ temporal on the eye's optical axis. It contains only cones with a density of approximately $1.4 \cdot 10^5$ cones/mm^2. As shown in Figure 2.2, the cones' density rapidly decreases with an increasing angle on the optical axis. Rods show the highest density at approx. $30\,^\circ$ with $1.5 \cdot 10^5$ cones/mm^2 decreasing to approx. $0.5 \cdot 10^5$ cones/mm^2. A closer look at the cross section of the retina reveals an indentation at the position of the fovea. In this cone-dominated region, the other nerve cell layers (bipolar, amacrine, horizontal and ganglion) are displaced from the fovea centralis to the peripheral areas [21].

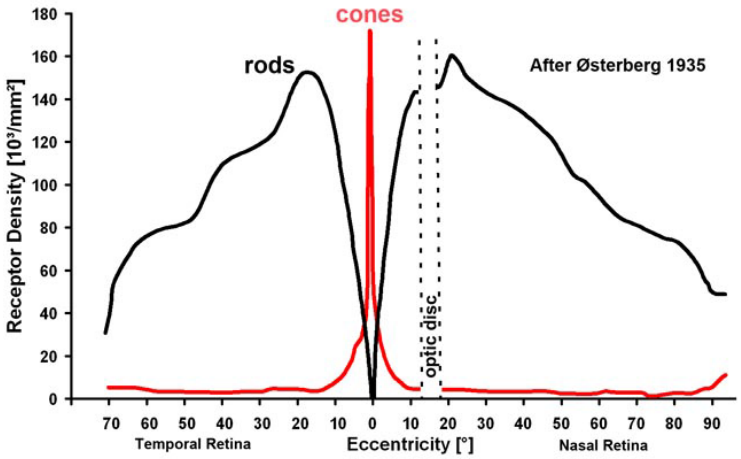


Figure 2.2: Retinal density distribution of rods and cones (figure from [39]).

Bipolar nerve cells create the connection between photoreceptor and ganglion cells. In the foveal region, the ratio of photoreceptors (10^8) to ganglion cells (10^6) is lower than in the rest of the retina. Hence, one bipolar cell is connected to only a few cones, allowing the ganglion cells a detailed processing of the receptor excitations, resulting in high visual acuity. As stated in Frotscher [22], electrophysiological studies have shown that groups of sensory cells are combined to receptive fields (RF) and act as functional groups. Those concentrically shaped receptive fields model the convergence of multiple photoreceptors to one single ganglion cell. As a consequence each ganglion cell has a receptive field [5]. These lateral connection links are established by horizontal and amacrine cells. The concept of RF divides the RF area into a circular area at the center and a toroidal surrounding area. Center-on RF cells show an impulse response when light falls into the center, while light in the surrounding area leads to repression of the impulse response. Center-off RF have the inverse characteristics. Consequently, this contrary signal behavior allows the perception of contrasts and object edges. [5, 12, 22]

Using the optic nerves, the output of the ganglion cells is projected to different areas of the brain in form of action potentials, where the topographical arrangement of the information is retained under the projection (retinotopic mapping). The main part of the ganglion cells projects to the lateral geniculate nuclei (LGN) of the thalamus, which is organized into six layers. Different cell sizes further categorizes the first two layer as magnocellular and the remaining four as parvocellular layers. Consequently, retinal ganglion cells can be further partitioned into M- and P-type cells. M cells provide information about fast changes and motions, which is enabled by their high- and low contrast flickering sensitivity. On the other hand, P cells transmit color information in terms of color opponency. This means that colors are represented as luminance, red-green and blue-yellow differences. As stated by Bruce et al. [5] the LGN has the function of a relay between retina and the visual cortex in the visual pathway. For further details about the visual pathways in the brain, the reader is referred to [5, chap. 3]. [5, 21]

Beside the sensorial functionality of the eyes, they have to be aligned to provide a foveal projection of an object, which is realized by the oculomotor system. In addition to the general

motion of the eye (abduction, adduction, elevation and depressive), the extraocular muscles are responsible to keep the eyes focused on the object. Image stabilization is achieved by information of vestibular organs (vestibular-ocular - VOR) and optokinetic reflexes (OKR). Target respectively object selection can be seen as initiator for higher visual and cognitive processes. Eye movements can be divided into saccades, slow pursuit movements and vergence movements. Saccades are quick (about $100\,^\circ\,\mathrm{sec}^{-1}$), short and abrupt motions of the eye for the foveal alignment of the target. Slow ($\ll 100\,^\circ\,\mathrm{sec}^{-1}$) pursuit movements allow the eye to follow the motion of the target and vergence movements are used to keep track in depth changes. Fixation relates to stationary phases of the eye where the optical information is processed. However, during saccades, a masking effect (saccadic suppression) occurs. Moreover, drifts and tremor movements in fixation phases are corrected by microsaccades. [12, 21]

2.1.2 Objection Recognition

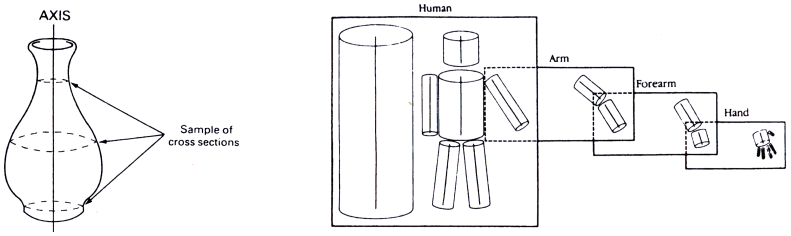
The previous section 2.1.1 presented the physiological pipeline of how the environment is processed by the eye and especially by the retina. The question of how we are able to utilize those resulting neuronal signals that arise from incoming light patterns on the retina and how they lead to impressions of objects is still a topic of research [5].

Marr et al. [46] formulate the visual perception as a computational theory to express which kind of information is essential for the visual system and how it is used. To process the variety of information types in a scene, an efficient representation is required. The modular framework proposed by Marr et al. [46] is structured into three separated and series-connected units where each uses one representation as input and transforms it into a different one.

1. **Primal Sketch**
   The first stage is based on a large gray-level intensity array as approximation of the retinal image. Introducing the concept of the primal sketch allows a reduction of a large 2D intensity array to a compact description by identifying local intensity changes in the image. Despite the information reduction, the relevant data for higher image analysis is preserved. Therefore, the primal sketch describes the position and the type of crucial changes in intensity. At first, the image is spatially filtered using image operators that correspond to the characteristics of the receptive fields of retinal and cortical cells. Applying filters of different size and combining features which are present at all scales results in the raw primal sketch. Defining a hierarchical relation between individual features of the raw primal sketch culminates in the full primal sketch that provides texture, contour and shading information. [5, 46]

2. **2½-D Sketch**
   In the second processing unit, the low level features provided by the primal sketch are used to derive a viewer-centered representation of the visible scene surface properties. This includes the relative position and depth as well as the orientation information and is formally known as 2½-D sketch. [46]

3. **3D Model Representation**
   In contrast to the 2½-D sketch, the third part of the modular framework aims at an object-centered representation of the scene. Based on the object shape and the shape axis, the object is decomposed into volumetric primitives. Figure 2.3a illustrates that symmetric objects can be described in terms of a set of generalized cones of variable size along the axis. When applying this kind of abstraction to an human figure as show in Figure 2.3b, this results in a set of cones, each representing different parts of the human body. Analogous to a stick figure, the components are connected in a hierarchical structure. Furthermore, Figure 2.3b demonstrates how the hierarchical aspect involves also different level of detail. This allows at the same time the matching of the object on several levels against the known models in memory. [5, 46]



(a) (b)

Figure 2.3: (a) Schematic decomposition of objects with generalized cones; (b) Modeling human figure using hierarchical structured representations with different level of detail. (figures from [5])

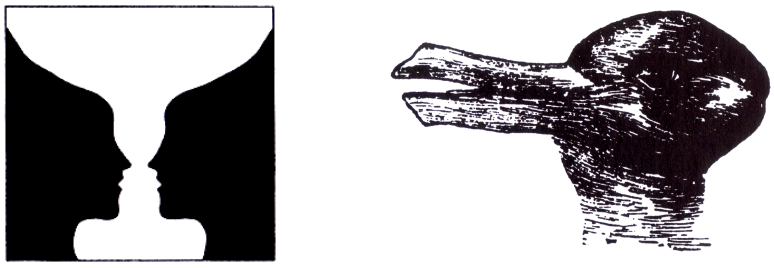
As we have seen, Marr et al. [46] proposed a one-way processing framework which aims at describing the visual processing in a purely bottom-up manner. It also defines where perceptually driven (top-down) aspects, like knowledge of the environment, are included to solve ambiguity problems [5]. Moreover, instead of finding direct physiological links to the behavior of cells, Marr's approach defines the algorithm on which the cells operate. It is part of Marr's three levels of theory: the *computational level*, as stated before, describes the problem of visual perception and the *algorithmic level* formulates the methods which are used to solve the problem; psychology and neurophysiological evidence explains the way how the algorithms are implemented (*implementation level*) in the pathway [5].

Gestalt Theory

According to Marr et al., the full primal sketch is formed by grouping individual low-level features together. In this section, we go into more detail on how perceptual organization is achieved in conjunction with the Gestalt theory.

In our usual view of the environment, we are able to recognize objects without problems as long as objects do not have camouflage properties. Furthermore, we can associate properties (e.g. texture, boundaries etc.) with objects. From this follows the ability to differentiate between the object and the background. Nevertheless, it is possible to create images like the well known

face / vase picture by E. Rubin shown in Figure 2.4a that demonstrates the possible ambiguity of fore- and background. At each point of time we are able to perceive only one of the two figures, either the pair of faces or the white vase. Figure 2.4b shows that the ambiguity can also occur in the figure itself. The figure in Jastrow's picture can be perceived as a duck or rabbit, but not at the same time. While both examples are constant in their 2D features, the higher level interpretation of the features leads to different descriptions that can vary abruptly. [5]



(a) (b)

Figure 2.4: (a) face / vase picture by E. Rubin (b) duck-rabbit image by J. Jastrow (figures from [5])

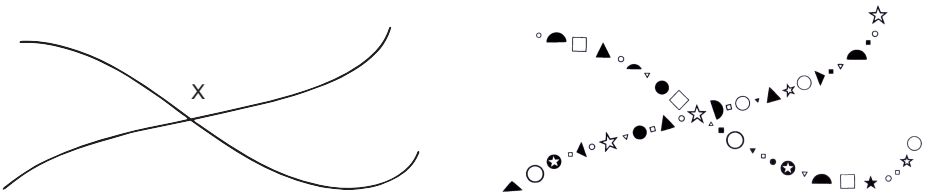
The Gestalt theory principles describe the grouping effects of the perceptual organization. Based on the following principles, the impression that individual subregions of a figure are part of a larger region emerges [5]:

- **Proximity and Similarity**
  Objects within a scene that are located near to each other are grouped together. The dots in Figure 2.5a appear either as rows (increased vertical spacing) or as columns (increased horizontal spacing). For equally spaced dots, neither rows nor columns can be perceived (Figure 2.5c). Similar to proximity, objects which have a similar appearance are grouped together. Figure 2.5d illustrates the combination of proximity and similarity, where despite higher vertical spacing, the grouping of proximity is overridden by similarity, resulting in column-wise grouping.

(a) (b) (c) (d)

Figure 2.5: Perception of columns (a) and rows (b) based on effects of proximity; (c) shows ambiguity for equal horizontal and vertical spacing; (d) despite higher vertical spacing the proximity effect is overridden by similarity. (figures from [5])

- **Common Fate and Good Continuation**
  Objects that have a common direction and speed (e.g. swarm of birds) are grouped together. Beside the grouping by motion, the property of *good continuation* is favored over rapid changes - e.g. the lines in Figure 2.6a are perceived as two smooth lines crossing at point X instead of two v-shaped lines with an abrupt change at X. Even when objects significantly differ in appearance, proximity and continuation can lead to a grouping impression (Figure 2.6b). „Good continuation may be considered the spatial analogue of common fate“ [5].



(a) (b)

Figure 2.6: Good continuation - (a) lines are perceived as smooth lines; (b) continuation despite different appearance. (figures from [20])

- **Closure**
  Even when parts of an object are not present, we tend to mentally fill in additional information to form a closed representation of the object. As a consequence, closure allows us to perceive objects. The set of arc-shaped lines in Figure 2.7b is perceived as a broken circle and Figure 2.7a appears to be a horseman instead of a cluster of meaningless speckles. [5, 20, 52]

(a) (b)

Figure 2.7: Due to the effect of closure, in spite of missing parts, our brain completes (a) to a horseman and (b) to a circle. (figures from [20])

- **Relative Size, Orientation and Surroundedness**
  Figure 2.8 shows overlapping objects with different relative areas. The smaller black object in Figure 2.8a is perceived as a figure on the large white circle. Decreasing the size of the black figure magnifies the impression of being a foreground figure (Figure 2.8b). Figure 2.8c illustrates the human preference of horizontally and vertically oriented figures. Hence, the white area seems to be a foreground figure on a black background.

(a) (b) (c)

Figure 2.8: Discrimination of fore- and background objects; (a) perception of a black propeller on white background; (b) enhances the effect of (a); (c) vertical and horizontal alignment of the white regions allows the white regions to be perceived as foreground. (figures from [5])

The presented Gestalt principles manifest in the *law of Prägnanz* or *law of good figure*. "Every stimulus pattern is seen in such a way that the resulting structure is as simple as possible." [23] When taking a closer look at the four dots in Figure 2.9, their arrangement implies the impression of corners of a square instead of a cross-shaped figure. "The square is a closed, symmetrical form, which the Gestaltists maintained was the most stable." [5] From the perspective of research in computer vision, Mitra et al. [47] proposed an approach that allows to render images from 3D models by considering the main principles of the Gestalt theory. The proposed synthesis method creates images that allow only humans to perceive the projections of the 3D models from the whole image, while selective parts of the image do not contain meaningful information.

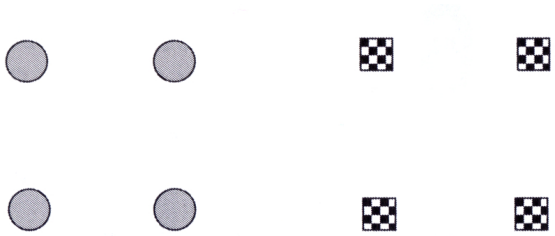


Figure 2.9: Law of Prägnanz - perception of dots as corners of a square instead of a more complex form, e.g. a rotated plus-sign. (figure from [5])

Depth Perception

Paintings and photographs can create the impression of depth, despite the fact that their flat 2D surfaces does not contain any real depth information. The visual result of this project corresponds to a series of 2D images viewed on a standard screen. Therefore, this section aims at on the derivation of monocular depth-cues based on "flat" 2D images. For example, *linear perspective* appears when viewing along railway tracks so that they seem to converge in the horizon.

Based on the fact that railway tracks remain parallel, even in the horizon, the linear perspective creates the effect of great distance. When arranging different instances of objects in depth, decrementing the *relative size* of an object implies a greater distance from the viewer. When the distance in depth between objects is small, relative size can lead to ambiguous information. *Overlap* or interposition between the objects resolves this ambiguity and creates a well-defined ordering of the objects. Occlusions can result in a partially visible object. Despite the reduced information about the occluded object, humans perform a filling-in-process of information to create a complete and stable representation of the object, which is known as amodal completion (also see 2.1.2). [5, 9]

Apart from pictorial depth cues for static images discussed so far, *relative motion* of objects in an image sequence also contributes to the perception of depth. For a static viewer, the motion of two objects at different distances (Figure 2.10) leads to different relative motions of the projected retinal image (i.e. the relation between the speed of the images is inverse-proportional to the object distance); the relative motion corresponds to the motion parallax. [9, 32]

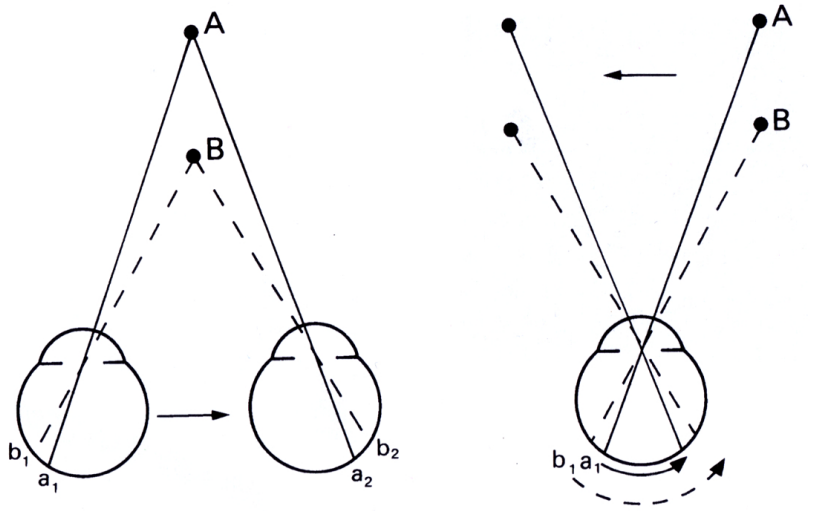


Figure 2.10: Illustration of the motion parallax when observing two objects A and B at different depths. Motion of the observer (left) or the objects (right) causes higher retinal distances ($b_1 - b_2$) for the nearer object B than for object A ($a_1 - a_2$). (figure from [5])

In Marr's computational theory, the several types of depth cues relate to parallel processing units. Despite the mixed information input from various depth cues, our depth-perception is encapsulated in a single form of representation; this relates to an integration of the depth cues. Marr's theory describes this kind of integration by the concept of the 2½-D sketch. As already discussed in the beginning of this section, the 2½-D sketch uses the available depth cues to construct a viewer-centered representation of the scene surfaces, containing relative depth information. [5]

2.2 CAPTCHA

The abbreviation CAPTCHA was first introduced by Ahn et al. [2] and stands for *Completely Automated Public Turing Test to Tell Computers and Humans Apart*. As the name already reveals, the concept of CAPTCHA is to formulate a test in a way such that humans can solve the task with minimal cognitive effort, while the given task remains a hard or unsolvable problem for computer programs. More detailed, the term *public* relates to the idea of an open source concept to avoid the establishment of black box systems (security by obscurity principle). In addition, the word CAPTCHA defines a relation to the Turing-Test. In this test, a human judge is confronted with two separated players, where one of the players is a machine. In terms of a natural conversation, both players try to convince the judge of being human. If the judge is not able to differentiate between human and machine, despite intensive interrogation, the machine passed the Turing-Test. That way, CAPTCHAs are similar to Turing-Tests, only that humans and machines are discriminated by a machine; CAPTCHAs allow to perform an automated Turing-Test. As a consequence of the definition, the underlying problem of a CAPTCHA challenge should relate to an open artificial intelligence (AI) problem (e.g. recognize distorted text). [2, 3] „A CAPTCHA implies a win-win situation: either the CAPTCHA is not broken and there is a way to differentiate humans from computers, or the CAPTCHA is broken and a useful AI problem is solved.“ [3]

2.2.1 Applications

A common problem of web-services are automated malicious interactions with their interface. Utilizing the CAPTCHA concept, the abuse through automatic processing is intended to be blocked in an early stage. The flexibility of CAPTCHA-tests allow the application in a variety of systems:

- **Online Polls**
  In 1999, the online magazine slashdot.com started an online voting about the best graduate school in computer science. This voting ended, despite the recording of the participant IP addresses, in a competition between voting-bots written by students of Carnegie Mellon University (CMU) and students of the Massachusetts Institute of Technology (MIT). While the MIT (21.156 votes) won the voting by a narrow margin against CMU (21.032 votes) other universities did not exceeded the level of 1000 votes and therefore illustrates the manipulability through automatic systems. [2, 3]

- **Email Provider, Social Networks and Online Marketplaces**
  Free email provider protect their sign up forms to avoid automated mass-registrations of email addresses, which are furthermore used as source for email-spam. [49] On social networks like Facebook.com, CAPTCHAs help to block the creation of fake profiles, whose main purpose are phishing and scam attacks. On Ebay, CAPTCHAs shield the marketplace against mass-generated defrauding auctions. Other potential applications for this kind of security mechanism are blogs and bulletin boards, because these platforms are usually target for the propagation of spam in form of fake comments and postings. [7]

- **Search Engine Indexing**
  While the robots exclusion standard is only a convention to control the indexing-process of a website by well-behaving search spiders and crawlers, a more strict access control can be achieved for instance by using CAPTCHAs to lock out crawlers from sensible parts of a website. [2, 3]

- **Protection against Dictionary Attacks**
  Testing thousands of passwords using a dictionary approach could be prohibited by including a CAPTCHA challenge after a certain number of wrong attempts. [2, 3]

Taking into account that Ebay delivers 14 million CAPTCHA challenges per week [7] and Ahn et al. [4] report to serve more than 100 million CAPTCHA challenges every day on over 100,000 websites, the CAPTCHA concept can be seen as well spread and common on the web.

2.2.2 State of the Art

CAPTCHA challenges are not necessarily limited to a visual representation. Yan and El Ahmad [61] describe the classification into the following three main types:

- **Text-based Schemes**
  Text-based CAPTCHAs are the most frequent type on the web. They are characterized by applying distortion-patterns to generated or natural language strings in a way such that they remain readable for humans but become unrecognizable for machines. [4, 7]

- **Sound-based Schemes**
  To pass this type of challenge, the user has to enter a sequence of signs, usually numbers, dictated by a distorted and superimposed voice. Due to accessibility reasons, audio-based systems are used in combination with other types. [6]

- **Image-based Schemes**
  The puzzle challenge focus on performing an image processing task based on synthetic or natural images.

Based on these types, we take a closer look at state of the art CAPTCHA approaches in the following.

reCAPTCHA

The most popular and representative example of a text-based CAPTCHA system is the reCAPTCHA approach proposed by Ahn et al. [4]. The idea of this approach is to utilize the human cognitive effort required to solve a CAPTCHA puzzle for a useful purpose. The high degree of distribution and the fact that more than 100 million challenges are served every day highlights the high potential of aggregated human workforce, which would otherwise end up being wasted.

When solving the text-based reCAPTCHA challenge, a user helps to transcribe words that stemming from a book digitization process that cannot be recognized reliably by state of the

art optical character recognition (OCR) systems. Due to the validation constraint, a challenge consists of two words, the *control word*, which is known, and an *unknown word* (see Figure 2.11). This concept is based on the idea that if the user transcribes the control word correctly, it can be assumed that the unknown word is also transcribed correctly. User inputs for unknown words are interpreted in the form of votes; when a certain transcription of an unknown word reaches a specific number of votes, it becomes a control word for future challenges.

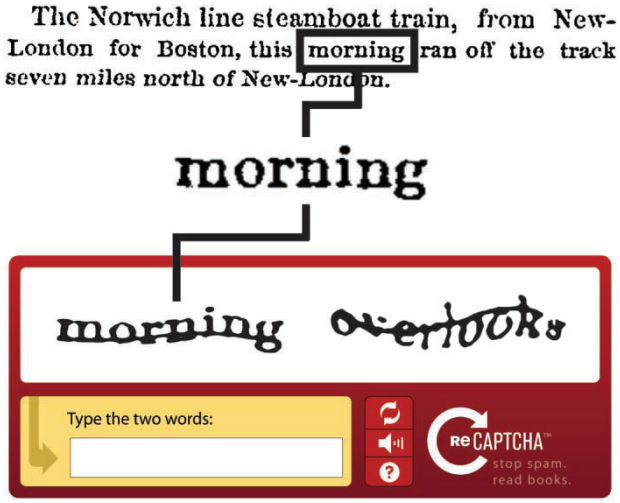


Figure 2.11: reCAPTCHA challenge and the relation to scanned book text. (figure from [4])

Based on the fact that the pool of control words consists of words that cannot be recognized by two state of the art OCR systems, an algorithm which reliably recognizes the given distorted words would imply an improvement in OCR research. Ahn et al. [4] stated that their approach reaches a transcription accuracy of 99.1% compared to 83.5% of standard OCR systems using two manual human transcriptions as ground truth. Moreover, the result after a runtime of one year are 1.2 billions processed challenges, which equals the transcription of about 17,600 books (assuming 100,000 words / book). [4]

TagCaptcha

Image retrieval provides a query-based interface which enables users to find and access relevant images in an image collection like a database. In text-based frameworks, the formalism of the query is defined by a high level keyword description of the desired images. As a consequence of text-descriptors, additional annotation metadata for the images is required. A common problem in the task of image retrieval is the non-existing direct relation between low-level and high-level features, known as *semantic gap*. Therefore, the semantic gap hinders an automatic annotation of the image database. [44]

With TagCaptcha, Morrison et al. [48] propose a concept to overcome the problem of image-annotation by including the task into an image-based CAPTCHA system. Similar to the re-CAPTCHA approach, a TagCaptcha challenge consists of a set of images. More precisely, the

set is composed of an *unknown* and a *verification / control* subset. As illustrated in Figure 2.12, the user describes the presented set of images using English keywords. If an image of the unknown set is tagged with at least two similar keywords from different users, the image becomes part of the verification set. For the verification of the user input with the corresponding set of control keywords, a two step matching-process is used. As a first instance, the base form of the input is determined and tested for matches. If the matching process ends without agreements, the similarity of the given input is compared against WordNet [54] using the WUP measurement proposed by Wu and Palmer [60]. The WUP distance defines the allowed hierarchical semantic distance between keywords (e.g. horse and camel are both animals) and therefore specifies the ratio between usability (accepting higher level descriptions like e.g. *animal* for *horse*) and vulnerability of the system.

