VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

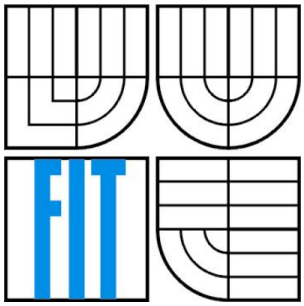
CHANGING OBJECT APPEARANCE BY ADDING FUR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. MARTIN PRAŽÁK
AUTHOR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

CHANGING OBJECT APPEARANCE BY ADDING FUR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. MARTIN PRAŽÁK

VEDOUCÍ PRÁCE
SUPERVISOR
Doc. Dr. Ing. PAVEL ZEMČÍK

EXTERNÍ VEDOUCÍ PRÁCE
EXTERNAL SUPERVISOR
Dr. ERIK REINHARD

BRNO 2008

Assignment details:

Licence agreement:

Abstract

The aim of this thesis is to demonstrate the feasibility of rendering fur directly into existing images without the need to either painstakingly paint over all pixels, or to supply 3D geometry and lighting. The fur is added to objects depicted on images by first recovering depth and lighting information, and then re-rendering the resulting 2.5D geometry with fur. The novelty of this approach lies in the fact that complex high-level image edits, such as the addition of fur, can successfully yield perceptually plausible results, even constrained by imperfect depth and lighting information.

A relatively large set of techniques involved in this work includes HDR imaging, shape-from-shading techniques, research on shape and lighting perception in images and photorealistic rendering techniques. The main purpose of this thesis is to prove the concept of the described approach.

The main implementation language was C++ with usage of wxWidgets, OpenGL and libTIFF libraries, rendering was realised in 3Delight, a Renderman-compatible renderer, with the help of a set of custom shaders written in Renderman shading language.

Keywords

Image-based material editing, IBME, high dynamic range, HDR, shape detection, fur, rendering, image-based lighting, image processing, C++, OpenGL, wxWidgets

Citation

Pražák, Martin: Changing Object Appearance by Adding Fur, *Master's Thesis*, Brno, FIT BUT, 2008.

Declaration

I hereby declare that I wrote this diploma thesis independently under the supervision of doc. Pavel Zemčík.
The main part of this work was made during the exchange study at the University of Bristol, under the supervision of Dr Erik Reinhard. All the sources, on which this work is based, are stated in the reference section.

.....
Martin Pražák
19.5.2008

Acknowledgements

I would like to thank Dr Erik Reinhard and doc. Pavel Zemčík for their guidance, supervision and comments about this work. Their support and help made this work possible.

I would also like to thank Tania Pouli for her help with the final renderings and the will to continue this research in her further studies.

Content

1 Introduction

Computer science and computers, in as many different forms as they might have, became a part of our daily lives. We have them in our pockets in a form of a mobile phone, use them to get and process information for our daily tasks, and even if we don't touch them directly, every newspaper, every building and practically everything we can buy or get is to some extent connected to or designed using computers.

But computers in their very basis are only machines processing numbers. The more complex this number processing is, the larger is a need to interpret the results the way we can understand, evaluate and use in our daily life. And this was the very beginning of computer graphics, a computer science field trying to find the most suitable way of representing results of complex algorithms in the most natural way for humans. Computer graphics has come a long way since then, improving reality in films where actual realisation of a scene would be extremely expensive or even impossible, creating virtual worlds and whole alternative realities in computer gaming industry.

The need to represent such worlds (but not limited only to this purpose) led to the development of rendering systems designed to transform a 3D representation of a scene to its 2D alternative via an approximation of light interaction with the given environment. However, the physical correctness of such process is not the most important issue when the visual plausibility together with the possibility to influence and change the actual result are more desired, or when the processing speed is a strong constraint.

In order to achieve the visual plausibility of the result, to significantly reduce processing time requirements, and to simplify the rendering process, we need to relate the simulation properties to the human visual system. There is no need to simulate the effects which are not visible in the final image, whilst other visually very important features can remain missed. For this reason the psychophysics research field is very important, merging psychology and neurology with the computer graphics in order to give necessary feedback on developed methods and algorithms.

Another relatively large field directly involved in this thesis is computer vision. As the name suggests, its purpose is exactly the opposite of computer graphics, trying to transform the images back into some form of scene description, or to assign semantic meaning to objects in scene, helping the computer to “understand” the real world.

The image-based material editing (IBME) field, into which this thesis is set, is a combination of approaches and methods originating from all previously mentioned fields. As opposed to the most of the computer graphics algorithms it builds on perceptual evidence in the first place, connecting image processing algorithms of computer vision with rendering methods in the way that may seem mathematically and analytically incorrect at first sight, but it leads to perceptually correct results. IBME has already found its practical use in 2D image editing software, where it is used for creating simple automatically made previews, or to simplify complex and painful tasks without the need of time-consuming manual drawing work.

The aim of this work is to extend the existing set of IBME techniques by the possibility of semi-automatic painting of hair or fur on the surface of an object, showing that complex high-level image edits, such as adding fur, can successfully yield perceptually plausible results even in the presence of imperfect shape extraction and lighting conditions. The base of this hypothesis lies in the

psychophysics research, which proved that the high-frequency content together with “diffuse” lighting conditions (applicable for most real world images) can significantly reduce the requirements laid on a shape-from-shading algorithm determining the shape of an object from a 2D image, by visually masking the inaccuracies and imperfections.

This text consists of five chapters and five appendices. The introductory chapter briefly describes the aim of the project, used techniques and expected results. The second chapter introduces state of the art research in fields applicable to this thesis. Because of a large extent of possible connected research only the most relevant work is described. The third chapter analyses the research mentioned in the second chapter in the context of this thesis; the implementation details are described in the fourth section. Finally, the fifth chapter summarizes the research results and suggests possible further work. The sixth chapter contains list of references for research used in this text and the final seventh chapter contains appendices with detailed descriptions of particular aspects, that were too extensive to be included in the main text.

This work is a research project meant as a “proof of concept” of the described hypothesis. The development of useful methods involved many changes in program structure when a particular method proved to be insufficient, therefore no exact design specification was possible in the first stages of the research. Therefore it is not included in this text and the final class structure is added to appendices. The nature of this work also suggests that no user-friendly user interface is required or expected, and for this reason the main application does not have any form of graphic user interface nor provides a generally usable solution to the problem, which would need to include better ways to specify the user-defined parameters of the algorithms.

2 State of the art

In the most natural way the computer graphics as a field is divided into two main domains – 2D and 3D graphics, representing the two different ways existing in the real world. Because all commonly used display devices can only display two-dimensional images, research attention has focused on the ways to convert a 3D representation into such 2D image, and to make this resulting image as realistic as possible. The conversion process, which for the purpose of realistic appearance usually includes a simulation of light scattering in the environment, is called rendering. But the division between 2D and 3D world is not only crossed by this process, some more creative approaches, especially those involving also computer vision, can cross the border in a more creative way.

The need for such methods is very pronounced in the requirement to enhance and change the content of a real photograph, set of static images or even videos, where a 3D description of the whole scene is not available and would be extremely hard to obtain. One solution, which is the usual option in the film and advertisement industry, is to use a fully manual interface (e.g. Photoshop, GIMP etc.) and a skilled artist to change the image the way we need. But this is an extremely long, painful and expensive process. For this reason any semi-automatic approach, which would rather than mathematically and physically correct results produce a perceptually plausible and acceptable image, would be very useful.

Such processes are well developed for adding virtual objects into a scene, as we can see in almost every action film in our cinemas. The rendering methods together with careful imitation of original lighting condition and guided composition to original scene produce extremely realistic results at the cost of large amount of manual work. The image-based material editing field, the subject of this thesis, is aimed to provide a set of algorithms, which can help with part of this manual work by exploiting the limitations of human perception and allowing to change the material of objects in an image.