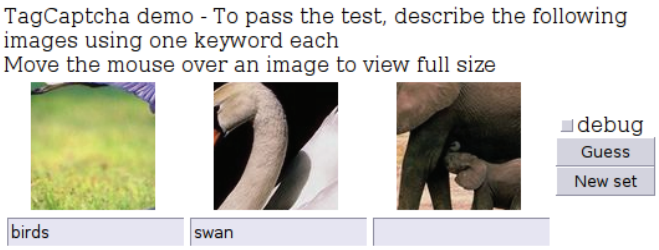


Figure 2.12: TagCaptcha interface (figure from [48])

Within the scope of a user study with 12 participants performing 20 CAPTCHA challenges, Morrison et al. reported a success rate of about 70%. The authors state that the relative low success rate, compared to text-based CAPTCHAs (approximately over 90%), could be reasoned by the high degree of subjectivity involved in the annotation process. Moreover, language barriers, context and the aspect of personal experience can have influence on the individual annotation performance.

Asirra

Another representative of image-based CAPTCHA systems is Asirra, introduced by Elson et al. [15]. Similar to TagCaptcha proposed by Morrison et al. [48], the challenge is defined by a higher-level description problem. More precisely, a challenge consists of 12 images provided by petfinder.com, where the user has to select either all cat or all dog images from the mixed set of images. The petfinder.com database consists of millions of manually annotated images of homeless pets, increasing by 10,000 new images each day. Besides the integration of links for adopting the shown pets (Figure 2.13), user-related location information is considered to select primarily pets that can be adopted in the user's area. Based on a user study including 332 participants, Elson et al. have shown that 99.6% of the users can solve a challenge in less than 30 seconds. Therefore, the Asirra approach shows how CAPTCHAs can be utilized for a general purpose in a social context.

Figure 2.13: Asirra challenge (figure from [16])

AniCAP

Chow and Susilo [8] introduced AniCAP, an animated 3D CAPTCHA which targets the human ability to perceive depth from relative motion (see also 2.1.2) based on the motion parallax. In this text-based approach, the main characters of the challenge are overlapping background characters which have the same properties regarding to font and color. By arranging main and background characters at different levels of depth, the impression of depth in an animated scene is enabled (Figure 2.14a). To avoid occlusion effects as described in Section 2.1.2, foreground characters have additional transparency-properties. The resistance regarding segmentation is realized by applying local (Figure 2.14b) and global (Figure 2.14c) distortions to individual characters respectively the whole scene of the animation.

(a) no distortions (b) local distortions (c) global and local distortions

Figure 2.14: Overview of AniCAP distortions (figures from [8])

2.3 Crowdsourcing

The term *crowdsourcing* was first introduced by Howe [33] and is the composition of the two words *crowd*, an abstraction of a group of participating individuals, and *outsourcing*. Moreover, the usage of this term in a variety of applications resulted in a continuous development and evolution of the term's definition and meaning. Motivated from the inconsistent definitions, Estelles-Arolas and Gonzalez-Ladron-de Guevara collected and compared 40 state-of-the-art definitions from scientific publications in their paper [18]. Based on the analysis and interpretation of the series of definitions, the authors stated the following general definition of the word crowdsourcing: „Crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. The undertaking of the task, of variable complexity and modularity, and in which the crowd should participate bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while the crowdsourcer will obtain and utilize to their advantage that what the user has brought to the venture, whose form will depend on the type of activity undertaken.“ [18] At this point it has to be noted that the proposed definition aims at covering a wide spectrum of applications. Hence, this generic definition has to be adapted and extended to a specific and focused context.

2.3.1 Amazon Mechanical Turk

A popular representative for crowdsourcing is the online platform Amazon Mechanical Turk (AMT) [34] with more than 200,000 registered workers. The concept of AMT is built on the idea to break down tasks or given problems to a series of *microtasks*, which can be performed by the crowd of workers. More in detail, requesters can post microtasks in form of Human Intelligence Tasks (HIT), which define the task and the reward for performing the HIT. The payment for HITs is specified by the requester and ranges from 0.01$ for short tasks like doing a Google search, to e.g. 30$ for more complex tasks like transcriptions of audio recordings. A massive on-demand workforce characterized by a high degree of diversity in education, knowledge, experience and demographic background gives researchers the opportunity to utilize a high number of individuals at low monetary costs. With the diversity mentioned, the question of significance and quality of the collected data arises. The popularity and the open-call concept of the AMT also attracts malicious workers, which have to be considered when discussing the aspect of quality. A common method of cheating is to provide random answers to minimize the completion time while maximizing profit. Furthermore, Kittur et al. [38] states that there is a relation between completion time and random answers. This process of submitting random answers can be automated by using bots that harvest available HITs. Well-organized spammers act in groups, sharing answers or trying to provide the same feedback for a given task to achieve a high agreement. Instead of sharing answers, another method of attacking the microtask market is to reuse prior submitted answers in cloned instances of workers. [13, 38, 55]

Motivated by the question of the practical quality, Kittur et al. [38] investigated how the Mechanical Turk can be used for the purpose of user studies. Within the scope of two experiments,

the task for the participants was defined by rating 14 Wikipedia articles based on the Wikipedia-featured article criteria. To estimate the quality of ratings, the results are compared to ratings from experienced Wikipedia users. The first experiment ended with 210 ratings submitted by 58 workers. When analyzing the free-text responses, about the half of the feedback relates to non-relevant data. As an additional indicator for fraud response, samples with completion times less than one minute (64 ratings) were marked as invalid. From about 123 (58%) flagged samples, 73% are related to only 8 users, representing only a small group of malicious workers. On the basis of the first experiment, the aim of the second experiment was to reduce the high amount of invalid responses. The task's concept of the first experiment was extended by additional quantitative and automatically verifiable questions like e.g. the number of figures or references. With the new design, 124 worker submitted 277 ratings. On the one hand the per user rating was lower, but on the other hand the amount of flagged responses was reduced to 7, compared to 127 from the first experiment. Kittur et al. concludes that considering quantitative and verifiable control questions into the process of designing the experiment can remarkably reduce invalid responses. Furthermore, the concept of the experiment should be designed in a way such that the effort of deliberately providing wrong feedback is equal to the effort of faithfully completing the task. Taking the proposed design recommendations into account allows the utilization of the AMT in a meaningful way for conducting research.

2.3.2 Image Annotation

As described in Section 2.2.2, image annotation relates to the process of generating additional image metadata which can be used e.g. for retrieval tasks. As an example for text-based annotation, we took a closer look at the TagCaptcha [48] approach that embeds the concept of image annotation and crowdsourcing in terms of a CAPTCHA system.

In contrast to microtask platforms like AMT, games offer a different kind of workforce which is mainly motivated by entertainment and the competitive challenge. With the ESP game for example, Ahn and Dabbish [1] presented an interactive online system where a crowd of players perform the task of annotation in a gaming situation. The game is designed in a way such that two players are randomly paired without knowing the identity of each other. Both players get the same images from a database and have to annotate them. The main idea of the ESP game is that the two players find a consensus regarding their annotations. More precisely, without knowing the keywords of each other, the current image is processed if both players enter the same keyword. Hence, the aim of the players is to find agreements in as much images as possible within 2.5 minutes (see Figure 2.15). To avoid repeated agreements on too general keywords (e.g. animal or person), each keyword match is added into an a taboo list related to the image. If the image occurs again in an other round, the users are not allowed to submit words contained in the taboo list. Using the concept of a taboo list increases the degree of difficulty and guarantees the diversity of the image tags at the same time. As a consequence, only keywords with an existing agreement are used for annotation task itself.

After a four-month evaluation of the ESP game, Ahn and Dabbish reported that 13,630 user annotated 293,760 images with 1,271,451 tags. Based on the average annotation rate of 3.89 images per minute, 5,000 permanent players could label a database containing 425,000,000 images in 31 days. These results highlight the potential benefit of crowdsourced metadata for large image collection as e.g. the Google image search. [1, 58]

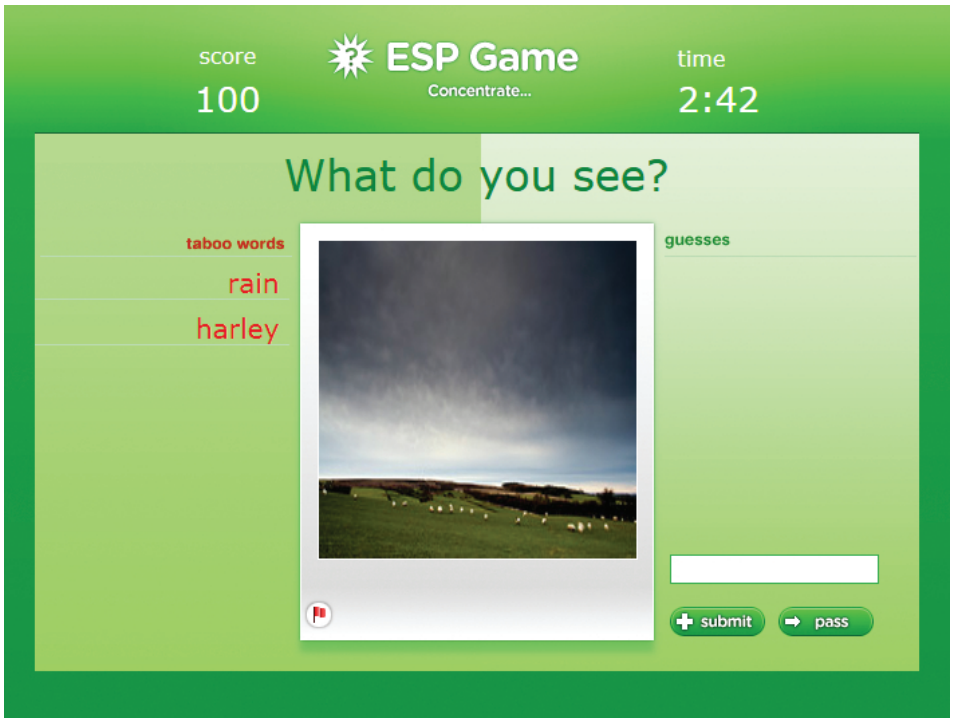


Figure 2.15: ESP game interface (Figure from [27])

2.3.3 The Dark Side

In previous sections, we pointed out that crowdsourcing can be a powerful concept to perform non-automatable tasks with purpose. However, negative drawbacks are usually related to the crowd's heterogeneity, worker payment conditions and quality. This section is focused on those aspects, where the immense workforce of a crowd processes tasks with unethical characteristics. The open structure of the internet and the related demographic diversity makes it difficult to enforce strict policies against unethical behavior. In this sense, the definition of unethical behavior can change remarkably between different demographic groups. Studies have shown that compared to real life situations, the „anonymity" of the internet lowers one's inhibition level for unethical behavior, which allows to find more people to perform such tasks at the same time. On the other hand, by design of a task, its unethical manner could be hidden by splitting the task in independent subtasks, while participants performing single subtasks are unaware of the whole picture. [29]

Harris [29] described three common techniques that are used by task requester with unethical motives:

- **Social Engineering**
  Social engineering is used to deceive and manipulate people to expose sensible information or perform certain actions. Identity theft and phishing are common examples for social engineering with a financial intention. In relation to the crowd, peer pressure specifies another form of social engineering; swimming against the tide is seen as inconsistent to the human nature. Forming an artificial group with the crowd could be used to establish the situation of peer pressure at minimum costs. [29]

- **Attack and Run**
  User accounts on websites are a wide-spread method to assign a specific identity to a user. Depending on the degree of anonymity, it can be distinguished between identity *intensive* and *relaxed* websites. While identity intensive sites aim at verifying the real identity of a user (e.g. paid subscriptions), identity relaxed systems preserve the user anonymity to some degree, which can have negative effects when using those identity for validation on other websites. This loose handling of identities can be seen as potential starting point for building false identities. [29]

- **Human Computation Tests**
  Using the human computation capacities in a malicious context directly addresses the previously described concept of CAPTCHAs (see Section 2.2). Per definition, CAPTCHAs are designed in a way such that humans can solve the challenges with minimal effort and therefore represent an ideal candidate for the human computation possibilities of the crowd. According to Motoyama et al. [49], bypassing CAPTCHAs with human laborers has grown to an industrial market. Costs for on-demand human-based CAPTCHA-solving services are oriented on high quantities of CAPTCHAs and start at 1$ per 1000 challenges. Outsourcing the task to low-wage countries over the internet in a global manner keeps the occurring costs at a minimum. Wages of workers are commonly arranged at approximately 0.75$ per 1000 CAPTCHAs. The increasing pressure of competition and the fact that solving CAPTCHAs does not require specialized skills leads to a shift of the human labor from Eastern Europe to workers from Bangladesh, China, India and Vietnam. Nevertheless, Motoyama et al. [49] tested 8 providers for their quality and reported a response accuracy of 86-89% , which is remarkable higher compared to the accuracy of 18% achieved with OCR solutions like CaptchaOCR. Due to the fact that a million of CAPTCHAs can be solved for less than 1000$, CAPTCHAs are no guarantee against automated large scale access. From an economic point of view, Motoyama et al. concludes that the usage of solving-services has a significant impact on the attackers profit and is thus limited to attackers whose business model is efficient enough to compensate these costs.

  As an alternative to paid solutions, another approach is to include the solving process into a different task. Based on this idea, Egele et al. [14] proposed the injection of 3rd party CAPTCHAs into a web session of an unsuspecting user. On the previously infected user

system, interactions with web applications are intercepted, allowing the attacker to seamlessly smuggle CAPTCHAs directly into the session. Since the smuggled CAPTCHAs are injected within the workflow of the web application, the victim can not differentiate between smuggled and authentic CAPTCHAs of the application. In the evaluation process, 17 working installations of the malicious browser plug-in were achieved, requesting 167 CAPTCHAs during a test period of 14 days. Results report a success rate of 75%. When resending unanswered challenges to other users, the success rate increases to 93%. Applying these results to large scale dimensions of botnets, the low costs for infrastructure and the capability of bot masters to upgrade „clients“ with the malicious plug-in points out the potential of the approach described.

Harris [29] highlights that combating this techniques of unethical behavior is still an open problem. Laws that sentence such behavior are too inflexible and difficult to enforce at a global scope. Moreover, Harris notes that teaching ethic as part of eduction or a reputation oriented identity system could reduce the unethical progress on crowdsourcing platforms. The global characteristic of the crowd minimizes the effect of the proposed suggestions at the same time, because the problem is shifted to regions where the revenue plays an overriding role to ethics. [29]

CHAPTER 3

Overview

This chapter provides an overview over the CAPTCHA concept and describes the role of the proposed obfuscation method in this context.

3.1 The Big Picture

As mentioned in Chapter 1, 3D model databases like Google's 3D Warehouse [24] require high-quality metadata to provide effective retrieval and exploration functionalities. The generation of this metadata is still an open problem. Ahn et al. [4] demonstrated with the reCAPTCHA approach that crowd-based data generation can reach large-scale dimensions while performing the role of a security mechanism at the same time. Inspired by the reCAPTCHA [4] approach, the main idea of this work is motivated by the annotation of 3D models with additional metadata using the CAPTCHA (see Section 2.2) process. Similar to distortions applied in text-based CAPTCHAs, our approach uses an obfuscated representation of the 3D models to meet the security characteristics of CAPTCHAs. A challenge is the visualization of 3D models in an obfuscated manner, which have to be annotated. More precisely, as in the reCAPTCHA approach, two models are used for a challenge, implementing the idea of an *unknown* object, which has none or sparse metadata, and a known or *control* object, which has a reference set of keywords assigned.

Figure 3.1 illustrates the schematic structure of the proposed system. From a user perspective, a challenge is the task of recognition and textual description of two obfuscated objects presented in form of a video. In detail, the user response can contain relevant perceived features or the semantic description of the shown object, given as keywords. Those keyword-responses are processed by the *Challenge Manager*. As a first step, the user is validated to be a human by checking the response for the control object with the reference set, stored in the metadata database. Comparing keywords can include a series of processing steps, starting from quick string matches up to a check against a semantic hierarchy as used in [48]. The advantage of verification on a semantic level is that a response does not have to match at character level, because keywords are compared based on their meaning. If the annotation of the control object

is successfully verified, it is assumed that the annotation of the unknown object is also correct, resulting in new keywords for the unknown object.

Furthermore, the Challenge Manager is responsible for maintaining the set of keywords for a specific model. Depending on the size of these sets, the models are selected for future challenges. A challenge puzzle is represented in form of a video containing the animated and obfuscated visualization of two selected objects. The Challenge Manager requests new challenge videos from the *Obfuscated Renderer*, which directly uses the 3D model repository for the generation of the obfuscated output. As the core element of the framework is defined by the visualization component, the scope of this work is the conception of the rendering element that will be explained in detail in the course of the following Chapter 4.

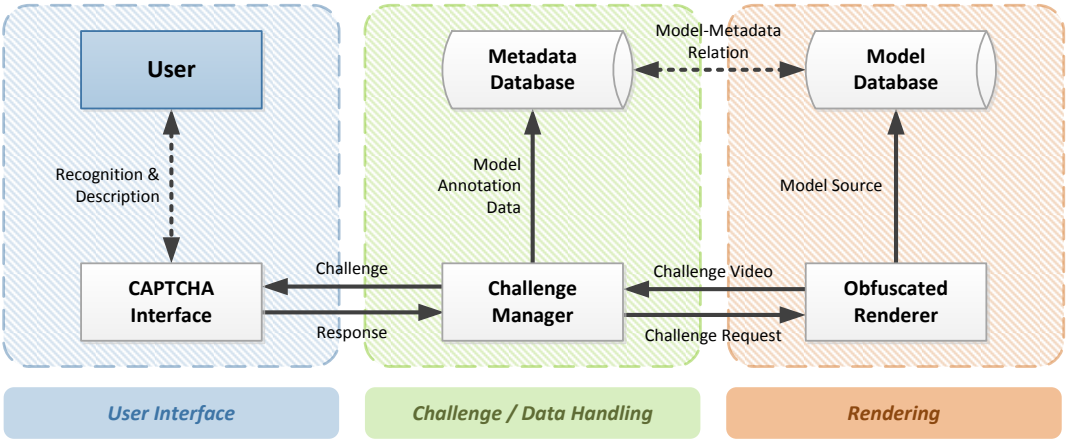


Figure 3.1: 3D CAPTCHA Framework Overview

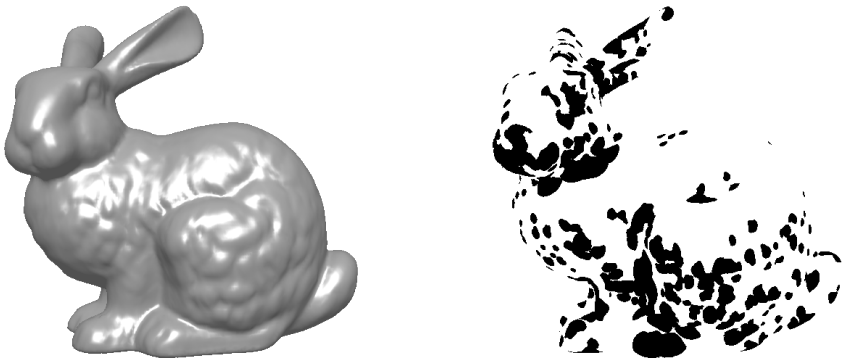
3.2 Obfuscated Rendering – The Main Principle

According to the definition of the CAPTCHA concept (see Section 2.2), only humans should be able to pass a challenge. To protect the system against automated attacks, we use a synthetization method that provides an obfuscated representation of a given 3D object.

The basic idea of this approach is to use just the salient parts of the objects for the visualization, while remaining parts are neglected (see Figure 3.2). Due to the capability of humans to mentally fill in missing parts in an image, the resulting sparse representation is sufficient for the recognition of the objects. The actual location of the object is hidden from automated approaches by introducing additional clutter in the background. This background obfuscation is achieved by copying parts of the object into the background and applying additional distortions to them.

Moreover, as the 3D databases contain primarily static models, continuous object transformations are used to create an animated scene. This animation enables the user to perceive the salient features of the object that could be distributed across the whole object geometry. In many

cases, color and texture features are crucial for the detailed identification of the objects shown. Based on the textures of the object, the dominant colors are determined and used for coloring the obfuscated visualization (context color). Texture features, on the other hand, are integrated in the representation by locally revealing the object surface. Providing these additional object features allows a more precise identification of the object and is likely to improve the quality of user annotations.



(a) (b)

Figure 3.2: The difference between source model (a) and the result of the object obfuscation (b).

3.3 Comparison to Related Work

Compared to related CAPTCHA approaches (see Section 2.2), which are primarily based on the obfuscation of text strings, the proposed method is motivated by obfuscating 3D models in a manner such that the most relevant object features are shown for recognition and annotation in terms of the CAPTCHA task. The approach introduced by Mitra et al. [47] represents the basis for this work; it aims at the generation of static images by revealing low-frequency geometry features. Our method extends this approach by focusing on the animated visualization of static models. Furthermore, the technique of Mitra et al. is extended by including additional color and texture features of the model, enabling a more specific annotation of the objects.

CHAPTER 4

Obfuscated Rendering

In Section 3.1, the rendering component was put into the context of the whole framework; this section provides a detailed description of the obfuscated rendering engine. Figure 4.1 describes the abstracted schematic view of the obfuscation pipeline. The schematic organization shows that the structure of our pipeline is composed of four components – the 3D model processing, the object obfuscation, the background synthesis and the combination of the object with the background.

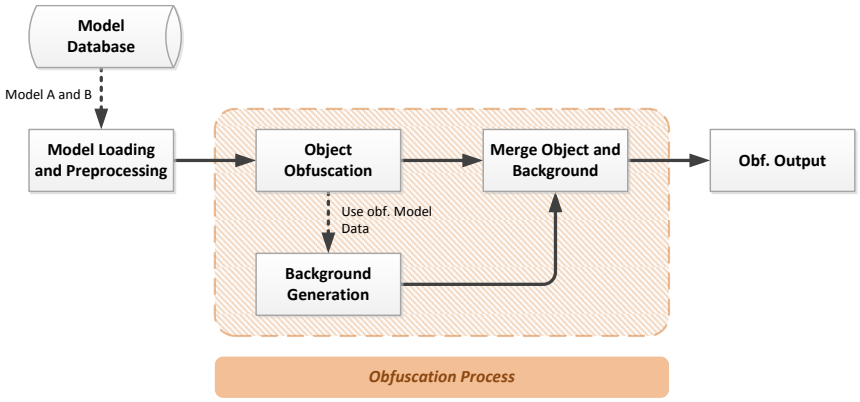
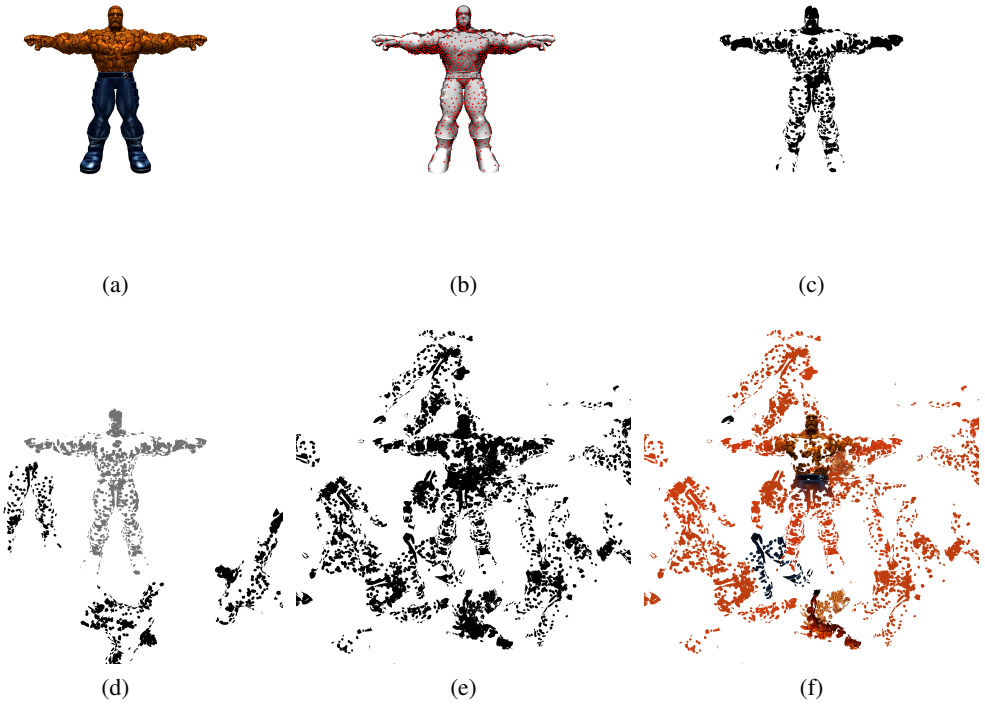


Figure 4.1: Obfuscated Renderer Overview

Section 4.1 describes how the scene has to be set up for the subsequent obfuscation process. This step includes scaling and aligning of the 3D models based on perceptual parameters as well as the transformation of the objects according to their desired pose in the animation sequence. The object obfuscation, presented in Section 4.2, is the first step of the actual obfuscation process. This step generates a sparse representation of the object by removing parts of the object which are not essential for the recognition task (see Figures 4.2b and 4.2c). Section 4.3 explains the generation of background clutter which shows similar spatial and temporal characteristics as

the object (Figure 4.2d). As described in Section 4.4, the results of both obfuscation steps are composed into a single image (Figure 4.2e). In course of Section 4.5, we introduce additional experimental features (see Figure 4.2f), to integrate color and texture features in the obfuscation process, which allow a more detailed annotation by the users. Section 4.7 covers the technical and design challenges of the implementation.



(a) (b) (c)

(d) (e) (f)

Figure 4.2: Obfuscation processing steps – (a) source model, (b) importance map and sampling, (c) object obfuscation, (d) background generation, (e) blending of the object with the background and (f) experimental features.

4.1 Model Processing

Well-known services like Google's 3D Warehouse [24] or TurboSquid [59] provide access to several hundred thousands of 3D models through their collections. In general, these models are used as 3D representations of real objects based on polygon meshes. However, the majority of available models are only static and do not contain any kind of animation. To model surface characteristics like color and structure, the usage of textures is common. Since 3D model databases do not always have textures, they have to be seen as optional data; for this thesis, we therefore use only *static* 3D models with *optional* texture information.

4.1.1 Foveal Coverage

As a first step of the processing pipeline, the 3D model is loaded and converted into a hierarchically organized structure. Due to its arbitrary scaling, it has to be normalized to a common scene dimension. A simple approach could be to extract the axis-aligned bounding box (AABB) and rescale the object so that the AABB has unit size. While this method allows an exact and well-defined alignment of multiple objects in a grid, we introduce a rescaling approach that is motivated by perceptual considerations. As described in Section 2.1.1, the fovea enables sharp vision at the center of the retina. Simplifying, this behavior can be modeled as an infinite cone, originating at the eye, with an aperture angle of approximately $1 - 2\,^{\circ}$ [12]. As the output of the rendering process is a 2D image displayed on a physical screen, intersecting this cone with the screen results in a conic section (see Figure 4.4). Assuming the viewing direction of the observer is orthogonal to the screen, the resulting circular area of the intersection corresponds to the area that can be perceived sharply by the fovea and it is given by:

$$\begin{aligned} r_f &= tan\left(\frac{\alpha}{2}\right) * d_{eye} \\ A_{fc} &= r_f^2 * \pi \end{aligned} \tag{4.1}$$

where α denotes the aperture angle of $1 - 2\,^{\circ}$, d_{eye} the distance from the eye to the screen and A_{fc} the foveal area. Knowing the physical dimensions $width_p$ and $height_p$ of the display device and the screen resolution allows determining the physical size of a pixel.

When rendering an object, the area of the projected image depends on the virtual view point. For example, the projected area of a flat screen TV model differs significantly between the front and side view. To overcome this issue, we use the AABB to approximate the projected object area. In detail, the cross sectional area of the AABB (depicted in Figure 4.3) is projected into screen space, resulting in the area A_{obj}. The perceived area of A_{obj} can be determined by multiplying the area with the physical pixel dimensions.

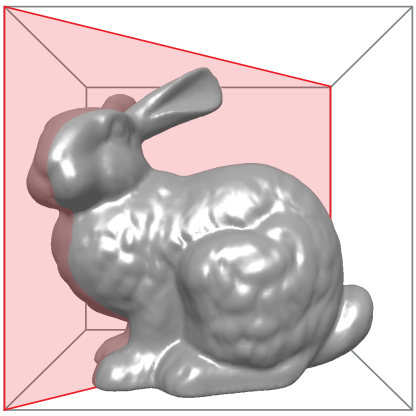


Figure 4.3: The highlighted quad describes the area which is used to approximate the object area.

The perceptual parameter *foveal coverage* F is defined as the *ratio* $F = \frac{A_{fc}}{A_{obj}}$ of the foveal area A_{fc} to the object area A_{obj} and therefore describes the percentage of the object area that fits in the foveal region. To meet a specific foveal coverage F, a scaling factor S has to be determined. To do so, the foveal coverage F_{orig} is first computed using the original scale of the object; the scaling factor S to reach the desired foveal coverage F is then given by $\sqrt{\frac{F}{F_{orig}}}$. Thus, the resulting scaling factor S is no longer a function of the largest side length of the AABB, it is based on the projected area with respect to the foveal coverage. In scenes containing multiple objects, the problem of unit scaling becomes more evident, as shown in Figure 4.5. In this example, the width-to-height-ratios of the two models differ severely; the bunny has an approximately cubic AABB, while the skeleton's AABB is cuboid-shaped. Although the objects are scaled to the same size, the projected (and therefore perceived areas) of the objects are significantly different. This difference influences the ability to recognize the individual objects, and the degree of difficulty in recognizing the objects would differ when using our obfuscation method. However, a scaling based on the foveal coverage balances the size of the objects so that their projected areas are equal.

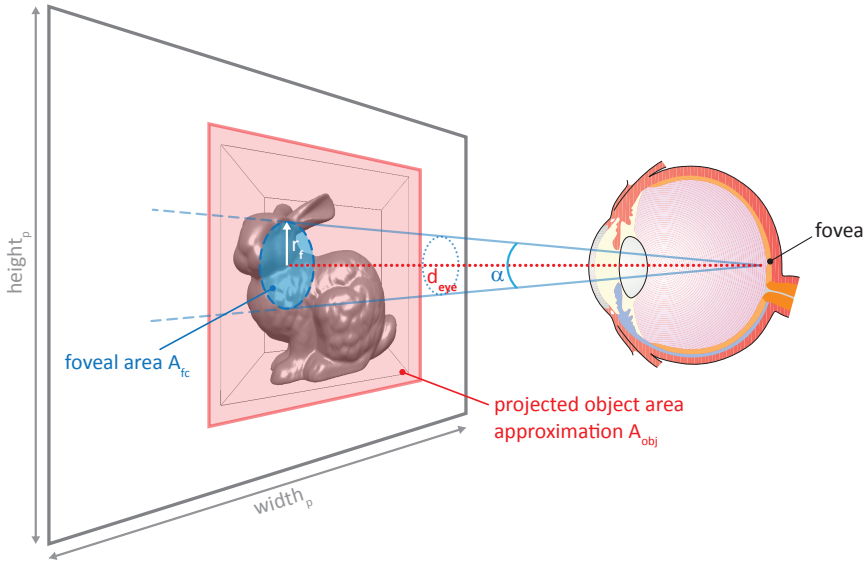


Figure 4.4: Foveal coverage - illustration of the observer and screen (with the physical dimensions $width_p$ and $height_p$). Outlined in blue, the conic model of the fovea and the resulting circular area A_{fc} on the screen can be seen. The orange region corresponds to the projected object area approximation A_{obj}. The foveal coverage describes the ratio of the foveal area A_{fc} to the object area A_{obj}. (figures adapted from [40])

4.1.2 Scene Grid

The next step in model processing is the alignment of the loaded objects to the scene so that all objects have a specific position on the screen. Scaling based on foveal coverage requires additional effort for the scene alignment, because once the objects are scaled to a common factor of foveal coverage, their actual scaling can differ significantly. To cope with varying dimensions of the individual AABBs we use a normalized screen-space alignment system.

The basic principle of this system is to equally distribute a set of points in screen space depending on the number of objects in the scene. For example, if the scene contains two objects, a single point is defined in each half of the screen. Next, a world-space position is determined for each point by back-projecting the screen-space coordinates to the x-y plane in world space. Each object is then translated to one of these world-space points. Using the steps described in this section, we created a scene where objects are normalized to a common perception-related scaling factor and distributed equally on the screen.

To integrate the concept of the reCAPTCHA [4] approach (see Section 2.11 for details), each scene requires at least two objects, one being the *unknown* object and the other one the *control* object. Beside the reCAPTCHA approach, the number of objects in a scenes depends on the design of the CAPTCHA challenge itself. For example, another form of metadata generation could be the selection of all upright oriented objects from a set of objects to determine the orientation of the objects. Therefore, the following sections are valid for an arbitrary number of objects and the reCAPTCHA equivalent is treated as a special instance of the general case.

(a) Unit-scaling (b) Scaling using foveal coverage

Figure 4.5: For scenes containing multiple objects, a scaling based on unit-scaling (a) produces significant differences between the perceived object areas. In contrast, our proposed scaling based on the foveal coverage (b) balances the size of the objects so that their perceived areas are equal.

4.1.3 Animation and Importance-based Orientation

Due to the assumption that models have unconstrained specifications regarding orientation and texture information, the visualization of the objects with a single image from a specific viewpoint leads to unreliable results when used in recognition tasks. Parts of the object that are crucial for recognition may not be visible from a static view. Assuming that models are upright, but their front- and back is not known, we decided to apply a continuous rotation transformation while maintaining a preselected point of view. This rotation enables to perceive multiple features

that are distributed across the object. When specifying the rotational axis, the main upright orientation of the object has to be maintained, otherwise, the uncommon orientation has negative effects on the object recognition. In our approach, we suggest a rotational axis which is similar but not equal to the model's y-axis. Selecting the y-axis as rotational axis is sub-optimal for rotationally symmetric objects (e.g. bottles), since the combination of animation and obfuscated rendering would not convey the impression of a symmetric shape.

In most cases, a slightly downwards oriented view provides an optimal perspective on the upright models. Due to the diversity of the models, especially flat objects (e.g. airplanes) do not reveal enough of their shape characteristics for the recognition otherwise. While the object rotation resolves the front and back sides issue, an appropriate slanting angle β_s has to be selected from a predefined interval to minimize recognition difficulties for such objects. At the same time, this requirement raises the question on how to determine the corresponding slanting angle β_s within a predefined interval. Considering the rotational transformations applied to each object, the corresponding angle β_s should stay constant along the entire animation to avoid an erratic motion of the object. To determine β_s, a minimization problem is defined based on the corresponding importance map information (see Section 4.2.1) [41, 47]. In short, the importance map highlights the crucial parts of each object at a specific point in time and from a certain point of view as a 2D texture. When defining the minimization problem, it has to be considered that the importance varies as the view on the object changes. Therefore, to determine the overall importance value for a certain angle β_s, each object has to be sampled from multiple directions. Based on the fact that the virtual viewpoint is static, the sampling is achieved by simulating the rotation process and aggregating the resulting importance maps. Hence the minimization problem is defined as

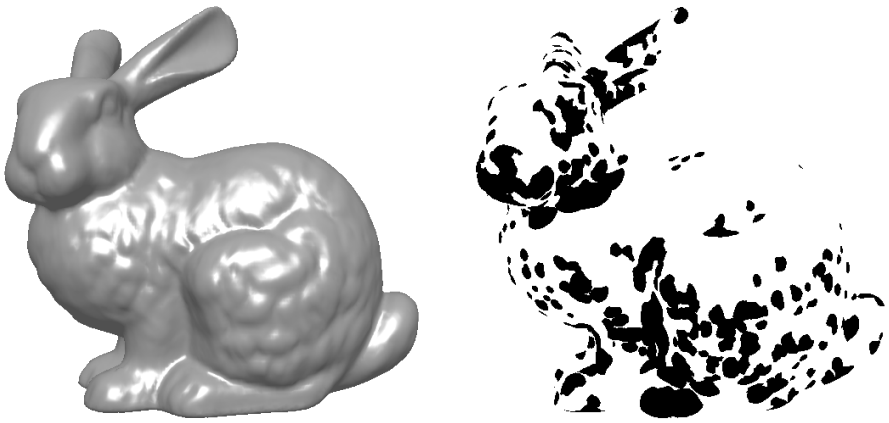
$$\min_{\beta_{s1},\ldots\beta_{sn}\in[-30\,^\circ,10\,^\circ]}((m * I_{max}) - \sum_{i=1}^{m} I(f_i(\beta_{sj}))) \tag{4.2}$$

where m is the number of sampled frames, I_{max} the maximal possible importance value in a single frame and $I(f_i(\beta_{sj}))$ the accumulated importance at the frame i using a slanting angle β_{sj}. Minimizing the function by using the method proposed by Nelder and Mead [50] results in a slanting angle where the importance over the whole animation is *maximal*. This angle remains constant as long as the viewpoint does not change and therefore has to be evaluated once for each object in an offline preprocessing step.

4.2 Object Obfuscation

The objective of the object obfuscation process is to extract an abstracted visualization which contains only those elements of the object that are most relevant for the recognition task. This abstraction, first proposed by Mitra et al. [47], results in a sparse representation of the object surface, including only important regions and neglecting the remaining parts. Regarding perceptual aspects, the sparse representation directly addresses the Gestalt principles (see Section 2.1.2), where especially the principle of closure enables us to fill in missing information to create a closed and stable impression of the object. Figure 4.6 illustrates the unobfuscated input model and the result when performing the object obfuscation steps.

As shown in Figure 4.1, the object obfuscation represents the *initial* step of the obfuscated rendering pipeline. A more detailed overview of the rendering loop is provided in Figure 4.17. In the subsequent sections, the individual parts of the pipeline are described in detail.



(a) (b)

Figure 4.6: The difference between source model (a) and the result of the object obfuscation (b).

4.2.1 Importance Map

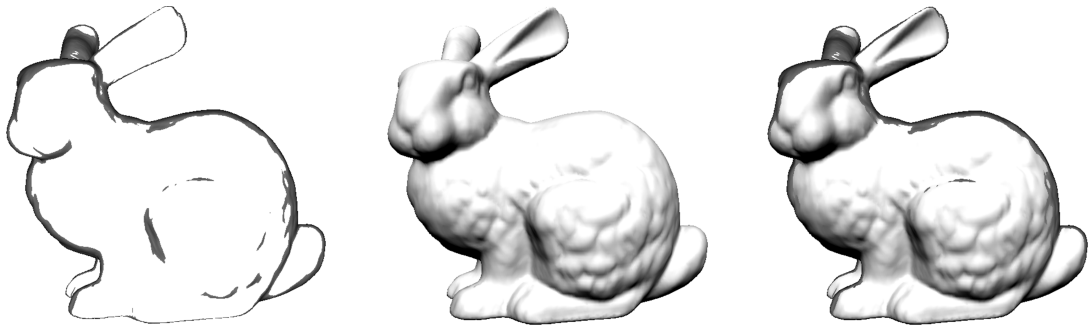
When using random subsets of the object surface for the sparse representation, object detection is made more difficult not only for computer vision algorithms, but also for humans. Consequently, the first step in the obfuscation process is to determine regions on the surface that contain meaningful characteristics of the object shape. Potential input data for determining these features are silhouettes and the shading of the surface.

We use the concept of an *importance map* introduced by Mitra et al. [47], which is a single-channel 2D texture defining the importance for each point of the projected object surface from a specific point of view. Values in the importance map are in the interval $[0, 1]$, where 1 corresponds to maximal importance and 0 to unimportant parts. As stated by Mitra et al., the formulation of the importance map includes information about the surface geometry, the viewpoint and the light position. In more detail, the importance map is composed by the following two individual maps:

1. **Silhouette Map**
   We aim to capture prominent features within the object boundaries which have similar characteristics as edge- and silhouette-features, as these are important for object recognition. This can be achieved by computing the cosine of the angle between the view and the normal vector of each fragment. This value is maximal if the view and the normal vector are orthogonal to each other and decreases as the absolute angle decreases. To limit the size of the silhouette, the minimal angle is clamped to a predefined limit. Evaluating the silhouette for each fragment of the object results in a 2D *silhouette map* (see Figure 4.7a).

2. **Shading Map**
   To include less pronounced geometrical details of the object, an inverse version of standard diffuse shading is used to highlight other relevant parts. Each fragment is set to $1 - (n \cdot l)$, where n is the normalized normal vector and l the normalized light vector. This calculation of the inverse diffuse shading for each fragment generates the 2D *shading map* (see Figure 4.7b). Values in the shading map are in the interval of $[0, 1]$ due to the use of backface culling. When comparing the shading and silhouette map, the silhouette map does not capture any lighting or depth information. In contrast to that, shading corresponds to a monocular depth cue and therefore describes the depth of the object surface.

Once the silhouette and shading map are generated, the importance map is derived by using a pixel-wise maximum from the two maps. Figure 4.7 shows the individual maps and the final product, the *importance map*.



(a) (b) (c)

Figure 4.7: The silhouette map (a) and the shading map (b) are combined to a single importance map (c) by using their pixel-wise maximum.

Regarding the implementation in the rendering pipeline, the importance map is rendered in the first pass, along with the Phong shaded version of the objects (see Figure 4.8). Due to the fact that the generation of the importance map requires only fragment operations and some trivial vertex operations, the method described can be implemented in a fragment shader on the GPU.

4.2.2 Importance Sampling

To establish a sparse representation of the object containing only the important regions, the surface of the object will be partially textured using a circular texture, called *splat* (see Section 4.2.3). A purely random distribution of these splats across the surface would imply that also non-salient regions get textured. Therefore, the importance map is used as a basis for the distribution of the samples, a process known as *importance sampling*. Ostromoukhov et al. [51] specify the main problem as follows: An importance density I is given over a domain D in form a 2D array, containing normalized discrete values with $0 \leq I(x, y) \leq 1 \; \forall (x, y) \in D$. The goal is to find a discrete set of sample points where the number of samples per unit area is directly proportional

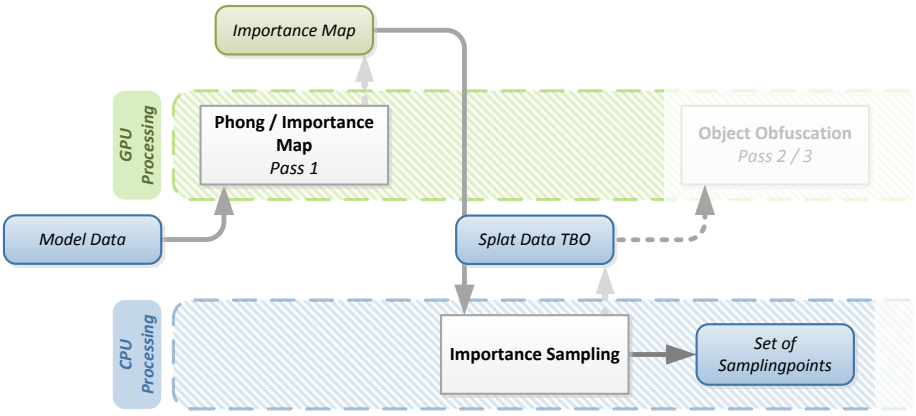
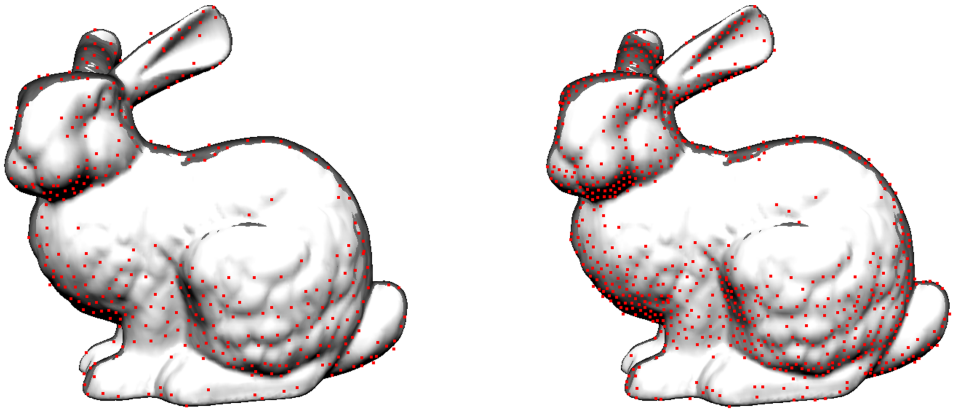


Figure 4.8: A subpart of the obfuscation pipeline describing the generation of the importance map and the subsequent importance sampling. The reader is also referred to Figure 4.17 for a complete view of the rendering pipeline.

to the corresponding importance density I. Ostromoukhov et al. [51] introduced an approach for 2D importance sampling based on the well-known Penrose tiling for subdividing a given importance map. Using the Penrose tiling in a hierarchical manner, Ostromoukhov et al. are able to provide a multi-level subdivision of a given importance map while maintaining a Blue Noise [51] spectral characteristic of the sample points.

As in the work of Mitra et al. [47], we apply this sampling method to the importance map (see Figure 4.9). The result is a dense sampling of the silhouettes and regions where the inverse diffuse shading is maximal. In addition, the scale of the sampling grid can be controlled by a parameter provided by the sampling method. If the scene contains multiple objects, the performance of the sampling method can be further increased by limiting the sampling space to a window derived from the projected AABB of the objects on the importance map. Taking into account that the constrained sampling space can vary significantly (see Section 4.1), the area-related ratio between the sampling window and the whole importance map is used for renormalizing the scaling factor of the sampling method, establishing an equally distributed sampling which is independent of the projected object area.

In the design of the rendering pipeline, the importance sampling is processed on the CPU in a multithreaded manner. Due to the animation of the object, the generation of the importance map as well as the importance sampling are processed for each frame of the animation. The resulting discrete set of 2D sample points is the basis for all further stages of the system (see Figures 4.8 and 4.17).



(a) (b)

Figure 4.9: Visualization of resulting sets of sampling points with different scaling factors ((a) 300 and (b) 600).

4.2.3 Object Splatting

To generate a new object representation where surface information is sparsely distributed across the object, the object is re-textured using the previously mentioned *splats*. More specifically, the discrete set of samples generated from the importance map mark the center of each splat. Due to the fact that importance sampling is a 2D process, the center points have to be back-projected to 3D, e.g. by casting a ray from the eye to through the 2D point and intersecting that ray with the object geometry, or by using a position buffer. As stated at the beginning of this chapter, we made no constraints and assumptions regarding the surface complexity and geometrical properties of the source objects. A robust texturing or decal method is essential to avoid artifacts when applying splats to surfaces with high-frequency details.

The texturing method [56] used by Mitra et al. [47] has major shortcomings in this regard, since the method cannot reliably handle surfaces with high-frequency peaks. In such cases the decals get significantly distorted along these features. Schmidt et al. compensate this effect by avoiding to texture the peaks, which would lead to unexpected holes in our case. Therefore, we combine the GPU-based volumetric approach by Engel [17] and deferred decal technique proposed by Lengyel [42] to handle decals on non-trivial geometry. In general, decals are 2D textures which are applied to a specific region of the surface. The main idea of the method introduced by Engel is to project a volumetric decal onto the surface using deferred rendering, so the whole decal rendering is done using fragment operations.

In our case, decals are just the previously mentioned splats. The number and position of the splats is determined by the importance sampling process. These (screen-space) points are used as texture coordinates for a lookup in a position buffer to retrieve the corresponding world-space position on the model surface. The point P_{center} describes the center of the splat. According to the volumetric decal approach, we define a splat as a black spherical volume V_s with a given (world-space) radius r_{splat} and centered at P_{center} (see Figure 4.10).

Due to the fact that the rendering of splats requires only fragment operations, first, the fragment processor has to be invoked for affected fragments by rendering a conservative dummy bounding volume V_d with a radius $r_d > r_{splat}$ (e.g. sphere or cube) centered at P_{center}. For each fragment, we need to find out whether the corresponding pixel belongs to an object and whether that object is in the splat radius at the center of the pixel. To this end, the world-space position P of the pixel is looked up in a position buffer. The distance from this world space position P to P_{center} is evaluated to test whether the fragment is inside the splat volume V_s. If and only if this distance is less than r_{splat}, the fragment is colored. The resulting textured surface region corresponds to the intersection of the object surface with the spherical splat.