Many fields need to be used in order to achieve the goal of this thesis – to change the material of a particular object in the scene to appear "furry", and most of them are extensive enough to deserve a book of their own. For this reason the next chapter describes only research directly relevant and used in the development of this work together with references to literature, providing more details about every aspect of a particular field of research.

2.1 High dynamic range imaging

High dynamic range imaging (HDRI) is a relatively new and rapidly developing field of both computer graphics and computer vision, which has the potential to expand into many areas of these fields. The main characteristic is the extension of luminance values range, or dynamic range, of the represented data, allowing to capture much larger interval of real world luminance values without under- and over-exposed regions and to avoid quantisation errors which would become more pronounced using naïve dynamic range extension with current graphic formats. For this reason, it can be compared to extension of image resolution observed so much in recent years development, providing even more detailed and accurate description of the real world.

Because of enhancements in data range, new approaches to acquisition, representation, data compression and display need to be used, as well as new techniques for capturing this type of data and

working with them on classical display devices with very limited dynamic range.

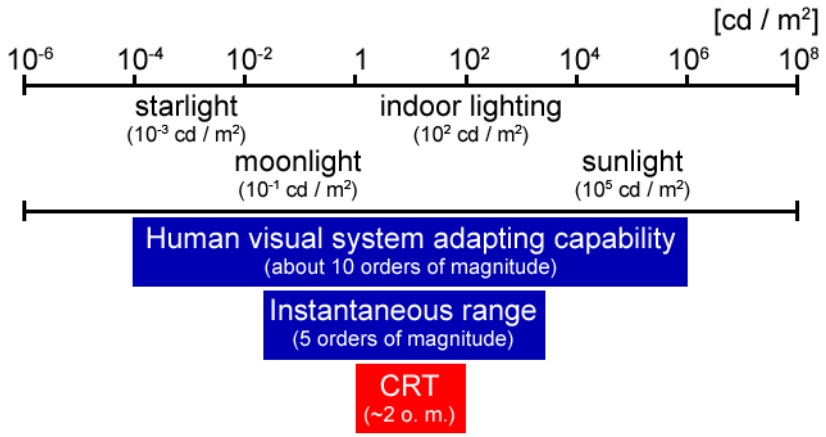
Even though the advantages of HDR for some fields are obvious, its usage in the topic of this work would be limited to two cases – first is its usage in input image where HDR can provide more detailed data with less noise and without the loss of information in underexposed and overexposed regions, and second for determination of lighting of the resulting scene in order to preserve and duplicate the same lighting conditions as the rest of the image already has [29]. Both cases are described in more detail in following sections.

Even though the actual research interest in HDR imaging has only a short history, its first practical application can be traced back some 15 years ago to Radiance light simulator / rendering system [52]. This system became a legend in computer graphics research, because for the first time researchers had an open-source tool and framework for their work, with the potential of professional rendering system together with physically correct light behaviour simulation, implementing most up-to-date rendering techniques. The need of physical correctness led to a new image description storing every pixel in terms of its colour as well as physical luminance units, and a set of methods for transforming this representation to classical image formats. In today's terms this is called HDR file formats and set of tonemapping algorithms.

Since then, the HDR research continued in many ways and gradually increased during past years, extending from representation of light simulation results to a field influencing large section of computer graphics and computer vision, and now even expanding to hardware in form of specialized capturing and display devices with extended dynamic range

2.1.1 Dynamic range in images and real world

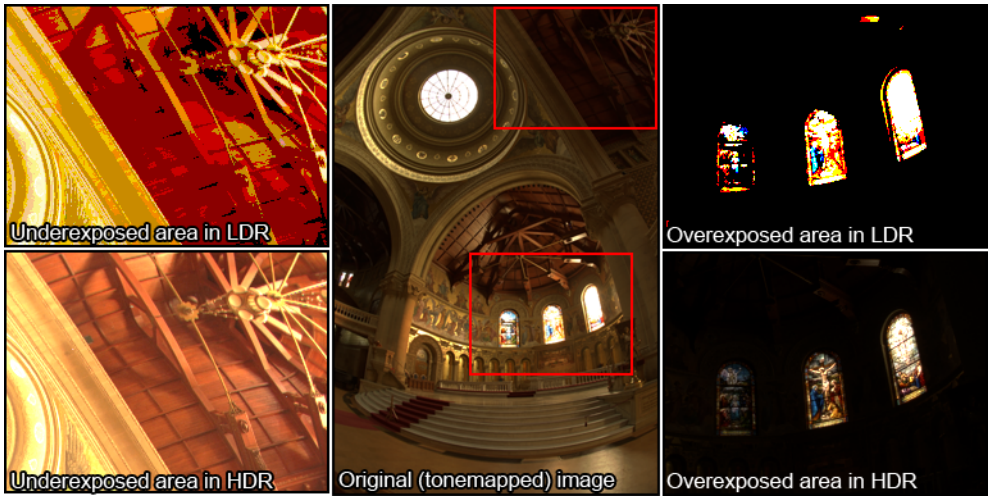
The range of luminance values in real world is extremely large. The variation from direct sunlight down to starlight can vary in order of 14 magnitudes, and even though we cannot perceive this whole interval, the human eye can still adjust to very impressive 10 orders of magnitude and see whole 5 orders of magnitude at one time. For the comparison, the best CRT monitors, which can have larger luminance range than LCD displays, are able to display only 2 orders of magnitude [45]. This fact can be nicely presented on a logarithmic scale, as shown on image 2.1.



Img. 2.1 - Luminance ranges in real world, shown in logarithmic scale (based on data from [45])

A naïve approach to overcome this issue would be to simply rescale the range of a common file format

to cover the whole interval of luminance in the scene. The problem with this is the fact, that a usual 24-bits per pixel image use only 8 bits, or 256 levels, to describe each channel. While this is enough for a photograph with limited range or an image displayed on usual monitors, for the HDR images with range many times larger the quantisation errors would become very pronounced and information in possibly cropped regions would be lost (as shown on image 2.2). For this reason it is necessary to use some form of floating-point representation, which would not imply the cropping for meaningful values and limit the quantisation issues, and to specify algorithms for conversion between HDR representation and its low-dynamic-range (LDR) equivalent.



Img. 2.2 - Information content and quantisation errors in HDR and LDR images (based on data from [45], image courtesy of Paul Debevec [13])

2.1.2 Information representation in HDR images

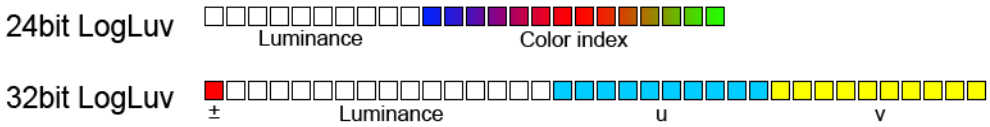
As discussed in previous section, the values range of HDR images has to be much larger and the floating-point data representation is necessary in order to avoid quantisation issues. But the representation of such numbers in computer science, which consists of mantissa and exponent, is very hard to directly compress (and the resulting image file takes too much storage space), because even small difference between two values can lead to relatively large change of binary representation of such number and most of the classical compression algorithms then prove to be ineffective [5].