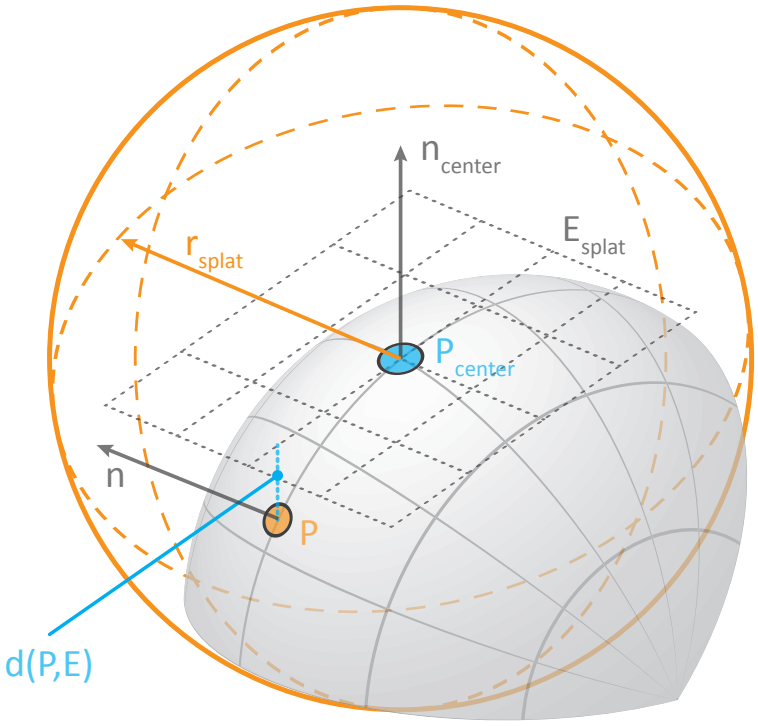


Figure 4.10: Schematic illustration of the volumetric decal approach – The spherical splat volume (highlighted in orange) with radius r_{splat} is centered at P_{center} on the object surface. E_{splat} is the plane defined by P_{center} and the corresponding normal $\vec{n}_{center}$. $d(P, E)$ denotes the Hesse distance of the point P to the plane E.

The method described so far, however, does not take the surface characteristics of the object into account. Therefore, we modify the calculation of the distance between P and P_{center} by using the method proposed by Lengyel [42]. The idea of this approach is to create a wrapping effect which is based on the surface slope and the normal distance of the fragment to the splat center. Hence, this wrapping effect acts as an additional cue to convey the local surface characteristics. To determine the surface slope, the required normal vectors $\vec{n}_{center}$ of P_{center} and $\vec{n}$ of

P are determined using a normal buffer; then, the length of the vector $abs(\vec{n}_{center} - \vec{n})$ is used as an approximation of the *surface slope*. As illustrated in Figure 4.10, the point P_{center} and the corresponding normal $\vec{n}_{center}$ define the plane E_{splat}. The normal distance $d(P, E)$ of a point P to a plane E is calculated by using the *Hesse normal form*, which is defined as follows:

$$d(P, E) = \vec{n} \cdot p + n_0 \tag{4.3}$$

where $\vec{n}$ denotes the normal vector of the plane E, p corresponds to the tested point and n_0 to the offset of the plane from the origin. As the origin of our reference frame is P_{center}, the offset n_0 is 0, hence the *signed* distance $d(P, E_{splat})$ is calculated as $\vec{n}_{center} \cdot P$. Weighting the previously determined surface slope with $d(P, E_{splat})$ results in an *offset-value*. This offset-value is subtracted from the distance of P to P_{center} before testing against r_{splat}. Therefore, the offset-value changes the size of splat volume implicitly and in further consequence, deforms the shape of the cross section of the object surface with the spherical splat according to the local surface characteristics.

In the rendering process itself, the method described can be completely implemented in the fragment shader. The required normal and position buffers are generated when rendering the Phong-shaded visualization of the objects (see Figure 4.11). In each frame, the model is rendered with the method described to generate a single channel texture, the obfuscated object mask. (see Figure 4.6b for an exemplary result of one object). Note that only the textured regions of the objects remain while all other parts are now handled as being *transparent*.

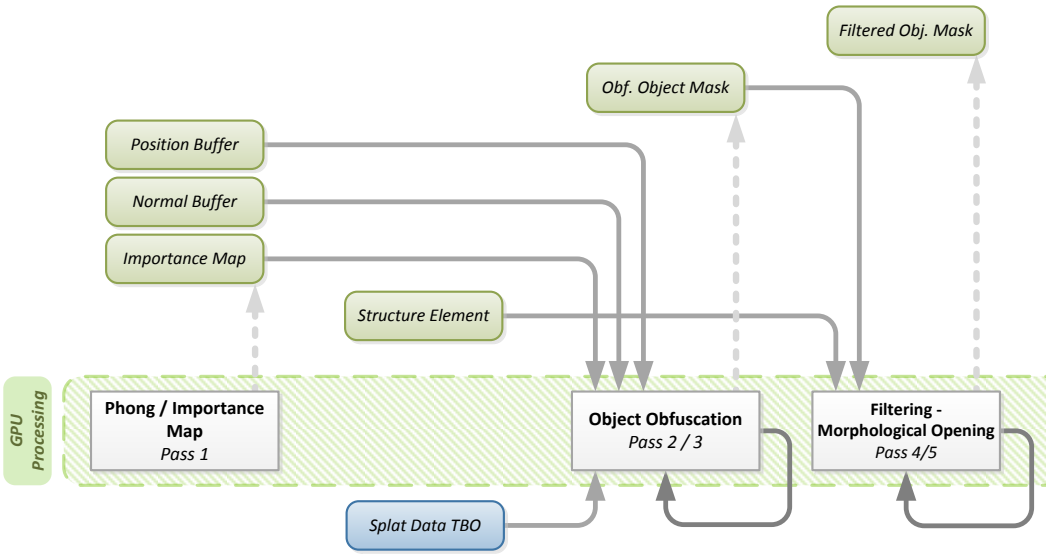


Figure 4.11: Illustration of the relevant parts of the rendering pipeline which are responsible for the object obfuscation. The reader is also referred to Figure 4.17 for a complete view of the rendering pipeline.

At this point in the processing pipeline, the object is textured at previously determined important regions. The previously mentioned blue noise characteristic of the importance-sampling method implies a regular spacing of the splat texture on well-formed surfaces where the shading map is relatively homogeneous. To avoid exploitation of these regular patterns, random *jittering* is applied to the sample points by shifting their coordinates on the tangential plane of the underlying surface point and reprojecting the new coordinates back to the surface. In addition, the degree of jittering for a specific sample point is weighted by the corresponding importance value from the shading map, so that the jittering increases with increasing importance.

By construction, the importance distribution is more dense along the silhouette, so that these regions have a higher number of splats per unit area. Splats on the silhouette tend to overlap each other due to the potentially higher magnitude of the surface gradient, which directly influences the wrapping effect of splats. Since those connected splats create edges along the silhouette, further processing steps are focused on breaking the continuities of these implicitly produced edges. The main idea is that after jittering of the sample points, the neighborhood of splats located on the silhouette is scanned for overlapping regions. This concept is realized in a 2-pass strategy (see Figure 4.11); in the first pass, the splats are rendered using the method described so far. In the second pass, the texturing is repeated and splats are filtered based on the obfuscated object mask M resulting from the first pass and the importance map. Before rendering a splat, an area of 16×16 pixels is sampled around the center point of the splat on the texture M on the GPU. To detect overlapping splat segments in M, the splats are additively blended in the first pass. Based on the accumulated sum of the samples, the current splat is marked for further filtering. Alternatively, if the surrounding sampling is too dense in important regions, the splat is not rendered in the second pass to break the continuity. Since the second pass only requires a single-channel texture, an additional channel is utilized to mark splats for the following filtering.

Morphological Filtering

As suggested in the work of Mitra et al. [47], we apply morphological operations to the splat segments marked for further filtering. Figure 4.11 depicts that this step is implemented by the next two passes of our pipeline. Mathematical morphology is an image processing method that mainly operates on shapes within images. For a given binary image, pixels with value 1 belong to the object, while pixels with a value of 0 represent the background. In the following, the concept of binary morphology is described. Based on the binary definition of the source image, an additional structuring element is introduced, which is also a binary image containing a predefined shape. The main idea of morphological operations is to test the structuring element (e.g. a circle or box shape) against the whole image by aligning the structuring element at each pixel. [28]

In a more formal manner, the given image A and the structuring element B are sets in an Euclidean N-space, denoted as E^N, with corresponding elements $a = (a_1, \ldots, a_N)$ and $b = (b_1, \ldots, b_N)$. When this principle is applied to discrete images, the Euclidean spaces correspond to 2D planes, while A and B consist of a number of non-overlapping square regions – the pixels. In the first basic morphological operation, the *dilation* of A by B corresponds to the set of all possible sums of the element vectors of A and B.

Haralick et al. [28] specify the dilation as follows: Let A and B be subsets of E^N. The dilation of A by B is denoted by $A \oplus B$ and is defined by

$$A \oplus B = \left\{ c \in E^N \mid c = a + b \text{ for some } a \in A \text{ and } b \in B \right\} \tag{4.4}$$

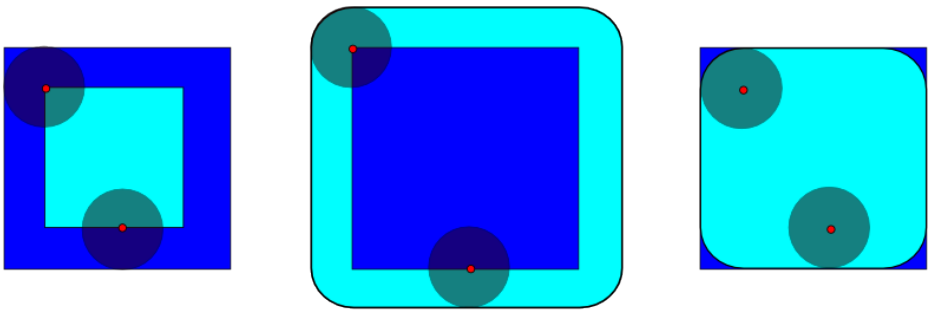
The morpholgical dual to dilation is known as *erosion*, which corresponds to the element-wise subtraction, defined as

$$A \ominus B = \left\{ x \in E^N \mid x + b \in A \text{ for every } b \in B \right\} \tag{4.5}$$

The basic operators are combined to create new operators like the *opening* transformation. The opening operator smooths the input as it removes elements smaller than the structuring element, including small islands and spikes. In terms of the previously specified erode and dilation operators, the opening operator is defined as

$$A \circ B = (A \ominus B) \oplus B \tag{4.6}$$

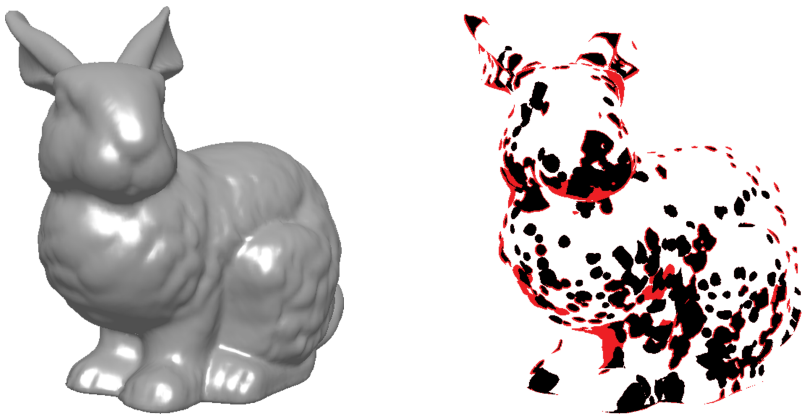
In our context, we use the opening operator to eliminate thin lines and splats which are smaller than the structuring element to deform splats and to open thin connections between individual splats to break their continuity. Due to the fact that the opening operator is the combination of two operators, the integration of this filter process in the rendering pipeline is achieved by using two passes (see Figure 4.11). The filtering uses the obfuscated object mask of the previous pass as input. Both passes require only fragment operations, hence, in each pass a fullscreen quad is rendered and the resulting output-texture of the predecessor is used as input. In addition, both operators use the same circular-shaped structuring element stored in a binary texture. For each fragment, the center of the structuring element is aligned to the fragment. Next, a pixel-wise comparison of the structure element and the „underlying“ texture is done. The pixels inside the structure element are accessed by precalculated offsets which are stored inside an array.



(a) Erosion (b) Dilation (c) Opening

Figure 4.12: Morphological Operators – Each operator (a-c) is applied to a dark-blue rectangle using a disk-shaped structure element. The light-blue area corresponds to the result. (figures from [35–37])

From the formal definition of the erode operator follows that the fragment is only colored if the structuring element completely fits into the tested region, otherwise, the fragment is set to transparent. Therefore, the erode operator (see Figure 4.12a) reduces the textured regions in size and removes segments which are smaller than the structuring element. In contrast, when using the dilation operator, a fragment is colored if at least one pixel in the tested region coincides with the structuring element (see Figure 4.12b). Hence, this operator extends a region by the structuring element. When filtering marked splats along the silhouette with the opening operator (see Figure 4.12c), thin lines and small speckles are removed, while thin connections between individual splats are opened. The filtered result of the texturing step is shown in Figure 4.13b.



(a) (b)

Figure 4.13: Resulting object obfuscation (b) of the object (a) with applied jittering, splat and morphological filtering; removed pixels are marked red.

4.3 Background Generation

From the schematic view of the obfuscation pipeline, shown in Figure 4.1, it is evident that the object processing is only the first step towards the obfuscation of the whole scene. When simply using the result of the object obfuscation (see Section 4.2), position and object boundaries are obviously detectable, providing a coarse approximation of the object shape. Assuming that the locations of the objects in the scene are unknown, an additional aim of the proposed method is that the geometry and location of the contained objects cannot be recovered from windows within a frame. However, as mentioned before, a blank background is not a practical solution to meet this requirement. As a consequence, we describe in the following the processing steps for synthesizing the background in an equivalent manner to the object obfuscation.

As proposed by Mitra et al. [47], the basic principle of generating the background is to introduce clutter patterns as similar to the object texture as possible. The basis for this background clutter is the result of the object obfuscation itself. Parts from the object of different sizes are selected, copied, transformed, and pasted back into the background. Repeating those steps suf-

ficiently often covers the background with splat patterns originated from the object. In detail, a rectangular window of random size is used to select a predefined number of points from the corresponding discrete set of sampling points (see Section 4.2.1). Arbitrarily selecting the samples within the window is the first step in disturbing the object pattern for the background. Once the set of samples for the window is determined, the window is transformed by applying a random rotation and translation in screen space, making sure that these transformations are limited to the viewport (see Figure 4.14). To further prevent the production of a series of exact partial copies of the object, sample points are individually distorted per window in screen space.

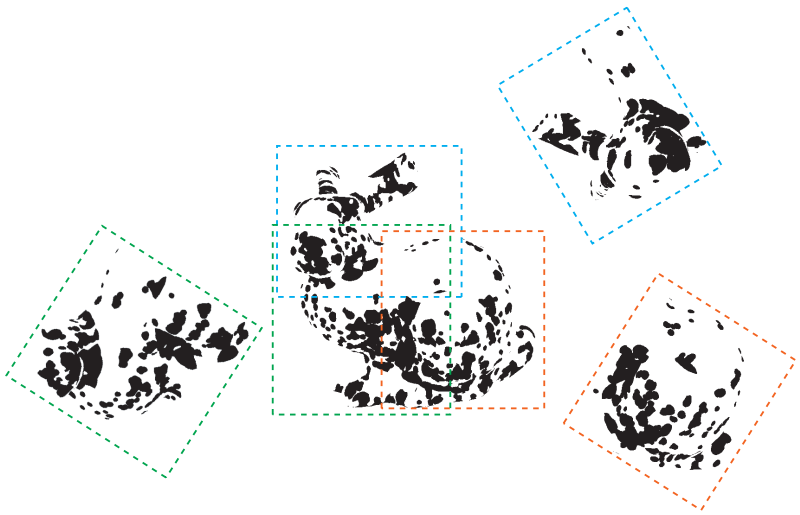


Figure 4.14: The principle of the background generation is to select windows (dashed selections) from the object region, then copy the window and finally transform the splats into the background.

Considering that the final result is an animated sequence of consecutive frames, the obfuscated visualization of the object alone shows a high degree of coherence in the temporal dimension. This property has to be carried over to the background clutter generation to preserve a strong masking effect. Synthesizing the background from scratch in each frame emphasizes the perception of the object; the consequential high-frequency noise characteristic of the background, however, produces a mismatch between the temporal coherence of the fore- and background (c.f. Mitra et al. [47]). To reduce this discrepancy, an additional elementary property for windows is introduced in our approach – the *lifetime*. The lifetime parameter allows a dynamic specification of a temporal frame for individual windows, directly influencing the window coherence by defining how long a certain window is rendered. Using the properties of windows described so far leads to a more detailed definition of two different types of windows:

- **Static Window**
  The main idea behind a static window is to select a rectangular region of the screen (so that the projected AABB of the object is at least partly covered) and to transform its obfuscated content into the background. This behavior is similar to a copy-paste process

which is repeated for each frame for the lifetime of the window. In detail, a different subset of the object samples is selected within a window in each frame and rendered according to the method described in Section 4.2.3. The splats are transformed into the background by using an additional screen-space transformation (random rotation and translation). The intended function of this type is to fill the background with clutter.

- **Sticky Window**
  In contrast to static windows, these windows are focused on replicating the object motion in the background. The motion characteristics of the object are captured by copying a selected region of the obfuscated object surface into the background. For this, a set of object samples is selected and stored within a window during its initialization. Compared to static windows, the set of samples for sticky windows remains fixed for the whole lifetime, while for static windows the corresponding set varies over time. As the object rotates during the animation, we want to reproduce a similar motion of the stored set of samples at a different location of the screen with a different orientation of the rotation axis. Therefore, the world-space coordinates of the selected samples have to be determined using the position buffer. Once the world-space position is determined, the per-frame object animation is applied to all selected samples. At this point, the object samples have the same motion characteristics as their corresponding surface point. To reuse the method for splat rendering (see Section 4.2.3), the sample points have to be projected back into screen space. Similar to static windows, an additional screen-space transformation (random rotation and translation) is used to create a random position and motion direction in the background.

  In general, the spatial coherence of sticky windows arises from the fact that the selected splats are located close to each other (within the window) and the number of splats remains the same over the window lifetime. Replicating the object motion with the samples provides the desired temporal coherence. Due to the fact that the set of the selected object samples is fixed over the window lifetime, the inter-sample (world-space) spacing also remains constant in this time interval. To avoid the generation of discriminative pattern constellations, the position of each sample is slightly iteratively jittered.

The combination of both window types introduces spatial and temporal characteristics of the obfuscated object in the background. A practical reason for increasing the coherence in the background are potential exploits based on Scale-Invariant Feature Transform (SIFT) features. The approach proposed by Lowe [45] describes a robust local feature descriptor which is invariant to scaling, affine transformations, partial occlusions and noise in the image. Without any coherence in the background, SIFT features can track prominent features of the object over multiple frames. By increasing coherence in the background, the background windows act as attractor for SIFT features. As shown in Figure 4.15, the definition and management of the windows in our rendering pipeline is done on the CPU side. By construction, windows correspond to a subset of the object splats, thus to render the splats within a window we reuse the volumetric decal method. Moreover, to distribute the window in the background, an individual screen-space transformation is linked with each window (see also Section 4.7 for implementation details). For sticky windows, an additional transformation matrix containing the object animation is required.

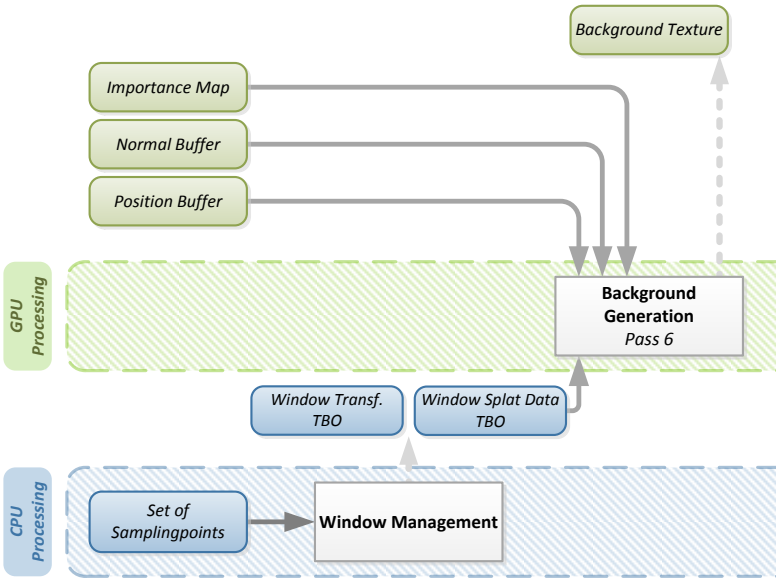


Figure 4.15: Illustration of the background generation in the rendering pipeline, outlining the required resources. The reader is also referred to Figure 4.17 for a complete view of the rendering pipeline.

Furthermore, a monochromatic color property is assigned to each of the background windows. The colors are selected from a limited pool of colors which differ in brightness. The individual color property is constant over a dynamic number of frames before another color is randomly assigned from the pool. With each frame, all windows (all splats) are rendered from scratch and are blended together into one texture.

4.4 Merge Object and Background

The last step of the rendering pipeline is the combination of the images resulting from the object obfuscation and background generation processes. The complete rendering pipeline is illustrated in Figure 4.17. Up to this point, the obfuscated object texture does not contain any color information, hence it can be seen as a mask. Colors from the previously mentioned color pool are associated with the mask of each object. The colors of the objects are changed, as for the background windows, after a dynamic number of frames. However, this pure image-based step only requires to render a fullscreen quad in which the obfuscated object mask is blended onto the background texture. Note that all pixels of the obfuscated object texture that are not covered by splats are handled as transparent, hence, these pixels are colored by background texture. Figure 4.16 shows the result of the proposed synthesis process.

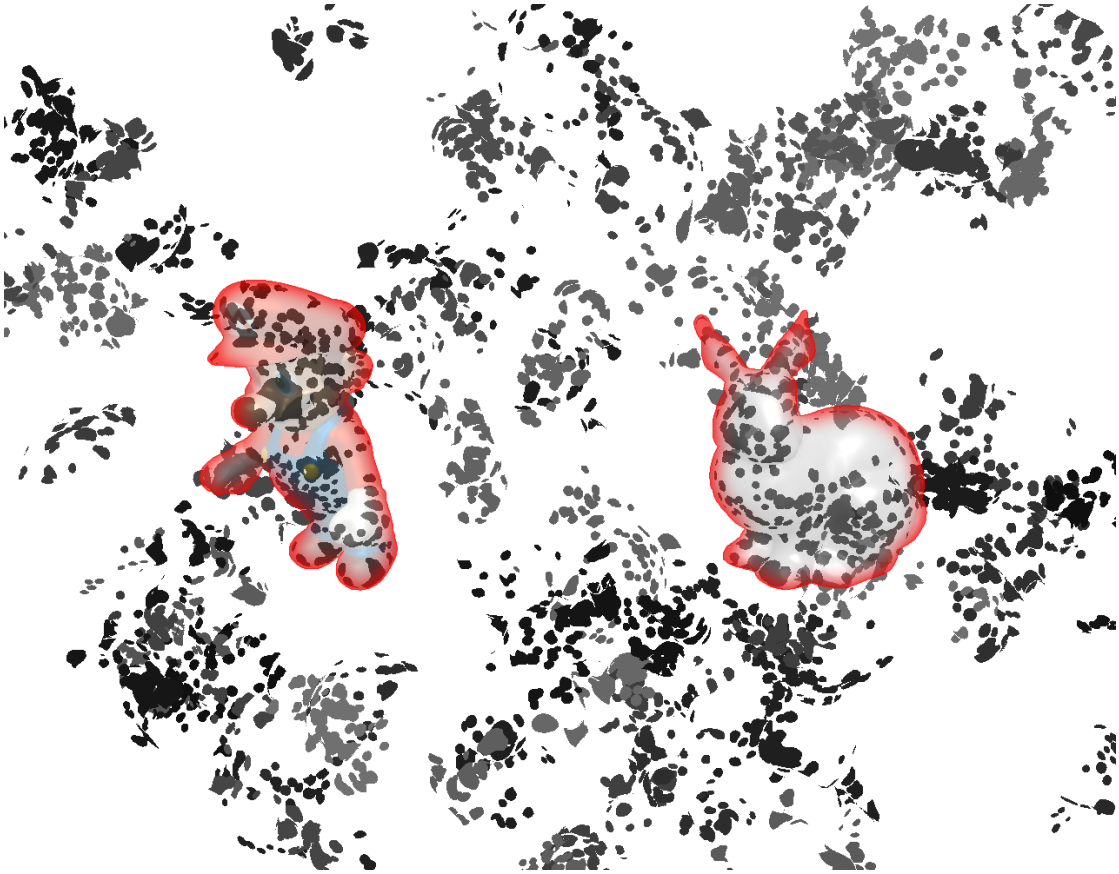


Figure 4.16: Combination of the obfuscated objects with the background (objects are outlined for illustrational purposes).