One of possible solutions, which was introduced in development process of Radiance system, was based on observation, that luminance range in each channel is very close. For this reason it is possible to store RGB mantissas, each with 8 bits per pixel, and only one common 8-bit exponent [51], leading to a 32-bit RGBE representation with systematic error less than 1% on range 10^{-38} to 10^{38} [5]. By statistical analysis of the transformed HDR image it is quite obvious, that there is relatively high probability of same channel values in neighbouring pixels, allowing effective compression using simple run-length-encoding (RLE) algorithm (after proper reordering of values on a scanline) [51].

Img. 2.3 - Structure of one pixel in Radiance ".hdr" file format [45]

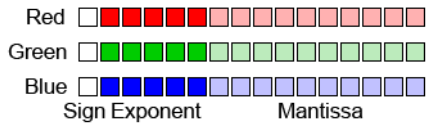
Another common approach, which actually builds on the same basis as Radiance file format, separates luminance channel, containing most of the information, by converting the image to a suitable colour

model. Compression itself is then performed only on luminance channel, whilst colour information is stored as is, using simple quantisation allowed by lower sensitivity of human visual system to this information [45]. This approach is used even for LDR information in the compression method of the broadly used JPEG file format. Practical implementation can be well described on HDR version of TIFF format, which uses LogLuv compression – Luv colour model for separating the luminance and colour information, with logarithmic transformation of the luminance channel. The TIFF implementation of this algorithm comes in two versions – 24-bit with 10 bits of luminance information and 14 bits serving as an index to a colour table, and 32-bit with 16-bit signed luminance value and separated 8-bit u and v channels (as shown on image 2.4).



Img. 2.4 - Structure of one pixel in LogLuv version of TIFF file format [45]

Other two important formats, which have the possibility of broad use in computer graphics are OpenEXR and HDR JPEG. OpenEXR, introduced with its open-source C++ library by Industrial Light and Magic in 2002, is a format with ambition of becoming industrial standard in the HDR field. Its basic structure is founded on IEEE 16-bit "Half" floating-point format (as shown on image 2.5), and the library encapsulates advanced compression methods to offer images with good compression ratio and without artifacts common for other HDR file formats.



Img. 2.5 - Structure of one pixel in OpenEXR file format [45]

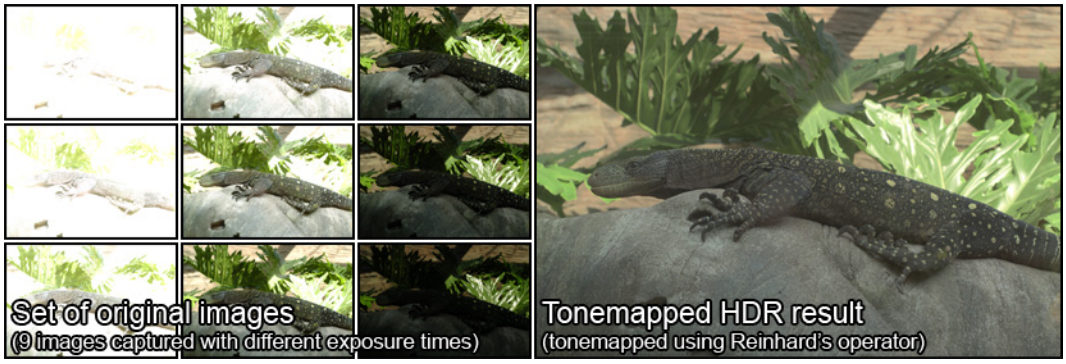
HDR JPEG is a classical JPEG (therefore backwards compatible with LDR JPEG) with 64 kB separated marker containing ratios between LDR and HDR information of the source image. LDR part then represents tonemapped version of the image, whilst undersampled HDR marker can be used to restore the full floating-point information contained in the original. The main problem with this approach is the limited storage size for HDR data, and even though its main drawbacks can be reduced by a specifically designed compression / decompression algorithm, for large images the resulting artifacts are too obvious.

The number of HDR file formats grew rapidly in last years and this thesis cannot contain description of all of them. For future reference and more detailed explanation please refer to [45].