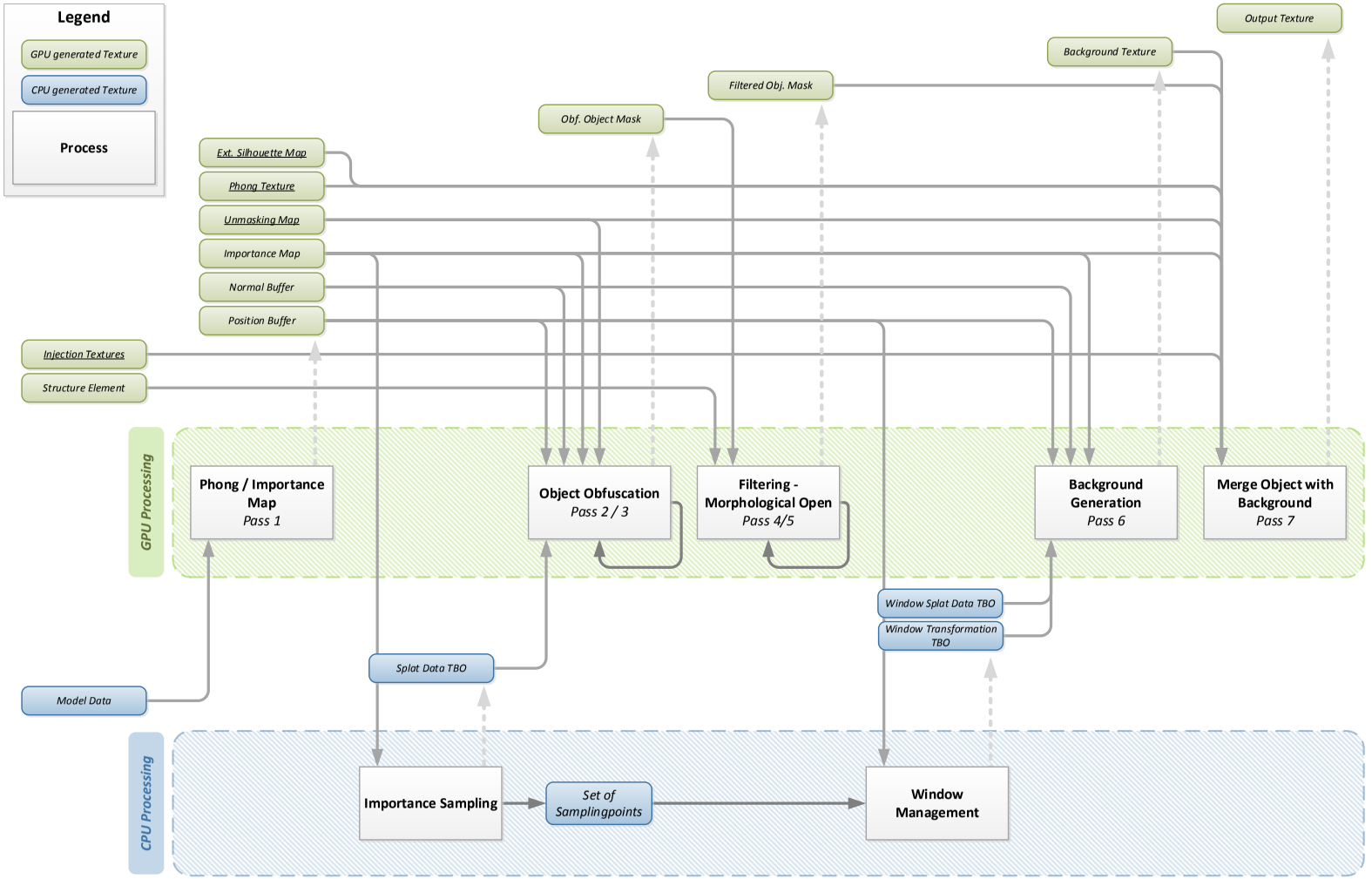


Figure 4.17: Rendering pipeline – This figure outlines the main processing steps and their implementation-environment (CPU respectively GPU). For each step the required input textures (inbound arrows) and the resulting textures are illustrated (outbound arrows). Resources, which are related to experimental features, are marked with underlined names.

4.5 Experimental Features

In this section, features of the obfuscation pipeline with experimental character are presented. Their experimental character comes from the fact that their evaluation is beyond our available resources and the scope of this thesis. Therefore, they are not tested via a user study. Moreover, there is a possibility of strong dependency on the actual objects and complex behavior depending on the selected parameter associated with these methods.

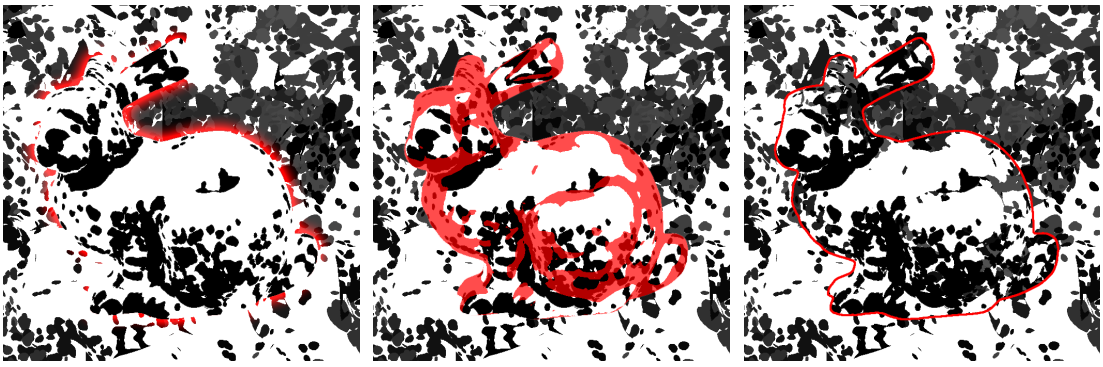
4.5.1 Partial Blending

The aim of this section is to enhance the object visibility for a human despite a high degree of clutter in the background, while keeping the detectability for automatic algorithms equally difficult. To make the recognition easier, untextured regions of the object could be made opaque instead of transparent as described in Section 4.4. On the downside, this can lead to a stencil-like effect when the background density is quite high compared to the textured area of the object. As a consequence of this effect, implicit edges are formed between sparsely textured areas of the object and the background (see Figure 4.18a). These edges could be exploited by edge detectors to find and subsequently determine the objects in the scene. Another way to enhance the object visibility could be achieved by reducing the rendered windows and/or splats next to the object. This systematic reduction of the background clutter results in a halo effect around the object, which becomes quite noticeable in the approach of Mitra et al. [47] when using animated scenes.

To overcome these problems, we limit the blending with the background to the silhouette of the object, while the remaining regions of the obfuscated remain opaque. Therefore, we propose an extended silhouette map as a mask where untextured object regions are handled as transparent and thus a blending with the background is explicitly allowed. To be more specific, within the first pass of the rendering pipeline, an additional extended silhouette map is computed without any clamping of the angle between the view and the normal vector. The resulting map is then used in the blending of the obfuscated object mask and the background texture in the seventh pass (see Figure 4.17). Using the extended silhouette map directly as a mask produces edges within the object boundary, which have the same form as the object shape. To overcome this issue, the inner contour line has to be perturbed so that their shape is no longer similar to the object characteristics. From the definition of the silhouette map follows that its maximum is located on the object boundaries (normal and view vector are orthogonal to each other) and decreases according to the object surface geometry. The inner shape of the silhouette is perturbed by clamping the lower bound with a dynamic threshold. The variable characteristic of the threshold is enabled by using the GPU-based noise function proposed by Gustavson [26]. Figure 4.18 describes the result of using a variable threshold, which generates a dynamic inner shape.

4.5.2 Color and Texture Features

The obfuscation pipeline as described so far reveals the salient geometry of the object, ignoring any material-related features. Especially when surface details of models are simplified by using a coarse geometry representation (e.g. buildings), offering only shape and geometric information,



(a) (b) (c)

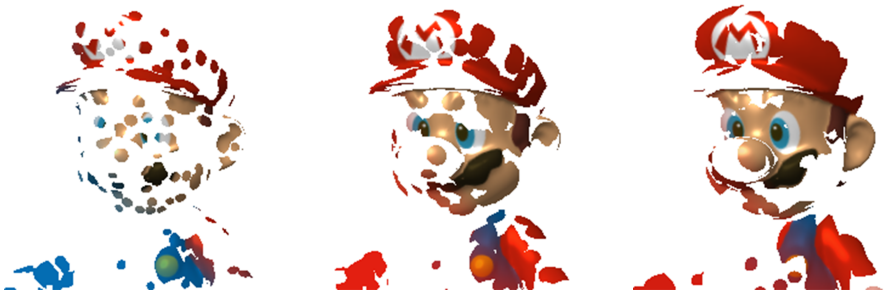
Figure 4.18: Figure (a) illustrates the problem of indirectly produced edges (marked red) if the background is dense while untextured areas of the object are not transparent. Using our dynamic mask shown in Figure (b), based on increased object silhouette, allows to pass the background over object boundaries, while main untextured areas of the object remain white. Figure (c) shows the result of blending the object with the background using the mask marked in (b) (object outlined).

the adequate recognition of the object becomes difficult. If the collection of models contains objects with similar geometrical features without any context information (such as relative size, model category, etc.), a high degree of ambiguity regarding the meaning of the object arises. Therefore, this section proposes the first steps to introduce additional color and texture features in the obfuscated scene representation. In some cases, where the color and texture information is crucial to recognize and describe an object, this enables better recognition. The two well-known models Mario and Luigi from the game series Super Mario Bros for example differ only in their texture.

First, relevant information for the integration of color features in the obfuscation pipeline is extracted when the model data is loaded. In this early stage of processing, texture and material information is collected. Assuming that the model has associated texture data, the first step is to constrain the used color domain to prevent a color noise characteristic. Dominant colors of each texture are extracted by determining normalized histograms based on the Hue Saturation Value (HSV) color model, calculating the average histogram per model and finally ordering the colors by occurrence. These colors are utilized as reference colors for window and object coloring. The combination with random colors or dominant colors of other models is possible to avoid exploits using the color distribution as signatures for objects. From the set of reference colors, one color is randomly selected and used to build a pool of jittered colors. Elements of this pool are generated by distorting the reference color slightly in hue and value. The object, as well as the background windows, are colored by selecting a random color from this pool (refer to the blending step in the pipeline described in Section 4.4). After a dynamically defined time interval, the reference color is switched. Within the time interval, additional jittering intervals are introduced, avoiding constant colors of segments over the whole interval and to replicate the

coloring behavior of the object. Optionally, to emphasize the fundamental task of recognizing two objects in the scene, each of the two objects has its own color domain.

Revealing texture information on the object is realized by limiting the information provided to small regions of the object surface. In detail, the idea is to use a discrete set of splats on the object as a mask for texture rendering. As we defined in Section 4.2, the splats are located along regions recognized as important in a geometric sense. For large flat areas the geometry is irrelevant but the texture could be highly relevant. To determine the visually salient regions of a textured object, more sophisticated methods have to be used (perhaps future work). It is assumed that revealing texture and surface details in those regions reveals relevant features to further support the object recognition. By locating a specialized attenuated light source L_S next to the object, splats in the radius of the light are marked for showing the underlying surface texture – thus, unmasking the object surface. Rotating the light source around the object with varying orbits unmasks different parts of the object surface, while limiting the effect at a specific point of time to a local region on the object – therefore, the effect is described as *local unmasking*. Introducing texture details on unsplatted object regions is achieved by modulating the importance map with the light source to induce the generation of sample points in the affected areas. Moreover, the inter-splat distance between splats can be controlled by weighting the splat size with the influence of the light, which corresponds to the distance to L_S (see Figure 4.19).

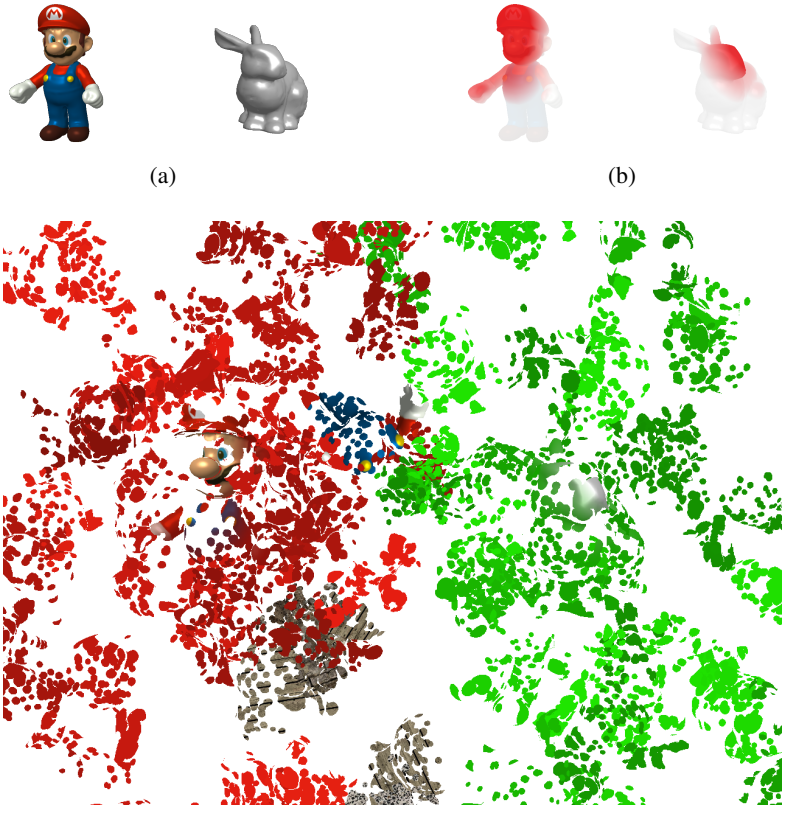


(a) (b) (c)

Figure 4.19: Local unmasking without increasing splat size (a) and with different degrees of weighted splat size magnification (b-c).

From the implementation perspective, the first pass of the rendering pipeline is extended by taking the rotating attenuated light source L_S into account. The result is an additional unmasking map (see Figure 4.20b) which is utilized to mark the splats for unmasking. In addition, the modulation of the importance map with the unmasking map is also processed during the first pass. As already mentioned, the unmasking map is used during the object obfuscation to increase the size of the splats. Before rendering the splat in the fragment shader, the influence of L_S is determined at the splat center in the vertex shader. This influence is then used to change the corresponding splat size. When blending the obfuscated object mask with the background in the last step of the pipeline, the unmasking map defines which parts of the obfuscated object mask are colored with the corresponding texture.

So far, the texture data is limited to the object itself, implying possible high-color frequencies in the object area. To create similar characteristics in the background, an additional window type is introduced, the *injection window*. Injection windows are subtypes of static windows, with the color property being extended by a texture. Procedural textures, arbitrary textures from a repository or shaded surfaces of the actual or other objects are possible texture sources for injection windows. By definition, unmasked areas are restricted to the object regions, the reoccurrence of texture is a potential information to infer the actual object location. To simulate this behavior of texture-reoccurrence in the background, a number injection points is allocated across the scene, excluding the object areas. The occurrence of injection windows is restricted to positions next to the injection points. Figure 4.20 shows the final composition of the coloring, unmasking and usage of injection windows.



(a) (b)

(c)

Figure 4.20: For given models (a) unmasking areas are determined (b) by using orbital moving attenuated light sources around the objects. Figure (c) shows the resulting obfuscated scene using the local unmasking approach with additional injection windows (textured windows).

4.6 Contributions Compared to Related Work

As the approach proposed by Mitra et al. [47] is the basis of our work, this section outlines the main contributions and differences:

- It is assumed that the perceived size of the object influences the ability to recognize the obfuscated representation of the objects. Therefore, the models are rescaled to a common perception-related scaling factor which is based on the projected area of the individual objects and the area covered by the foveal vision – the foveal coverage.
- In scenes containing multiple objects, rescaling the objects to a common foveal coverage results in quite different scaling factors and dimensions for each object. To create a well-structured scene, we use a normalized screen-space alignment system.
- As the 3D databases contain primarily static models, continuous object transformations are used to create an animated scene. This animation enables the user to perceive the salient features of the object that could be distributed across the whole object geometry.
- For the object obfuscation, we adapted the main principle of Mitra et al.. To enhance the performance of the importance sampling task, we limit the sampling to the projected AABB of each object, which allows to process the objects in parallel. For the rendering of the splats, we use the concept of volumetric splats, which combines the deferred decal technique proposed by Lengyel [42] and the GPU-based volumetric approach by Engel [17]. This approach enables us to handle splats on non-trivial geometry and provides an interactive visualization of the obfuscated scene. In contrast, Mitra et al. uses the texturing approach proposed by Schmidt et al. [56] to apply 2D circular shaped textures (splats) onto the object surface.
- Due to the fact that the scenes are animated, we extended the idea of background generation of Mitra et al. by introducing static and sticky windows. The aim of both window types is to balance the spatial and temporal characteristics between the background and the object.
- With the proposed partial blending approach, the visibility of the obfuscated object is enhanced and the creation of implicit edges between the object and the background is minimized.
- In some cases, the color and texture information of an object is crucial for the recognition of the object. To integrate object related color features, a color is used which is based on the dominant colors of the object texture for the rendering of the scene. With the local unmasking concept, the texture features are locally blended with the obfuscated visualization of the object. The additional injection window type replicates these texture characteristics in the background.

4.7 Technical Details

In the previous sections, the algorithms of the obfuscation approach were presented, in this section, the main technical and design challenges are outlined.

4.7.1 Implementation Challenges

One feature of the implementation done for this thesis is the interactive visualization of the proposed obfuscation method by making use of the capabilities of state-of-the-art graphics hardware. We used the OpenGL 3.3 Core Profile interface for this purpose. Figure 4.17 illustrates the rendering loop and the most relevant components involved in the processing pipeline. Despite of the fact that the main part of the computations is implemented in shaders (OpenGL Shading Language), the importance sampling and the linked window handling is processed on the CPU.

As mentioned in Section 4.2.1, we use the method proposed by Ostromoukhov et al. [51] to perform importance sampling. Instead of sampling the whole importance map texture, sampling is limited to the individual areas of the objects. The limitation enables a parallel processing of the importance-map using the OpenMP framework, improving the performance of this CPU intensive task. Due to the fact that the actual positions of the sampling points are required for further processing, the transfer of the importance map to RAM, the sampling, and the final transfer of the positions back to the GPU have to be executed before any subsequent computations and constitute the main performance bottleneck of the implementation (pipeline stall). The main principle of the GPU based part of the pipeline is inspired by deferred rendering techniques, where information like normals, position etc. are stored in textures and used in the subsequent rendering passes. The rendering of the splats (see Section 4.2.3) breaks this purely GPU-oriented data-structure, due to the fact that the position of the splats are the result of the importance sampling step. To integrate additional data that is generated on the CPU side into the pipeline, Texture Buffer Objects (TBO) are used. TBOs are one-dimensional texture-arrays providing a flexible structure for transferring data to the GPU. Rendering the splats requires rendering of a dummy geometry (e.g., a cube) for each individual splat. For rendering a given number of objects with equal geometry, the concept of instancing is used. Each instance has a consistent ID throughout the pipeline, which provides a reference to the associated data in the TBOs. Hence, each instance has its own dataframe assigned in the TBO, containing the splat position, size, etc. in aligned form.

The windows that represent the background obfuscation are specified by selecting a subset of object sampling points within a rectangular window of arbitrary size. The splats for the background windows are rendered in the same way as the object splats. By applying an additional screen-space transformation to the location of the splats, the windows are positioned apart of the object and in the background. For this purpose, the window splats TBO, which encapsulates again the position, color, scale etc. of the rendered splats, includes an index that points to another TBO holding the transformation matrices for each window. A special case are splats that correspond to sticky windows (see Section 4.3), because here the idea is to define a fixed set of samples whose relative position is the same throughout the lifetime of the associated window. Therefore, the initial world position is stored for each sample on the CPU side and for consecu-

tive frames, the transformation between two frames is applied to the set, which yields the effect of a coherent motion.

At the time of writing, there was no GPU-implementation of the importance sampling method available and porting the method to the GPU was out of scope of this work. Despite this drawback, the current implementation provides an interactive visualization of the obfuscation approach averaging at 15 FPS (Intel i5 750 and NVIDIA GeForce GTX 260).

4.7.2 Offline Implementation

With respect to the big picture of the whole CAPTCHA system (see Section 3.1), we decided to implement the obfuscated rendering engine in an offline system. While WebGL integrates hardware-accelerated rendering of 3D visualizations directly in the browser environments and a real-time implementation of the pipeline in the browser seems natural, the negative aspects of such a solution is the missing native support of common browsers and the fact that WebGL is, at the time of writing, not part of the W3C standard. Besides these formal issues, the current implementation involves CPU- and GPU-intensive processing, which makes it impossible to outsource the rendering part to a heterogeneous client side due to the low performance figures on mobile devices. Furthermore, a client-side implementation of the rendering requires to transfer the scene geometry and the texture data to the client, hence, the scene complexity affects the required amount of data to transfer. Assuming that models could have high-definition textures and a high polygon count, this can exceed multiple megabytes, which is generally not compatible with the resources in a web application.

Based on these issues, the decision for an offline rendering system using videos as output is the most compatible and portable solution for the intended application. For this purpose, the videos for the challenges are created in advance, as both rendering and encoding cannot be achieved in real-time. Using the open source FFmpeg library, the obfuscated scenes are encoded with state-of-the-art video compressions standards, namely the h.264 (MP4) and VP8 (WebM) standards. At the time of writing, all common browsers support at least one of the used compression methods. In combination with the HTML5 video element, no further plugins are required to show the CAPTCHA challenges in form of a video.

CHAPTER 5

Results

This chapter covers the evaluation of the obfuscated rendering method previously presented. The first part covers a perceptual user study that we conducted to determine the limits of human object recognition abilities when using the obfuscation approach presented. In the second part we evaluate the robustness of our method against a state-of-the-art feature-based optical flow attack.

5.1 User Study

Since the intended application of the rendering engine described in Chapter 4 is its use as a fundamental part of a CAPTCHA system, the evaluation of the perceptual limits of the approach is mainly user oriented and performed in form of a user study. Automatic approaches for exploiting CAPTCHA systems are usually specialized and focused on one specific CAPTCHA method. As no automated attacks to estimate the required level of obfuscation was available at the time of creation, we decided to find the limits on a perceptual level using the aforementioned user study. In this context, the question arises, which effects of the rendering process directly affect the object recognition and how these impacts could be measured. Furthermore, we are interested in the correlation between the obfuscation effects and the degree of difficulty.

Taking into account the aspects previously mentioned, we can formulate the main motivation of the user study: Due to the nonexistence of definite limits and requirements regarding the degree of obfuscation, we are searching for the upper bound of the human ability to recognize objects. Estimation of this upper bound defines the limit before users are no longer able to reliably identify the objects.

5.1.1 Distracting Parameters

In general, an obfuscation can be achieved in a wide variety of ways due to the high number of degrees of freedoms of this problem. Our method represents a subset of the possible design space, which is characterized by a number of technical parameters. In the following user study,

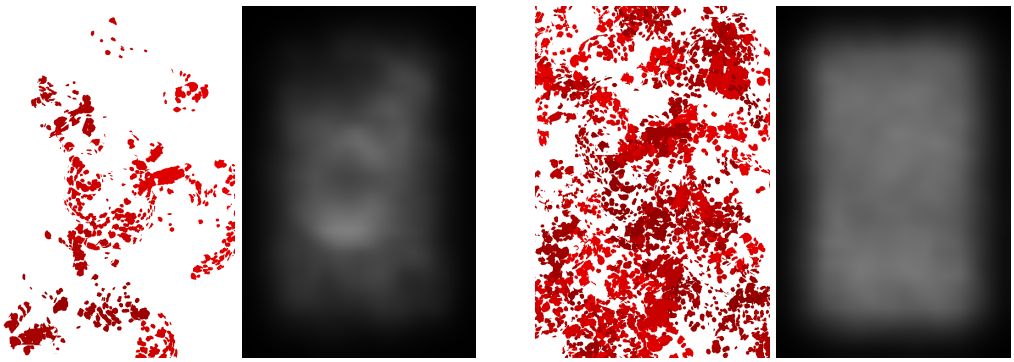
we are interested in the effect of these technical parameters on the degree of difficulty of the object recognition task. Due to the large number of technical parameters (approximately 20), a test including all of these parameters is infeasible for the following reasons: Testing all parameters simultaneously with, for example, 10 different values for each parameter would require 10^{20} test scenarios. It is obvious that these requirements are not compatible with a user-based evaluation method. When testing multiple parameters, the effect of potential *interactions* between the parameters has to be considered. Limiting the tests to *pairwise* parameter tests corresponds to a subset of all possible scenarios, where one parameter is variable and the other is fixed. It is assumed that pairwise tests enable us to isolate the main characteristics of the parameters. Despite this limitation, the number of required tests $\left(\binom{20}{2} * 10 = 1900\right)$ is still too high, hence the number of tested parameters has to be reduced dramatically. The main idea is to select *two* technical parameters that are as independent from each other as possible and show their relation to the following two perceptual-related parameters that we identified as the main perceptual parameters in our context:

- **Object Coherence**
  The idea of the object obfuscation process is to generate a sparse representation of the object that contains only the important parts of the object. A sparse representation implies that certain characteristics of the object are removed. In an obfuscated sequence, the *object coherence* describes the amount of the visible object area in relation to the area of the unobfuscated object over the whole sequence. In more detail, the largest connected area is used to estimate the object coherence. It is assumed that the ability to recognize the object is effected by the spatial characteristics of the representation. Thus, the recognition of the object becomes more difficult with a sparser object representation.