2.1.3 HDR image acquisition and editing

To be able to fulfil the requirement of higher dynamic range together with the need to limit the noise introduced by their processing, new data acquisition approaches need to be used. This problem usually does not appear for results of light simulations from renderers, because such software has to use some form of HDR data structure internally. The technology of HDR cameras capable of capturing higher dynamic ranges at once is being developed, but for now the only way to capture luminance interval exceeding the ability of a digital camera is to combine a set of multiple exposures of the same scene into one HDR image.

The simplified description of this method is quite intuitive. Starting from the "longest" exposure (e.g. image captured using longest exposure time with keeping the aperture settings intact), which will contain many overexposed regions (containing almost no data), the algorithm fills these regions using data from a longer exposure multiplied by their relative exposure difference. Iterative processing the whole set towards the "shortest" exposure image will produce an HDR image using data from all the source images (as illustrated on image 2.6). This very simple explanation missed a few very important steps, like determination of camera response curve, aligning of the source images or dealing with "ghosts" caused by moving objects in scene. For more information about this topic please refer to [45].



Img. 2.6 - Set of LDR photographs and assembled HDR image (tonemapped) (source images provided by Dr. Erik Reinhard)

The most common practical usage of HDR imaging is image-based lighting, which utilize a specially arranged HDR photograph of a reflective sphere (called light-probe) exactly mapping lighting conditions of a point of given environment. For more information about this topic please refer to section 2.3.4 .

As the HDR imaging research continues, the support is added to more consumer software. Starting from renderers such as Radiance, Lightwave, Renderman etc. implementing image-based lighting, through specialized HDR software employing all mentioned techniques (HDRShop [47]), to the broadly used professional image editors such as Adobe Photoshop CS3.

2.1.4 Tonemapping

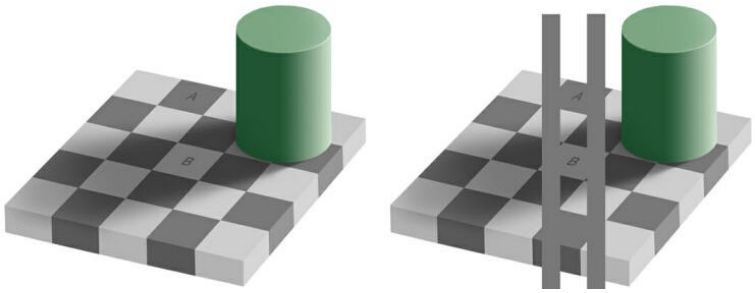
In order to display HDR image on an usual LDR display device, the luminance interval has to be reduced. Algorithms achieving this reduction are called tonemapping operators, and their goal is to compress the luminance interval in a way, which would not be perceptually unrealistic, while keeping most of the information contained in the full HDR version of the image. These two constraints proved to be hard to fulfil and for this reason many different solutions exist, spreading from very simple global operators (influencing every pixel of the image the same way) to very complex locally-adaptive approaches mimicking the behaviour of early human vision [45].

Most of these algorithms usually involve the transformation of image representation to a colour space which separates luminance and colour information as their first step (e.g. Yxy, Luv) [45]. The advantage of this preprocessing step is the fact, that the tonemapping itself then needs to be applied only to luminance channel, while colour information can be transferred from HDR image directly.

The first practically used application of tonemapping operators was implemented in Radiance [52] to convert the results of light simulation, described in physical units (cd/m^2) to displayable values.