  To compute the object coherence, the obfuscated object mask M_i (see Section 4.2.3) and an index map I_i of the unobfuscated representation are required for each frame i of the sequence. Each frame is processed as follows: The aim of the first processing step is to create a connection between splats which are located close to each other. An average filter (10×10 pixels) is applied to the mask M_i resulting in a blurred image N_i; next, N_i, which has values in the interval of $[0, 1]$, is converted into a binary image using a threshold $t = 0.3$. The resulting image N_i now contains multiple regions (connected pixels with a value of 1) that correspond to connected splats. From the following segmentation and labeling of N_i, the boundary B_j of the largest connected area is determined. The boundary B_j is used to encircle an area in M_i; the splats contained in this area are selected and their total area S_i is computed. The resulting object coherence O_i of frame i is defined as the ratio $O_i = \frac{S_i}{A_i}$, where A_i denotes the area of the unobfuscated object based on the index map I_i. The object coherence O of the whole sequence is defined as the mean value of the object coherence O_i of all frames.

- **Scene Complexity**
  We introduce the term *scene complexity* to describe the distracting effect of the background in relation to the shown object. It is assumed that this distraction increases with increasing clutter in the background and therefore has a direct effect on the ability to recognize the object. To determine this effect on a perceptual level, we use the concept of

salience. For a given obfuscated video sequence, the salience for each individual frame is computed using the approach proposed by Hou et al. [31]. Additionally, we use an object index map to accumulate the salience of either the object or the background in the resulting salience map. These steps are executed on all frames of the sequence and the corresponding mean object salience S_{Obj} and mean background salience S_{Bg} are computed. The scene complexity C is defined as their ratio $C = \frac{S_{Bg}}{S_{Obj}}$. Figure 5.1 illustrates the concept of the salience ratio; with increasing clutter in the background, the salience of the background increases. In scenes with low scene complexity (Figure 5.1a), the salience is concentrated in the object area, while in scenes with a large degree of background clutter (Figure 5.1b), the salience is quite balanced between the object and the background.



(a) low scene complexity (b) high scene complexity

Figure 5.1: Comparison between different values of scene complexity using the mean salience map of a sequence (higher values correspond to higher salience).

It is assumed that the scene complexity as well as the object coherence have a direct effect on the ability to recognize objects in a scene. Therefore, two parameters have to be selected, one affecting only background clutter, whereas the other influences the object representation. The technical parameters *window count* (see Section 4.3) and *maximal splat size* (see Section 4.2.3) are suited for this purpose, as they show not only the desired characteristics but also have minimal influence on each other. From a perceptual point of view, Figure 5.2 illustrates that the window count correlates with the scene complexity, while the object coherence correlates with the splat size. Hence, these two parameters are selected as the two parameters which describe the amount of user distraction.

5.1.2 1st Experiment – Degree of Difficulty

The aim of the first experiment is to find the limits of the human ability regarding the object recognition task. As stated in the previous section, window count and splat size show the desired influence on perceptual characteristics of the obfuscation result, which leads to the natural assumption that they are main factors in describing the difficulty of the recognition task. The main idea of the first experiment is to use these two distracting parameters as input and to sample

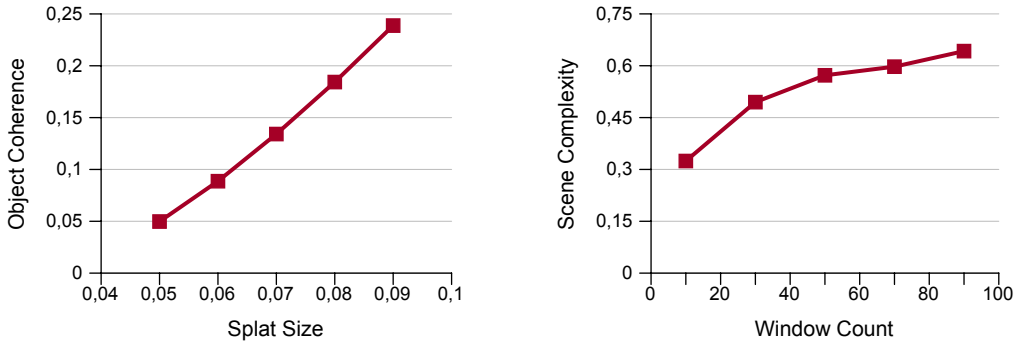


Figure 5.2: Correlation of the technical parameters and the perceptual quantities for a characteristic object.

them in a pairwise test for a limited set of parameter values. To measure the impact of a specific parameter combination on the degree of difficulty, the users' response time is used. The response time is specified as the time that a human test subject needs to perform the object recognition task. Our hypothesis that we want to validate is that the response time directly correlates with the level of difficulty indicated by these two parameters.

Test Procedure

We use conventional grid sampling to evaluate the degree of difficulty for difference values of window count and splat size. Each point of the 6×6 sampling grid represents a video containing one object, located randomly in the left or right half of the video and with the associated values of window count and splat size (see Appendix A.1 for all settings). The response time of a test subject performing the object recognition task is measured and a validation of a correct recognition is achieved by an additional textual annotation from the subject. A model repository is created that contains 36 different objects that are well known around the world. Potential effects based on the colors used are considered by the definition of 6 reference colors (red, green, blue, yellow, gray scale and inverted gray scale). Object- and color-related influences are minimized by shifting a fixed ordered list of objects through the sampling grid, meaning that after 36 participants, each model was tested on each sampling point of the grid and each occurred in every reference color six times. To avoid a gradual adaption to the parameters, the samples are traversed in random order.

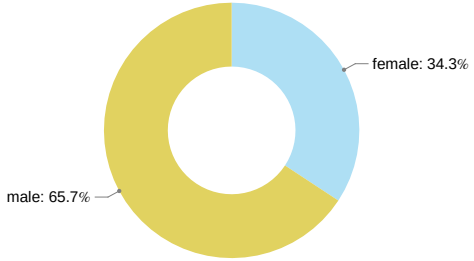
The study is implemented as an online experiment using state-of-the-art web technologies like HTML5, JavaScript, PHP and MySQL. According to the previously described concept of the experiment, all videos are prerendered and encoded with *h264* (640×480, 25 FPS, 5 sec, CRF 26 and the 'veryslow' preset) resulting in a repository of *1296* videos. With the limitation to a single codec, the inter-codec calibration with other well established codecs, like e.g. VP8 (WebM), is avoided. On the other hand, the common cross-browser compatibility problem limits the compatibility to browsers that support h264 video playback natively.

From the perspective of participants, the experiment starts by asking for demographic data (age, gender and visual impairments) and the calibration of the user position according to the foveal coverage (see Figure B.1a). Determining the resolution of the screen using JavaScript requires only the specification of the physical screen dimensions to calculate the recommended distance to the screen (for details see Section 4.1.1). In the test situation, first, the logic waits until the video is loaded to enable playback without further buffering. Afterwards, the user can start the video associated with a sample of window count and splat size by pressing a button, which initiates the timer for measuring the response time at the same time (see Figure B.1b and B.1c). In this phase the subject is asked to perform the task of recognition as fast as possible. If the user is able to recognize the object, he stops the video and the time measurement by pressing the spacebar, otherwise the system switches to the next sample after 30 seconds and the obfuscated object is marked as not recognized. In case of a recognition, the video is faded out and the user has to provide a textual description of the recognized object. After submission, a possible solution is displayed to provide feedback (see Figure B.1d).

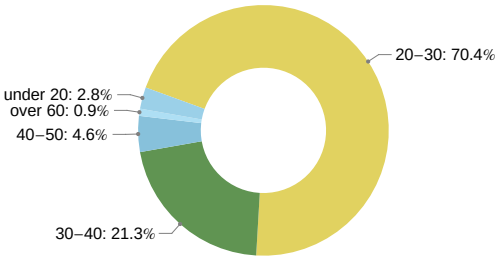
Participants have to process all 36 samples of the grid, and an additional training phase of 4 examples is integrated at the beginning of the test. This phase is equal to the test design, except that the models are not from the test repository and do not vary in any parameter dimension.

Demographic Data

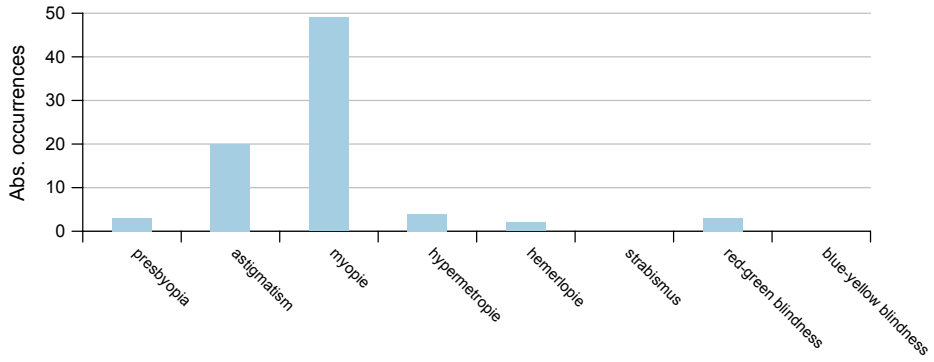
The experiment was conducted as an online experiment and participation was enabled by an open call. In addition, the crowdsourcing service CrowdFlower [10] was used to involve higher numbers of participants and to accomplish a certain degree of demographic diversity. According to the users' country of origin, the main three contributions came from Austria (59%), the United States of America (22%) and Germany (8%). Summarizing the visualization of the demographic data in Figure 5.2, the experiment was examined by *108* participants with a ratio of 65% male and 35% female users. Considering the age of the participants, the main age groups correspond to 20-30 (70%) and 30-40 (21%) years. Regarding visual impairments, 45% of the participants stated to have myopia and another 18% specified to have astigmatism. Note that a participant can have multiple visual impairments.



(a) gender distribution



(b) age groups



(c) visual impairments (60 participants out of 108 are affect; note that a participant can have multiple visual impairments)

Figure 5.2: Visualization of demographic data from the first experiment.

Degree of Difficulty

Each of the 108 participants performed all 36 samples of the grid, resulting in 3888 samples. Each of the samples was augmented with the following quadruple of information: the *sample-coordinates* in the grid, the used *model* and scene *color*, the user *response time* and the object *description* and the classification of the response as *correct* or *wrong*.

Besides responses that are classified as incorrect based on the object description, timeouts (no recognition after 30 seconds) are another implicit class of incorrect responses. Consequently, the total response error is specified by the sum of the timeouts and the number of incorrectly described samples. In Figure 5.3b, the amount of incorrect responses is shown – the distribution of the errors across the grid is depicted and normalized with respect to the maximum possible occurrences. From analyzing the trend of the total response error in the grid, an approximately monotonic behavior and the presence of interaction between the selected dimensions is observed. This implies that with increasing window count and decreasing splat size, the error increases linearly. By definition of the total response error, the error shown in Figure 5.3b is the aggregation of the errors provided by each of the color classes plotted in Figures 5.3c to 5.3h. Here, the limited set of samples per grid point (18 samples) causes a rather large variance of the distributions of errors among the samples. However, the overall similarity of the trend is observable in each color class. Table 5.1 lists the proportions of each color to the total response error, showing that the error is equally distributed among the color classes. In comparison with other color classes, the error distribution (see Figure 5.3f) of the yellow color class has a well-formed step slope along the grid diagonal. At the same time, this slope defines a border where less than the half of the samples are correct.

The analysis of the error distribution strongly points to the existence of an interaction between the selected parameters. Furthermore, the error rate can be interpreted as another indicator for the degree of difficulty at a specific point of the grid. Based on this interpretation, the difficulty increases with increasing window count and decreasing splat size.

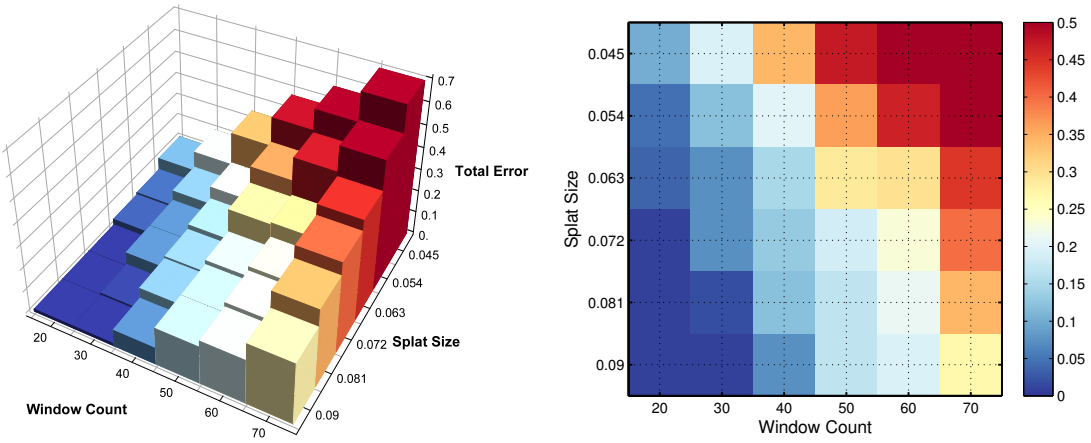
	Red	*Green*	*Blue*	*Yellow*	*Gray*	*Inv. Gray*
Error fraction [%]	15.07	17.49	16.57	17.49	16.34	17.03

Table 5.1: The error fractions of each color class on the total response error.

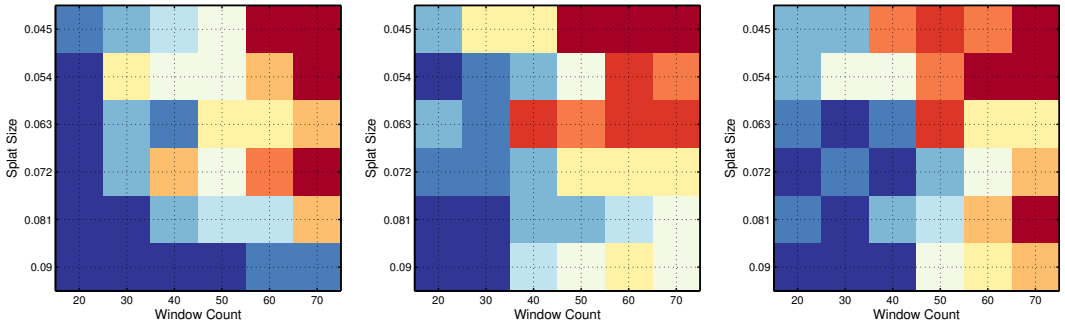
In contrast to the analysis of the error rate, an isolated observation of response timings, neglecting incorrect samples, provides a quite limited way of interpretation. The problem becomes more evident when taking a closer look at regions in the grid where the degree of difficulty is intended to be high. Here, the number of valid responses decreases to a minimum and thus makes it difficult to interpret the remaining samples in a meaningful manner. Therefore, to define an expressive measurement for the degree of difficulty, a metric has to be specified that incorporates the error rate with response timings. For each point of the sampling grid, we collected a series of response timings. This series consists of three types of responses: valid response times, timeouts, and response timings with incorrect object description. To achieve a sorting according to the response timings, the timings of responses with wrong descriptions are changed to 30 seconds and therefore are now equal with the set of timeouts. Sorting of the resulting series of responses establishes a specific structure of the sampling data. The first segment corresponds to valid responses and the following tail set contains samples which are interpreted as incorrect. Determining the 75^{th} percentile represents the upper bound response time of at least 75% of the participants, providing at the same time a valid object description. Thus, the defined metric is seen as a measurement for the degree of difficulty, represented by the resulting response time. Plotting the response timings according to the metric specifies a height field, which can be interpreted as connected surface by using linear interpolation (see Figures 5.4a and 5.4b).

Motivated by the aim to find an upper bound for the ability of human object recognition in the context of our obfuscated rendering method, the metric introduced allows identifying significant changes of the slope and, consequently, a change in the degree of difficulty. Based on the contour plot in Figure 5.4b, this change can be observed approximately along the grid diagonal, where the gap between the contour lines decreases, revealing an increasing slope. At the same time, the increasing slope specifies a well-defined limit where the response time increases dramatically to 30 seconds, outlining those areas in the grid that are unsolvable for 75% of the users. Analyzing the 6 scene colors (see Figures 5.4c to 5.4h), considering the limited sample size, shows a similar distribution with no significant variations. Based on these observations, we selected a threshold (outlined in Figure 5.5) such that the recognition performance is stable while the degree of difficulty is still acceptable.

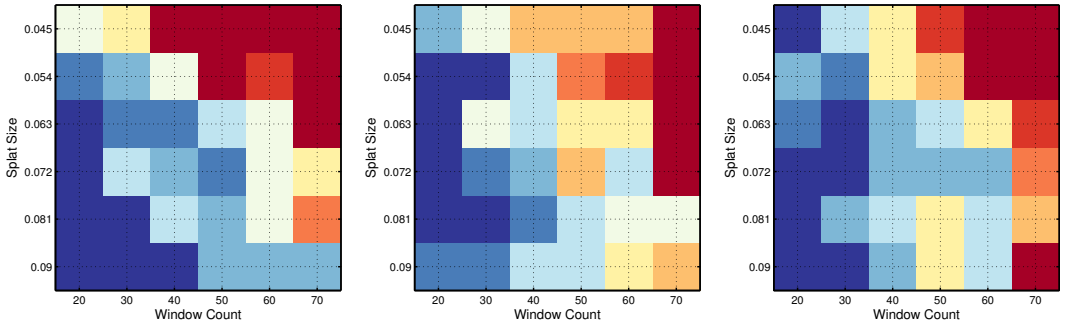
With this first experiment, an approach for determining and evaluating the degree of difficulty was proposed, highlighting the existence and effect of interaction between the tested parameters. Based on the results, a threshold (highlighted in Figure 5.5) was determined that corresponds to the limit of the human ability of object recognition using our obfuscation method.



(a) (b)

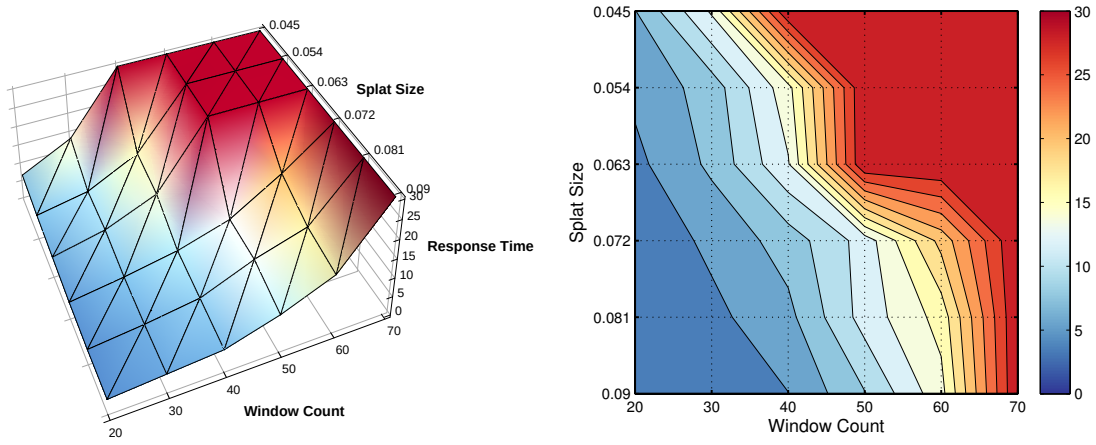


(c) red (d) green (e) blue

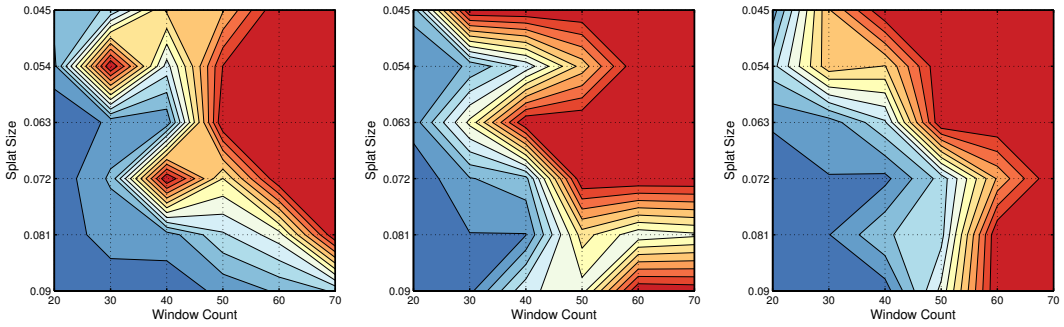


(f) yellow (g) gray (h) inv. gray

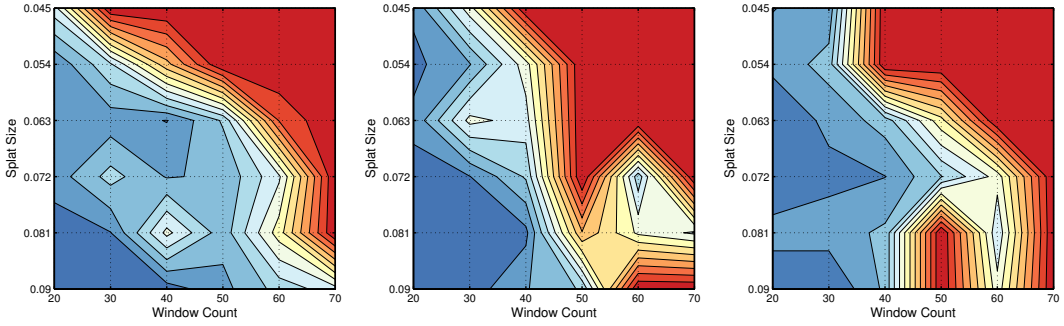
Figure 5.3: Normalized error occurrences across the sampled grid – (a-b) the total response error, normalized to all 108 samples; (c-h) errors of each color class, normalized to the max. possible occurrence of 18 samples;



(a) (b)



(c) red (d) green (e) blue



(f) yellow (g) gray (h) inv. gray

Figure 5.4: Visualization of the specified metric based on the 75^{th} percentile of the responses – (a-b) overall distribution and (c-h) each color class independently.

5.1.3 2nd Experiment – Usability

In the first experiment, the degree of difficulty was determined by introducing a metric that involves all available types of responses. As outlined in Figure 5.5, the derived threshold (see Section 5.1.2 for details) corresponds to a limited area in the grid, consisting of a subset of grid points. We interpret this threshold as the limit of the human ability of object recognition using our obfuscation method. From the specifications made in the first experiment follows that sample points on the threshold area equal in difficulty and model a specific configuration regarding the window count and splat size. Based on these findings, the aim of the second experiment is to test and compare those grid points to each other to find the most usable and pleasant configuration for the user.

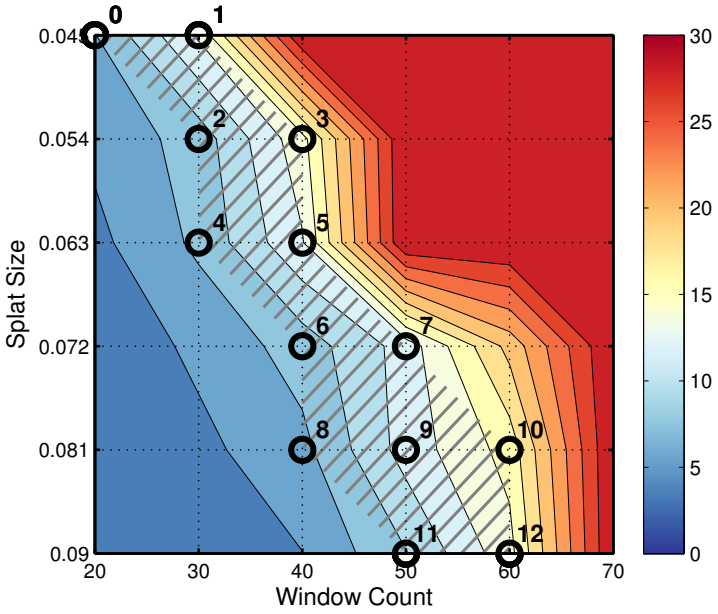


Figure 5.5: Selected points and the resulting threshold area.

Pilot Study Experience

Before testing the complete set of points in the threshold area, we decided to perform a pilot study to test if the following methodology acts as intended. For this purpose, a reduced subset of 7 sample points from the test area (shown in Figure 5.6a) is used for the comparison. Each point of the test set is directly compared with all other points of the set, except comparisons with the point itself and points that are direct neighbors in the grid, resulting in 13 pairwise comparisons.