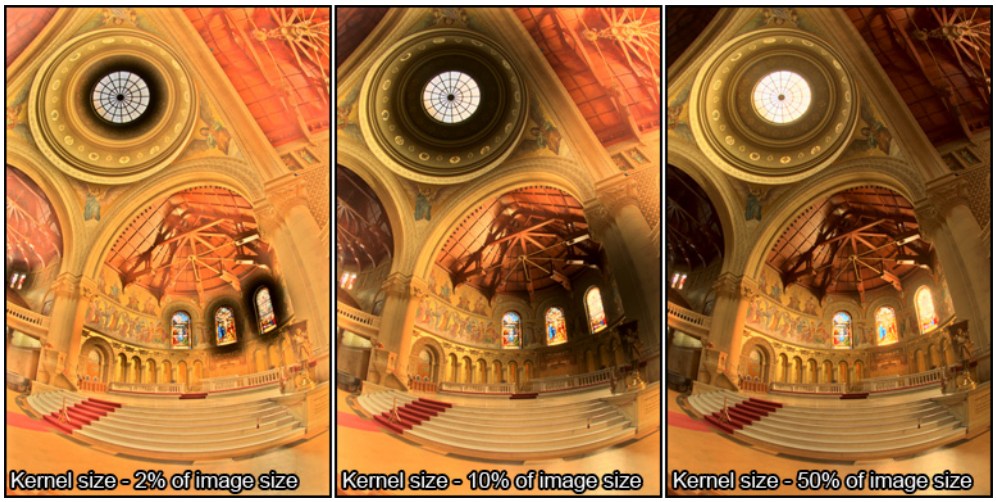
Even in this pioneering work in this field it was observed, that the linear transformation does not represent the perceptual experience well, leading to first logarithmic operators (based on S curve photographic film sensitivity) [45].

The hypothesis behind locally-adaptive operators is based on the fact, that some kind of local adaptation of perceived image is applied directly in human eye within early vision process, which compress the visual information into a thin stream sent to brain processing centres (classic example shown on image 2.7). If it would be possible to create the same preprocessing step in the tonemapping operator "skipping" the compression in the eye, it should provide the full HDR experience on LDR displays[45].



Img. 2.7 - A very famous example of local adaptation in early vision. The A and B fields on the checker board have the same colour, but our visual system is convincing us about opposite. [45]

First local operator, whose adaptation was based on Gaussian-blurred luminance channel, was Chiu's operator. Even though this operator (with proper parameters settings) performs relatively well for images with middle dynamic range, for larger interval of luminance values it creates very strong artifacts, typical for most local operators, described as dark halos (as shown on image 2.8) [45]



Img. 2.8 - Results of Chiu' operator with different kernel-size parameters (source image courtesy of Paul Debevec [13])

More perceptually-guided method of tonemapping, although originally not intended for HDR images, is Multiscale Retinex [28]. The word Retinex is a combination of words "retina" and "cortex", forming a term describing the early vision as a combination of processing in retina and primary visual cortex. This method separates image onto three scales and combines them back together it a way that

successfully compresses the luminance information in a perceptually correct way, while colour information is adjusted in order to correspond with local adaptation of the retinex. Even though this method works extremely well on LDR images, for which is was developed, it fails for large intervals of luminance in HDR images [45].

The multiscale decomposition approach is nowadays used in many local operators, separating the high-dynamic layer of image from the detailed one, compressing only the HD information, and combining the layers back. This leads to reduction of luminance interval in the whole image (which usually very differs between different objects and / or parts lit by strong directed light), while leaving the edge and detail information intact. The classical example of such operator is Bilateral Filter [15], which actually performs to large extent as the described separation process.

Most of the mentioned techniques are implemented as plugin modules in HDR shop [47], but some of them also made their way to more commonly used Adobe Photoshop CS3, supporting full editing interface for 32-bit HDR images and a set semi-automatic global and local tonemapping algorithms. Another very popular application is to create "fake" HDR effects in real-time 3D graphics applications (e.g. light halos around very bright objects), improving realistic appearance of rendered results.

2.2 Reconstruction of 3D shape from a single image

The 3D shape reconstruction from a single 2D image, or shape-from-shading, is one of the main questions in computer vision from the very beginning of this field. Even though new approaches and solutions are still developed, it is already proven that there can be no general solution to this highly underconstrained problem due to high number of unknowns and insufficient input information [60].