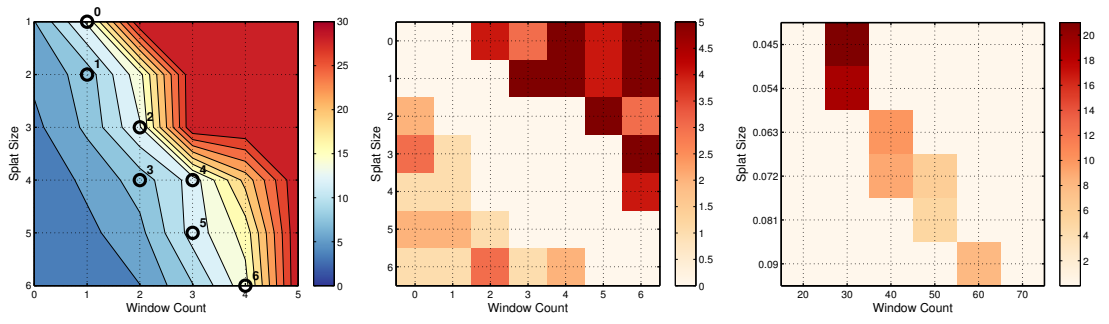
Analogous to the procedure of the first experiment, the study is performed online using the same video settings as stated in Section 5.1.2. In contrast to the first experiment, the main idea of this test is to model the comparison of grid points by presenting the user two videos side by side. The videos show the same object in the same scene color with different configurations that

correspond to different points on the grid. The user can directly compare the configurations and has to select the subjectively *more pleasant* one by clicking on the video. Participants have to perform all comparisons twice, resulting in 26 responses per user. Equal to the first experiment, models and colors are iterated through the sampling grid such that each user sees 26 models at different positions in the grid. To avoid patterns in the sequence, the comparisons are arranged randomly as well as the video arrangement within a comparison.

The concept of pairwise comparisons of t objects is modeled by using a $t \times t$ adjacency matrix A, where the test objects are the set of sample points describing different parameter configurations (illustrated in Figure 5.6b). Elements of the matrix hold the voting counter A_{ij}, which is interpreted in the way that configuration i is preferred A_{ij} times to configuration j [57]. To estimate a ranking of the tested sample points, the total number of votes a_i for each configuration i is calculated by:

$$a_i = \sum_{j=1}^{t} A_{i,j}, i \neq j \tag{5.1}$$

Figure 5.6c illustrates the ranking of the 156 collected responses, provided by 6 participants in the sampling grid. The plot reveals an obvious preference to a low window count and splat size. Despite the clear results of the ranking, the direct user feedback was more important to check if the procedure evaluates the intended aspects. The conclusion of the feedback is that comparing the scenes simultaneously leads to a negligence of the object recognition and shifts the focus to a purely aesthetic comparison of the scene as a whole. From the perceptual point of view, by definition of the perceptual parameters (scene complexity and object coherence), the used experiment design focuses on the scene complexity and ignores the effects caused by the object coherence. Finally, the results and user feedback of this study lead us to the conclusion that the object recognition task has always to be integrated into the testing procedure even if the aim of the study are aesthetic and usability aspects.



(a) (b) (c)

Figure 5.6: Pilot experiment – (a) the set of used sampling points, (b) the adjacency matrix visualizing the votes from pairwise comparisons and (c) ranking of the test samples based on the total number of votes in the sampling grid.

Test Procedure

In the following experiment, the user acceptance is tested across the threshold area defined by 13 grid-points (see illustration in Figure 5.5), using pairwise comparisons between the individual points. Similar to the pilot experiment, the main idea of the experiment is that two configurations are presented and the user selects respectively votes the configuration which is more pleasant.

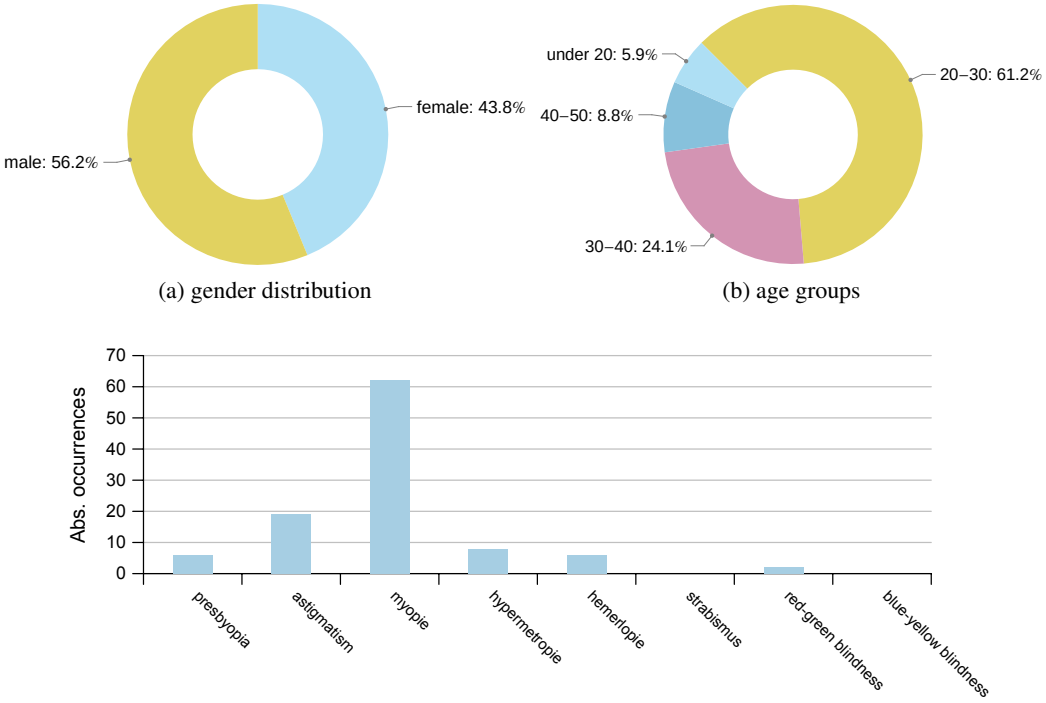
Based on the outcomes of the pilot study, the design of the experiment was changed such that the intended task, the object recognition, is integrated in the comparison process. Inspired by the procedure of the first experiment, the object recognition is integrated by using two videos containing different objects with the same scene color for the specification of one comparison. To validate the user response, the objects have to be textually annotated by the user. In detail, the videos are shown independently of each other, hence, the user performs two object recognition tasks and selects afterwards the more pleasant one (forced-choice principle). Reusing the model repository (36 models) of the first experiment allows sampling 18 comparisons per participant. In addition, another 2 comparisons are included at the beginning of the experiment as a training phase, extending the model repository to 40 models.

Each possible combination of configurations results in $\binom{13}{2} = 78$ comparisons. Due to the high number of comparisons, the concept of sampling allocation is changed to a progressive approach, using an adjacency matrix as representation for the sampling. The samples are randomly assigned to participants such that the sampling rate over all samples is equal (samples are only used once per user). Two additional statistics (histograms) are linked with each entry of the adjacency matrix, balancing the occurrence of models and colors. Compared to the first experiment, each user response is stored, except the samples from the training phase.

This experiment is also implemented in form of an online experiment using the same technologies and video settings as stated in Section 5.1.2. The required set of videos is prerendered using all possible variations regarding models and colors (3120 videos). From the perspective of the participants, the experiment starts by inquiring demographic data (age, gender and visual impairments) and the calibration of the user position according to the foveal coverage (see Figure B.2a). In contrast to the pilot study, the user has to perform the recognition task for each configuration independently (see Figures B.2b and B.2c), followed by the pairwise comparison based on the unobfuscated visualizations (see Figure B.2d). Using the unobfuscated representations provides a feedback about the solution and simultaneously avoids that the comparison is focused on the overall scene complexity.

Demographic Data

Like in the first experiment, the crowdsourcing service CrowdFlower [10] was used to enable a high number of participants. According to the users' country of origin, the main three contributions came from the United States of America (60%), Austria (20%) and Germany (9%). As illustrated in Figure 5.7, the experiment was examined by *170* participants with a ratio of 56% male and 43% female users. Considering the age of the participants, the main age groups correspond to 20-30 (61%) and 30-40 (24%) years. Similar to the first experiment, the main visual impairments are myopia with 57% and astigmatism with 18%.



(a) gender distribution

(b) age groups

(c) visual impairments (74 participants are affect; note that one participant can have multiple visual impairments)

Figure 5.7: Visualization of demographic data from the second experiment.

Ranking of Configurations

The 170 participants of the experiment performed 1653 valid comparisons (2540 total), which corresponds to an average of 22 samples per comparison. A sample is classified as valid if both objects are recognized correctly. According to the specification of the experiment, the result of each comparison is a vote for a specific configuration. The results of the voting are modeled using a 13×13 adjacency matrix A (shown in Figure 5.8), where elements A_{ij} of the matrix refer to votes for configuration i over j.

Based on the decisions made in the pairwise comparisons between the individual configurations, the Bradley-Terry-Luce (BTL) model is used to determine a ranking of the preferred configurations [11, 53]. The BTL model can be formulated in terms of generalized linear models using logit as the link function:

$$logit[pr(i \text{ preferred to } j] = \theta_i - \theta_j \tag{5.2}$$

where θ_i can be estimated by maximum likelihood. As detailed background information about the BTL model is beyond the scope of this section; the reader is referred to the paper of Pietsch [53]. A crucial advantage of the BTL model is that it is able to handle incomplete data sets, thus it is not necessary to have an equal number of samples at each point of the sampled matrix.

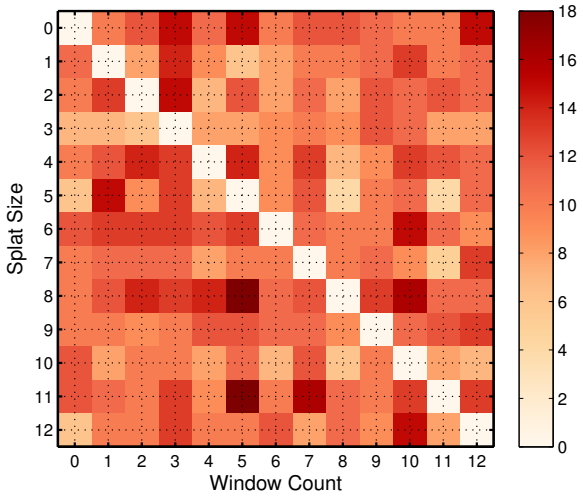


Figure 5.8: Adjacency matrix visualizing the votes for all possible combinations.

The collected data is fitted into a special matrix design where the columns denote the number of comparisons and the rows to the response of one participant. Entries of the matrix can have three values, 1 is interpreted as object A is preferred over object B, where the value 2 has the inverse meaning. If no data is available for one comparison, it is marked with the value NA. From these specifications and the number of participants, the final matrix has a dimension of 170×78.

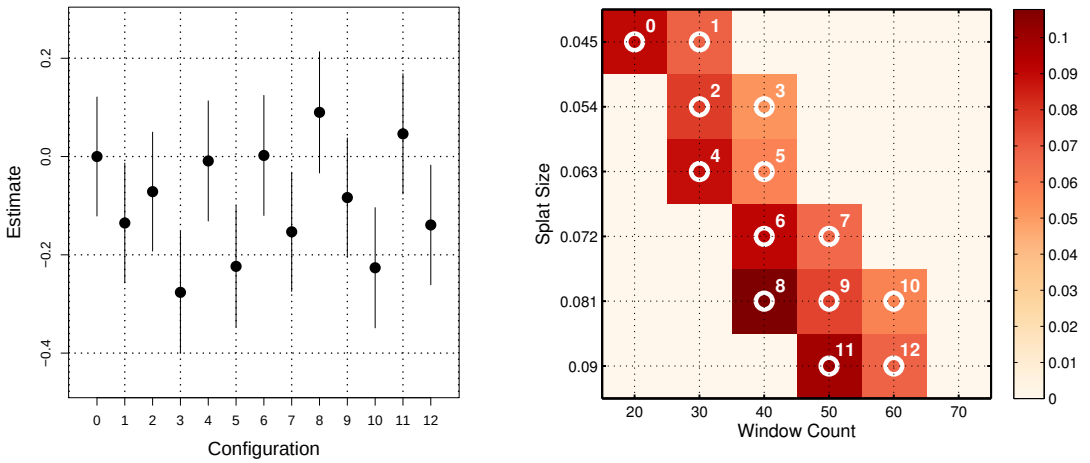
Using this matrix in combination with the BTL model results in the estimated worth parameters listed in table 5.2 (AIC 885.33). Since the estimates correspond to the ranking of the individual configurations, the high significance of configuration *11* and *8* is interpreted as the preferred configuration within the threshold area.

0	**1**	**2**	**3**	**4**	**5**	6
0,0901	0,0687	0,0781	0,0518	0,0885	0,0576	0,0905
7	***8***	**9**	**10**	***11***	**12**	
0,0663	**0,1078**	0,0762	0,0573	**0,0988**	0,0682	

Table 5.2: Estimated worth of the BTL approach using a generalized linear model; highly significant configurations are highlighted.

Furthermore, the significance of the ranking is evaluated using a *likelihood ratio test*. The idea behind this test is to determine how likely it is that the same ranking is achieved under the assumption that the used statistical model is based on a random generator. To be more specific, the distance between two statistical models, the random and our estimated one, is compared. For this, the maximum likelihood (L_1 and L_2) is determined for both models as described by Pietsch [53], and the ratio $\lambda = L_0/L_1$ is calculated, which is between 0 and 1. The ranking is less likely to be reproducible by a random generator with decreasing λ. From the ratio λ the χ^2 is determined by $\chi^2 = -2ln(\lambda)$, which increases with decreasing λ. Using the Chi Square

distribution, the assumption is rejected if a certain confidence level is achieved. In our case, the resulting $\chi^2 = 41.19$ corresponds to a p-value of 0.99 with 13 degrees of freedom. Hence, the assumption that the ranking could be the result of a random model is rejected.



(a) (b)

Figure 5.9: (a) visualization of the estimated worths including std. errors and (b) resulting ranking colorcoded in the original grid.

When comparing the ranking with the results (compare with Figure 5.6c) of the pilot study, the monotonic behavior of the preferences changed substantially to a more balanced distribution. Through integration of the recognition task, the ranking no longer depends on one of the tested parameters. When the ranking of the tested points is compared to the degree of difficulty (see Figure 5.9a and 5.5), a slight correlation can be observed.

Ranking of Objects

Based on the gathered data of the pairwise comparisons, a second ranking is performed to identify possible preferences between the used set of objects. In this context, user votes of the second experiment are interpreted as a preference in the two presented objects. Consequently, the used pairwise votes and model repository, containing 40 objects, is represented by a 40×40 adjacency matrix modeling all 780 possible combinations.

When studying the ranking between the objects, it has to be considered that by construction of the second experiment, an object recognition task is linked with the comparison. Moreover, it can be assumed that the difficulty of the recognition task influences the subjects' voting behavior. This assumption is supported by the fact that 83% of the participants voted for the correctly annotated model if the other object was not identified correctly. We define the preference of object A over B if either object A was selected and both objects were recognized, or A was recognized and B not.

Equivalent to the evaluation of the threshold area, a ranking of the preferred objects is determined using the BTL model. The result of this ranking is depicted with Figure 5.10. Supported by the fact that the used obfuscation method reveals mainly low-frequency features, the plot shows that objects with quite simple geometry containing sharp edges are preferred by the participants. However, objects like the snail do not contain enough relevant low-level features and are therefore less preferred. Analogous to the previous ranking of configurations, the ranking is tested using log likelihood ratio test, resulting in a Chi Square value of $\chi^2 = 359.40$, which corresponds to a p-value of 1.00 with 40 degrees of freedom. Therefore, the reported ranking and the associated model is highly significant.

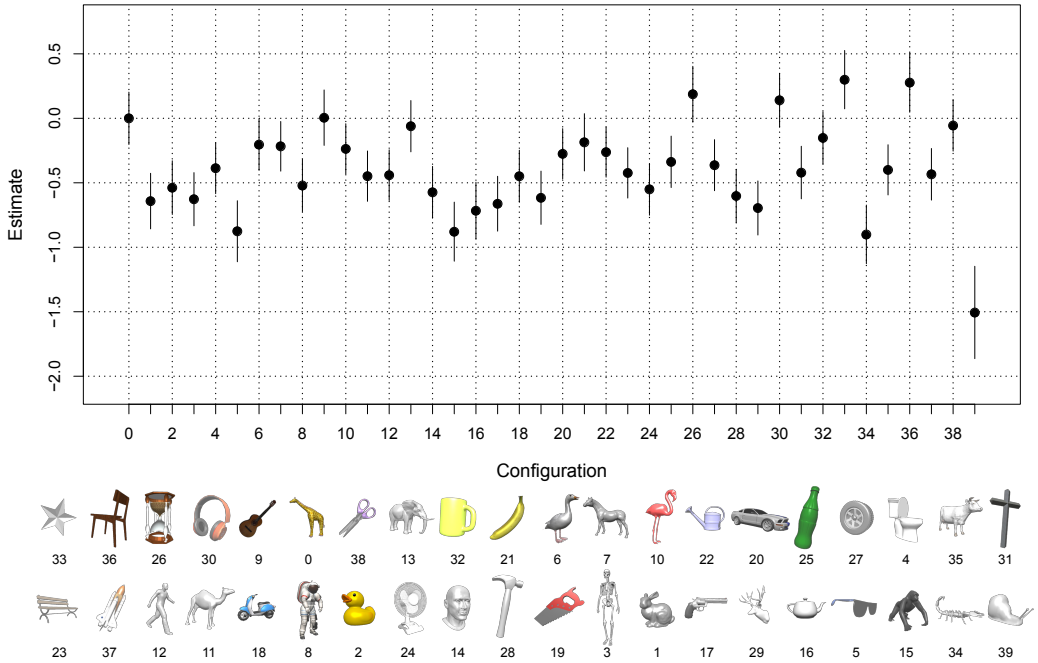


Figure 5.10: Visualization of the model ranking based on the estimated worth parameters. The models are sorted in descending order according to their ranking (from left to right).

5.2 SIFTflow

In contrast to the user-centered evaluation of the previous Section 5.1, this part aims at testing the obfuscation approach from a computational perspective. At the same time, this evaluation represents a possible attack against the proposed method. Due to the focus on animated sequences, we are interested in information which is generated between frames, especially in motion-related features. Liu et al. [43] proposed with SIFTflow a method that is based on the optical flow of image pairs. It uses the well-known SIFT features [45] for resolving feature correspondences between frames instead of the pixel-based matching as used in traditional optical flow. A 128-dimensional SIFT descriptor is determined for each pixel in the frame resulting in SIFT images. Matching is done by using a coarse-to-fine principle, where at the top-level, a coarse matching between SIFT descriptors is estimated. The coarse-to-fine hierarchy is processed by using the information of the coarse matching to successively derive a finer and more detailed matching between the SIFT images. From the correspondence of SIFT features, a 2D flow-vector can be determined representing the motion of a specific feature from one frame to another. In our context, this motion information could be utilized to identify the position of the object and its motion characteristics (e.g. direction of rotation).

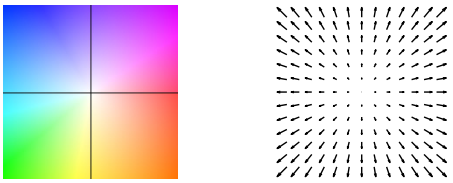
5.2.1 Test Procedure

The underlying idea of this test is to compute the flow vectors for a given obfuscated sequence and check the resulting flow maps against a ground-truth. The ground truth is generated by evaluating the flow in the unobfuscated version. The following three different degrees of obfuscation are tested in the test scenario:

1. object obfuscation without any kind of background processing
2. object and background obfuscation without the usage of experimental features (see Section 4.5)
3. full set of features including experimental options

Test sequences consists of 125 frames and are generated using the resulting preferred configuration of the user study (configuration 8 – window count: 40, splat-size: 0.081 – see Section 5.1.3) and a model repository of 6 representative models derived from the object ranking (see Section 5.1.3), randomly paired to scenes.

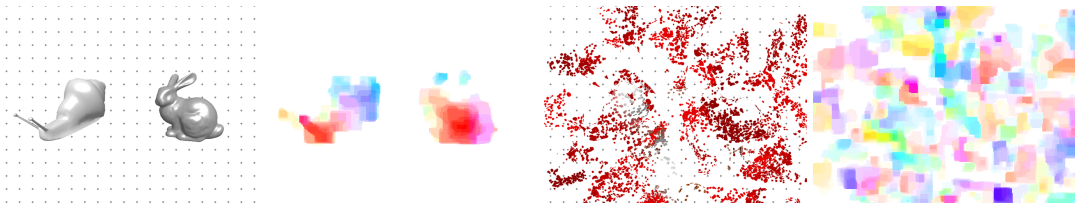
Flowmaps are computed between successive frames using the SIFTflow approach [43] and compared with the ground truth data provided by the unobfuscated sequence. Figure 5.12 delineates examples of the obtained flowmaps from the obfuscated and unobfuscated scene using the color encoding illustrated in Figure 5.11. The similarity between flowmaps is specified by the element-wise comparison of the flow vectors based on differences in angle and magnitude, resulting in a map of difference vectors. For evaluation, multiple degrees of allowed deviations (0%, 10% and 20%) were defined; in detail, the allowed degree of difference in the vector magnitude, to get classified as sufficient similar, is normalized to the minimal and maximal deviations of the whole difference-vector map. In contrast, an allowed angular difference of 20%



(a) (b)

Figure 5.11: Flow visualization – (a) color encoding of corresponding flowvectors (b). (images from [43])

corresponds to a maximal variation in orientation to the reference flow of $\pm 36°$. At this point, it has to be stated that SIFTflow shows an indeterministic behavior for blank white backgrounds. Therefore, a regular grid of dots is integrated in all frames of the test samples and the ground truth data.



(a) (b)

Figure 5.12: Example for flow maps of (a) an unobfuscated scene and (b) an obfuscated scene, along with the respective ground truth.

5.2.2 Results

The procedure previously described implies that processing the sequences produces a set of flow maps, where each vector of the map is consequently classified as correct or incorrect flow depending on the degree of difference allowed. To further describe and compare the results of the classification, the metric used corresponds to the ratio of the sufficiently similar flow vectors to the whole scene – the *scene flow*. In addition, the *object flow* is specified as the correct flow within the object boundaries. Both measures are calculated by the average over all objects. Regions where no flow is detected in the test sequence and the ground truth are explicitly considered as correct flow.

Figure 5.13 describes the resulting rates of matching flow when processing sequences containing only the obfuscated objects. The sequences are evaluated for all defined degrees of deviations (0%, 10% and 20%) as a function of the frames. As the mean values in Table 5.3 reveal, the overall detected scene flow matches to approximately 92% with the ground truth data and is nearly independent from the rate of deviation. The high agreement with the ground truth is reasoned due to the unbalanced ratio between the areas of the objects and the background.

	Obj. Flow (Scene Flow) [%]		
Configuration	*20%*	*10%*	*0%*
Obj. obfuscation	39.9(93.6)	22.4(91.7)	9.8(89.9)
Std. obfuscation	34.9(31.9)	18.7(30.0)	5.3(28.4)
Std. obfuscation & exp. features	37.7(34.2)	20.5(32.2)	6.3(30.5)

Table 5.3: Mean results of object and scene flow for all test-configurations and for all three degrees of deviation.

Taking the object flow into account highlights that up to 40% of the object motion could be detected. Contrary to the scene flow, the object flow has an approximately linear characteristic depending on the matching accuracy.

When introducing background-processing, the mean scene flow decreases from approximately 92% to 30% and therefore shows a significant impact on the scene flow (see Figure 5.14). The concept of copying parts of the object into the background, which involves the object motion (see sticky windows in Section 4.3), functions as attractor for SIFTflow. Additional fluctuations of the scene flow are the consequence of the dynamic spatio and temporal definition of the background (see Section 4.3 for the definition of the lifetime parameter). Since windows in the background are allowed to overlay untextured object-areas, the mean object flow slightly decreases compared to the first configuration (see Table 5.3). Despite the introduction and effects of the dynamic background, the ratios between the degrees of differences remain approximately the same.

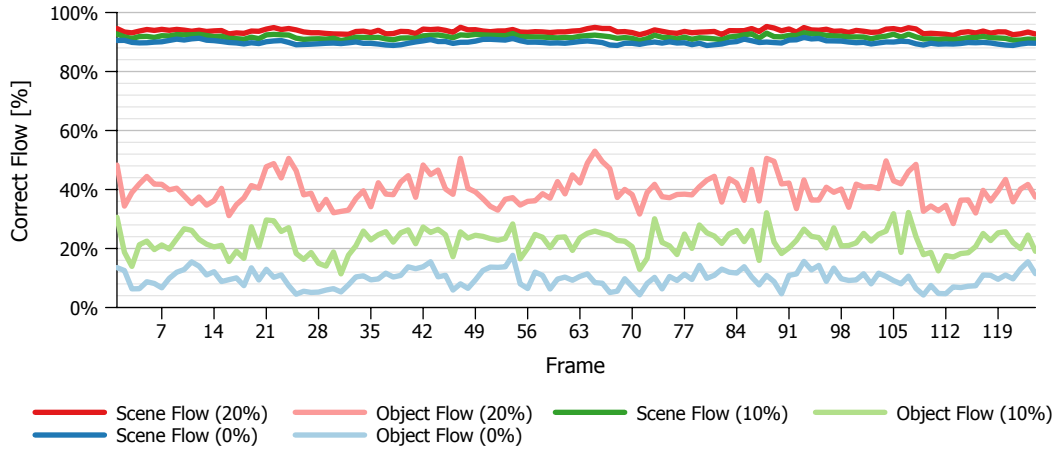


Figure 5.13: Test result using object obfuscation without background processing.

The previous configuration is the most conservative variant of the background generation. Using the partial blending and local unmasking features produces, at first view, quite similar results as without these features (see Figure 5.15). The change becomes more evident when taking a closer look at the object flow. Compared to Figure 5.14, the main characteristic of

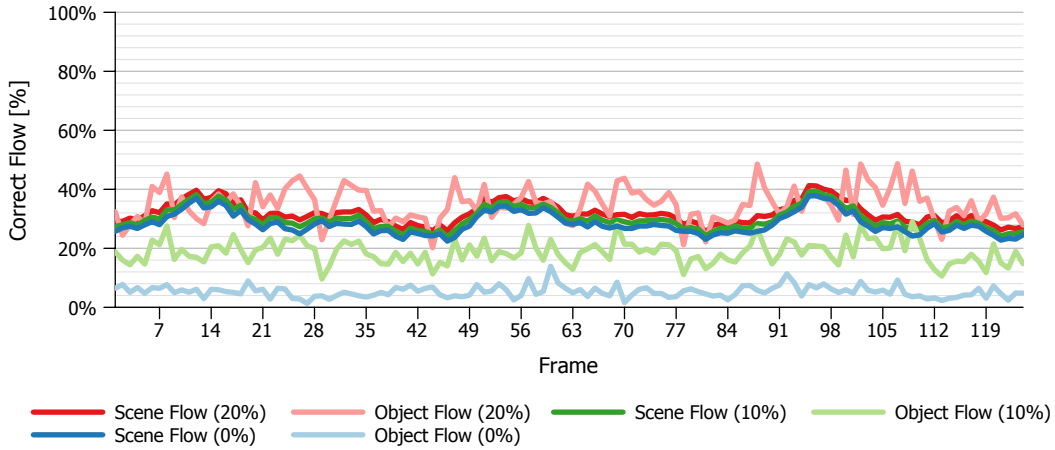


Figure 5.14: Test result using object obfuscation and background processing, excluding experimental features.

the object flow changed, containing less outliers and showing a high-frequency behavior. By construction of the local unmasking feature, additional splats are generated and dynamically increased to reveal more of the surface information but in a sparse manner. The smoother trend over all frames indicates that the continuous motion of the unmasked area on the object surface favors the tracking of the underlying surface features by SIFTflow.

The presented experiments showed that background processing reduces the correctly detected scene flow to one third, so background processing is an effective attractor for motion detection algorithms.

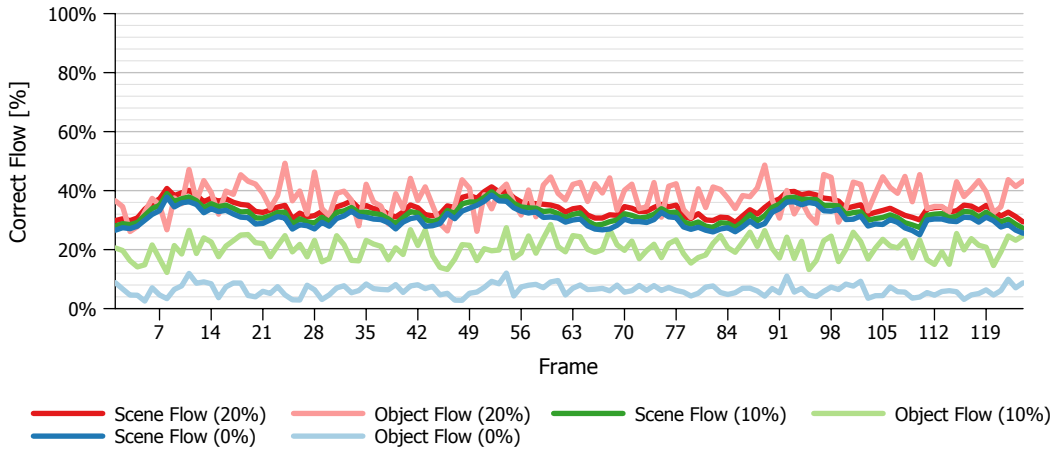


Figure 5.15: Test result using object obfuscation and background processing, including experimental features.

CHAPTER 6

Conclusion and Future Work

In this work, we presented an algorithm for the generation of an animated and obfuscated visualization of 3D scenes containing static 3D models. Inspired by the general-purpose idea behind the reCAPTCHA approach and the problem of barely annotated 3D model repositories, we integrated the human computation task of annotation into the context of a CAPTCHA challenge.

Taking a closer look at the main problem revealed that an obfuscation can be achieved in a variety of ways. Due to the fact that the 3D databases contain primarily static models, we use continuous object transformations to create an animated scene. This animation enables the user to perceive the salient features relevant for the recognition of the object that could be distributed across the whole object geometry. However, new problems regarding spatial and temporal aspects in the obfuscation process arise when using animated sequences; issues that are directly addressed by our method. With the introduction of new components for the generation of background clutter, the object is camouflaged by a background with similar temporal and spatial coherence characteristics. The main idea of the obfuscation method is to reduce the representation of the objects to their low-frequency geometry features. To enable a more detailed and specific annotation of the models, additional surface features are revealed using the novel local unmasking approach. Evaluating the method with SIFTflow, a state of the art optical flow algorithm, has shown that the proposed background processing acts as attractor for motion features and that a background obfuscation is required, otherwise main parts of the object motion are trackable.

The second focus of this work was to determine the performance and limitation of human object recognition capabilities in our context. By selecting two technical parameters as representatives for abstracted perceptually related dimensions, the performance was evaluated first in an online user study by using a grid sampling method. The results point to an interaction between the selected dimensions and that these parameters affect the object recognition ability. The introduction of a new metric enabled us to interpret the gathered results in terms of degrees of difficulty. In addition, a well-defined threshold area was determined, corresponding to the limits of human object recognition in this context. Due to the fact that the required degree of obfuscation can not be determined, this perceptual limit defines the upper bound for the degree

of obfuscation. In a subsequent experiment, the determined threshold area was tested for its user acceptance. We presented a test procedure to create a ranking between the individual configurations by performing pairwise comparisons, while focusing on the integration of the intended recognition task at the same time. As part of the second experiment, the collected data was used to derive a ranking of preferred objects. The resulting ranking has shown that objects with quite simple geometry containing hard and dominant edges are preferred by users.

Besides these technical facts, from the direct user feedback, we observed a competitive component in the course of the first experiment. Users tried to compare their individual recognition performance with each other (e.g. number of correct samples and recognition timings). Hence, the used experiment methodology has the potential to be integrated into a gaming scenario.

Future Work

In this work, we presented our obfuscated rendering method as part of a CAPTCHA system. The algorithm requires two models as input and generates an obfuscated representation of the scene in form of a video. To complete the big picture of the CAPTCHA framework, further investigations has to be done on how to handle the annotation process. An important part of the annotation process is the verification of the user input, which involves quite new problems when comparing the annotations on a semantic level.

With respect to a practical use of the rendering algorithm, further improvements regarding the reduction of the overall noise characteristics of the obfuscation method have to be implemented to enhance the acceptance by the users. The experimental features (color and texture of the object) raised additional open issues which have to be studied with respect of their security and perceptual aspects. Due to these unresolved issues, a specialized user study has to be created to understand their effects. Another problem to solve is that the approach shows a strong object dependency, therefore not all objects have the same degree of difficulty. A potential improvement of the method could be the change of the importance sampling to a preprocessing step, where the object is completely sampled and the resulting set is filtered and optimized for the rendering. The sampling optimization might consider relevant features of the surface texture to enable a targeted unmasking.

When evaluating the security of our method, we noticed that there is a lack of a standardized testing method and tools, making it impossible to estimate the realistic degree of security compared to other methods. However, in this context it can be stated that, based on the fundamental principle of this work, a successful automated recognition of the obfuscated objects implies at the same time an advance in recognition technologies.

APPENDIX A

User Study – Test Parameters

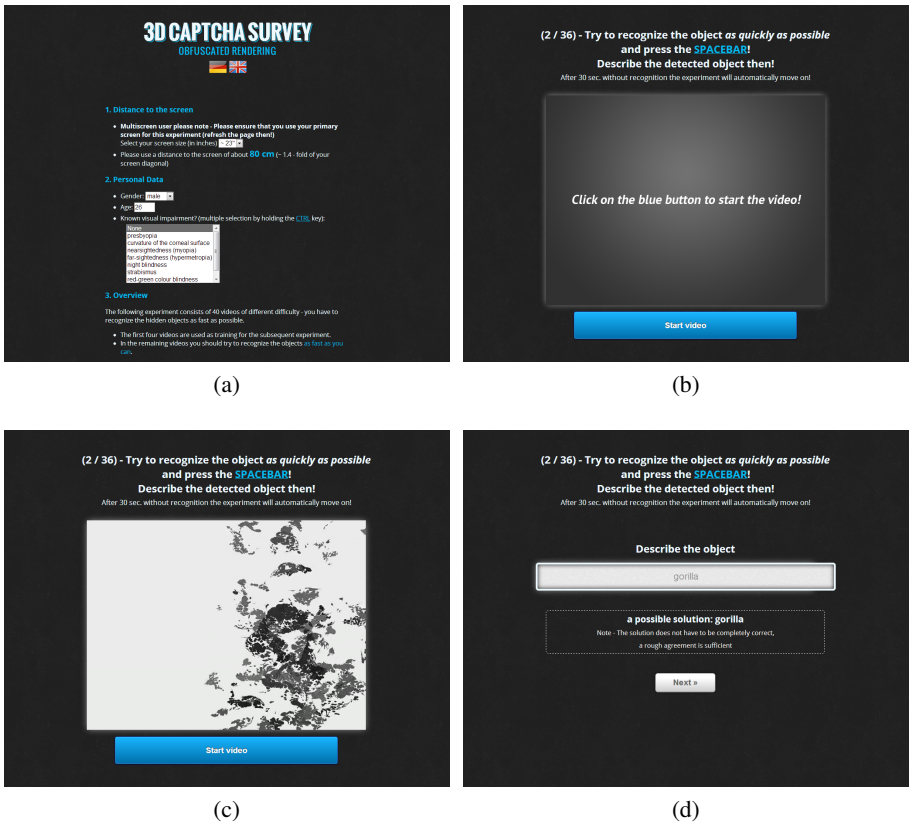
Obfuscation	
window splat count	[20 140]
window lifetime	[0.25 0.8]
window size	[0.35 0.65]
window color hue jitter	[0.0 0.0]
window color value jitter	[0.0 0.20]
window color saturation jitter	[0.0 0.0]
window use random splat selection	true
use additional background noise	false
window parameter jitter interval	[0.1 0.2]
use object unmasking	false
object alpha	[1.0 1.0]
object parameter jitter interval	[0.5 1.5]
object fast parameter jitter interval	[0.08 0.12]
light position	[0.0 -25.0 30.0]
Object Properties	
self rotation speed	216.0
self rotation axis	[0.0 0.3 -0.025]
splat dense factor	450

Camera	
foveal overview	0.225
eye position	[0.0 0.0 -1.4]
yaw	0
pitch	-0.1
fov	80
Video	
duration	5
width	640
height	480
fps	25
outputcontainer	mp4
h264 rc	crf
h264 crf	26
h264 preset	veryslow

Table A.1: Test parameters of the user study.

APPENDIX B

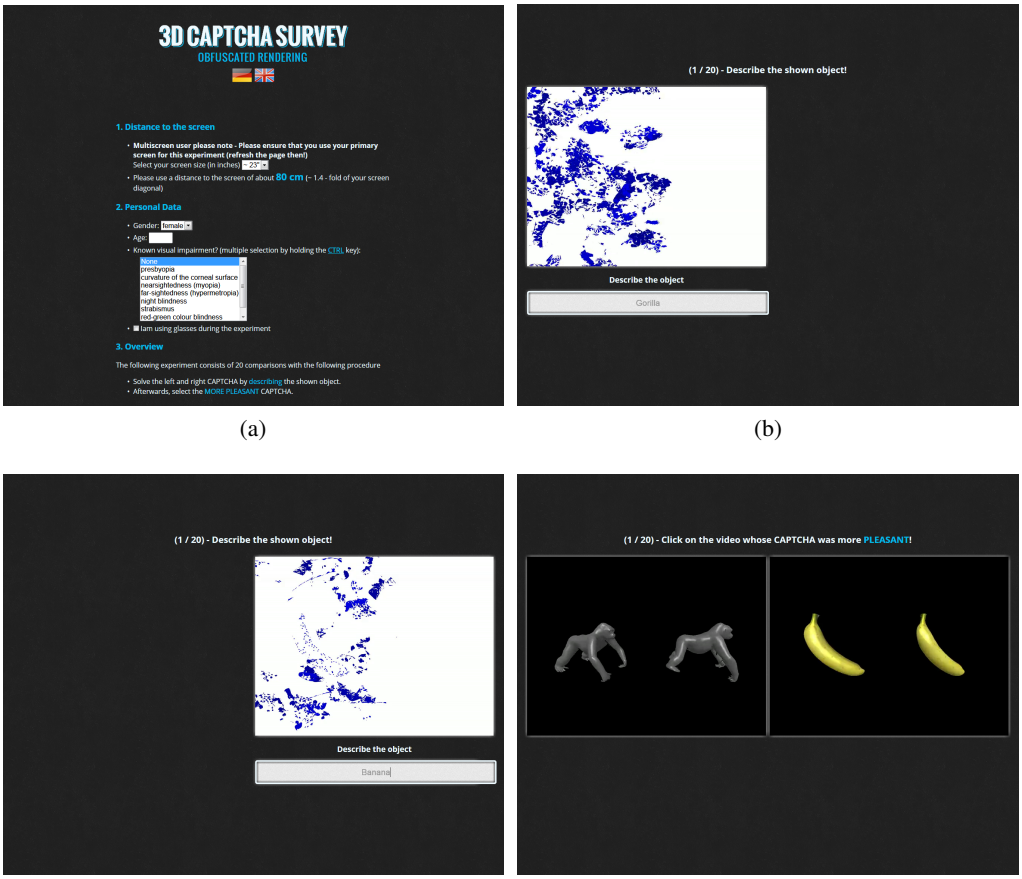
User Study – User Interface



(a) (b)

(c) (d)

Figure B.1: User interface of the first experiment – (a) introduction and explanation, (b) loaded sample, (c) test sample and (d) response validation using object annotation.



(a) (b)

(c) (d)

Figure B.2: User interface of the second experiment – (a) introduction and explanation, (b-c) independent recognition tasks and (d) preference selection using the unobfuscated representation of the scenes.

Bibliography

[1] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the 2004 Conference on Human Factors in Computing Systems - CHI '04*, pages 319–326, New York, New York, USA, 2004. ACM Press.

[2] Luis Von Ahn, Manuel Blum, N Hopper, and John Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques*, pages 294–311, 2003.

[3] Luis Von Ahn, Manuel Blum, and John Langford. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60, February 2004.

[4] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science (New York, N.Y.)*, 321(5895):1465–1468, September 2008.

[5] Vicki Bruce, Mark A. Georgeson, and Patrick R. Green. *Visual Perception: Physiology, Psychology and Ecology*. Psychology Press, 2003. ISBN 1841692387.

[6] Elie Bursztein, Steven Bethard, Celine Fabry, John C. Mitchell, and Dan Jurafsky. How Good Are Humans at Solving CAPTCHAs? A Large Scale Evaluation. In *2010 IEEE Symposium on Security and Privacy*, pages 399–413. IEEE, 2010.

[7] Elie Bursztein, Matthieu Martin, and John Mitchell. Text-based CAPTCHA strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security - CCS '11*, volume 2011, page 125, New York, New York, USA, 2011. ACM Press.

[8] Yang-Wai Chow and Willy Susilo. AniCAP: an animated 3d CAPTCHA scheme based on motion parallax. In *Proceedings of the 10th International Conference on Cryptology and Network Security*, pages 255–271, 2011.

[9] Dennis Coon and John O. Mitterer. *Introduction to Psychology: Gateways to Mind and Behavior*, volume 20. Cengage Learning, 2008. ISBN 0495599115.

[10] Crowdflower Inc. Crowdsourcing, Labor On Demand, 2012. URL `http://www.crowdflower.com/`.

[11] Jan de Leeuw. Journal of Statistical Software. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):128–129, July 2009.

[12] Agamemnon Despopoulos and Stefan Silbernagl. *Taschenatlas der Physiologie*. Thieme, Stuttgart, 2003. ISBN 3135677060.

[13] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-mauroux. Mechanical Cheat : Spamming Schemes and Adversarial Techniques on Crowdsourcing Platforms. In *CrowdSearch 2012 Workshop at WWW 2012*, pages 26–30, 2012.

[14] Manuel Egele, Leyla Bilge, Engin Kirda, and Christopher Kruegel. CAPTCHA smuggling. In *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*, page 1865, New York, New York, USA, 2010. ACM Press.

[15] Jeremy Elson, JR Douceur, and Jon Howell. Asirra. In *Proceedings of the 14th ACM conference on Computer and communications security - CCS '07*, pages 366–374, New York, New York, USA, 2007. ACM Press.

[16] Jeremy Elson, JR Douceur, and Jon Howell. ASIRRA - Microsoft Research, 2012. URL `http://research.microsoft.com/en-us/um/redmond/projects/asirra/`.

[17] Wolfgang Engel. *GPU Pro2: Advanced Rendering Techniques*. CRC Press, 2011. ISBN 1568817185.

[18] E. Estelles-Arolas and F. Gonzalez-Ladron-de Guevara. Towards an integrated crowdsourcing definition. *Journal of Information Science*, 38(2):189–200, March 2012.

[19] Florian Felberbauer. *Games with Purpose - Improving 3D Model and Land Cover Data using Crowdsourcing*. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, November 2012.

[20] Lois Fichner-Rathus. *Foundations of Art and Design*. Cengage Learning, 2011. ISBN 1111771456.

[21] John M. Findlay and Iain D. Gilchrist. *Active Vision: The Psychology of Looking and Seeing (Oxford Psychology)*. Oxford University Press, USA, 2003. ISBN 019852479X.

[22] Michael Frotscher. *Taschenatlas Anatomie 03. Nervensystem und Sinnesorgane*. Thieme Georg Verlag, 2009. ISBN 313492210X.

[23] E. Bruce Goldstein. *Sensation and Perception*. Cengage Learning, 2010. ISBN 0495601497.

[24] Google Inc. Google 3D Warehouse, 2012. URL `http://sketchup.google.com/3dwarehouse/`.

[25] Google Inc. Google Images, 2012. URL http://images.google.com/.

[26] Stefan Gustavson. Simplex noise demystified. Technical report, Linköping University, Sweden, 2005.

[27] Gwap. ESP Game, 2012. URL http://www.gwap.com/gwap/gamesPreview/espgame/.

[28] Robert M Haralick, Stanley R Sternberg, and Xinhua Zhuang. Image Analysis Using Mathematical Morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):532–550, July 1987.

[29] Christopher G. Harris. Dirty Deeds Done Dirt Cheap: A Darker Side to Crowdsourcing. In *2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing*, pages 1314–1317. IEEE, October 2011.

[30] DC Hood and MA Finkelstein. Sensitivity to light. In *Handbook of Perception and Human Performance*, chapter 5, pages 1–66. John Wiley & Sons Inc, New York, 1986.

[31] Xiaodi Hou, Jonathan Harel, and Christof Koch. Image Signature: Highlighting Sparse Salient Regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1): 194–201, July 2011.

[32] Ian P. Howard. *Perceiving in Depth, Volume 3: Other Mechanisms of Depth Perception*. Oxford University Press, 2012. ISBN 0199764166.

[33] By Jeff Howe. The Rise of Crowdsourcing, 2012. URL http://www.wired.com/wired/archive/14.06/crowds.html.

[34] Amazon.com Inc. Amazon Mechanical Turk, 2012. URL https://www.mturk.com.

[35] Renato Keshet. Dilation - Wikipedia, the free encyclopedia, 2008. URL http://en.wikipedia.org/wiki/File:Dilation.png.

[36] Renato Keshet. Erosion - Wikipedia, the free encyclopedia, 2008. URL http://en.wikipedia.org/wiki/File:Erosion.png.

[37] Renato Keshet. Opening - Wikipedia, the free encyclopedia, 2008. URL http://en.wikipedia.org/wiki/File:Opening.png.

[38] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with Mechanical Turk. In *Proceeding of the twenty-sixth annual CHI Conference on Human Factors in Computing Systems - CHI '08*, pages 453–456, New York, New York, USA, 2008. ACM Press.

[39] Helga Kolb. Circuitry for Rod Signals Through The Retina – Webvision, 2011. URL http://webvision.med.utah.edu/book/part-iii-retinal-circuits/circuitry-for-rod-cells-through-the-retina/.

[40] Florian Lang and Philipp Lang. Visuelles System. In *Basiswissen Physiologie*, Springer-Lehrbuch, chapter 17, pages 389–408. Springer Berlin Heidelberg, 2007.

[41] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. *ACM Transactions on Graphics*, 24(3):659–666, July 2005.

[42] Eric Lengyel. *Game Engine Gems, Volume One*. Jones & Bartlett Learning, 2010. ISBN 0763798967.

[43] Ce Liu, Jenny Yuen, and Antonio Torralba. SIFT flow: dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33 (5):978–94, May 2011.

[44] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, January 2007.

[45] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[46] D Marr, S Lal, and HB Barlow. Visual Information Processing: The Structure and Creation of Visual Representations. *Philosophical Transactions of the Royal Society of London*, 290 (1038):199–218, 1980.

[47] Niloy J Mitra, Hung-Kuo Chu, Tong-yee Lee, Lior Wolf, Hezy Yeshurun, and Daniel Cohen-Or. Emerging images. *ACM Transactions on Graphics*, 28(5):163–171, December 2009.

[48] Donn Morrison, Stéphane Marchand-Maillet, and Éric Bruno. TagCaptcha. In *Proceedings of the international conference on Multimedia - MM '10*, MM '10, page 1557, New York, New York, USA, 2010. ACM Press.

[49] Marti Motoyama, Kirill Levchenko, Chris Kanich, Damon McCoy, Geoffrey Voelker, and Stefan Savage. Re: CAPTCHAs: understanding CAPTCHA-solving services in an economic context. *Proceedings of the 19th USENIX conference on Security*, USENIX Sec: 1–18, 2010.

[50] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, January 1965.

[51] Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. Fast hierarchical importance sampling with blue noise properties. In *ACM SIGGRAPH 2004 Papers on - SIGGRAPH '04*, volume 23, pages 488–495, New York, New York, USA, 2004. ACM Press.

[52] Ellen E. Pastorino and Susann M Doyle-Portillo. *What Is Psychology? Essentials*. Cengage Learning, 2012. ISBN 1111834156.

[53] Robert Pietsch. Statistische Herausforderungen sozialwissenschaftlicher Studien: Bradley-Terry-Luce Modell, 2011. URL http://www.statistik.lmu.de/~thomas/Lehre/wise1011/SeminarSozi/RobertPietsch.pdf.

[54] Princeton University. WordNet, 2010. URL http://wordnet.princeton.edu/wordnet/citing-wordnet/.

[55] Joel Ross, Lilly Irani, M. Six Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the crowdworkers? In *Proceedings of the 28th of the International on Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '10*, pages 2863–2872, New York, New York, USA, 2010. ACM Press.

[56] Ryan Schmidt, C Grimm, and B Wyvill. Interactive decal compositing with discrete exponential maps. *ACM Transactions on Graphics (TOG)*, 2006.

[57] Iwan Setyawan and Reginald L. Lagendijk. Human perception of geometric distortions in images. *Electronic Imaging 2004*, 5306:256–267, June 2004.

[58] Stefan Tahler, Katharina Siorpaes, Elena Simperl, and Christian Hofer. A survey on games for knowledge acquisition. Technical report, STI Innsbruck, University of Innsbruck, 2011.

[59] TurboSquid. TurboSquid, 2012. URL http://www.turbosquid.com/.

[60] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics* -, ACL '94, pages 133–138, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[61] Jeff Yan and Ahmad Salah El Ahmad. Usability of CAPTCHAs or usability issues in CAPTCHA design. In *Proceedings of the 4th Symposium on Usable Privacy and Security - SOUPS '08*, SOUPS '08, page 44, New York, New York, USA, 2008. ACM Press.