On the other hand many relatively good solutions are already found for multiple source images and animations, as in stereo vision or motion-flow methods, and more research in these fields promises very interesting results [20]. Another approach can be based on implying more constraints on source image, or using a semi-automatic approach with user help in geometry reconstruction process, again leading to promising results (e.g. [38]).

In the language of mathematics these algorithms can be explained in terms of conversion of input luminance function $I(x, y)$ as a function of pixel position (alternatively the separated RGB components can be used) into a 2.5 shape description or "image with depth". The result then can be expressed as depth function $Z(x, y)$, surface normal function $n(x, y) = (n_x, n_y, n_z)$, surface gradient $g(x, y) = (dz / dx, dz / dy)$ or surface orientation [60]. The most complete description is the first one (depth function), because other descriptions can be derived from it using simple mathematical means. Other representations may yield an "unconstructable" resulting shape, in the meaning of consistent continuous surface.

To lighten up the claims from previous paragraphs it is worth mentioning, that there are approaches allowing a good shape reconstruction for images with some particular properties. As one example can be used an algorithm using the depth-of-field information inside image, exploiting the fact, that this image is a convolution of a sharp image with a 2D Gaussian function having sigma parameter directly related to depth of that particular segment of image [6]. The obvious drawback of this approach is in the fact that its accuracy directly depends on the amount of distance blurring in the source image and the ability of contrast-based sharpening algorithm to determine the sharp version of the input.

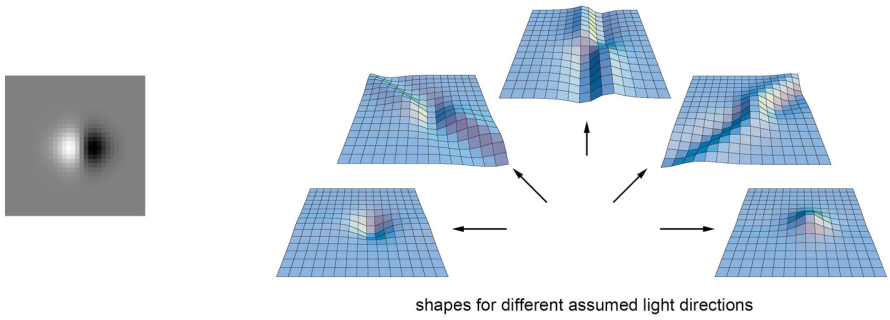
Recent psychophysics research shows that despite of the assumptions made by many scientists in shape-from-shading field, even our visual system is not capable of determining the proper object shape based only on shading information [32]. For this reason the psychophysics research focused attention on creation of a model describing of the ambiguity in vision processing as well as further information used in the later cognitive operation. However this "holy grail" of cognitive neuroscience is still far from realisation.

But for now it is well possible to use the preliminary conclusions of the ongoing research and exploit the ability of brain to fill "missing bits" of input information, providing a perceptually correct experience even with some intuitively important details in the image completely missed.

Closing this very short introduction to the shape-from-shading subject it is possible to conclude with statement, that there is no good general solution to this problem (due to many possible solutions [8]), and all the techniques in this field are limited to some special cases of input images, or imply strong constraints on the shape of the recovered object. Following subsections show a few examples of such methods together with their limitations and relevance to the subject of this work.

2.2.1 Shape-from-shading techniques

As it was already described in the introduction to this section, general shape-from-shading (SFS) techniques try to solve a highly underconstrained problem, transforming image shading (or colour information) described as a luminance function $i(x, y)$ into a 2.5D depth representation function $Z(x, y)$, surface normal (n_x, n_y, n_z) or surface gradient $(dz/dx, dz/dy)$. Most of such methods use a simplified Lambertian lighting model to construct a set of solvable equations. Despite of its strong assumptions about diffuse nature of input object lighting and direct illumination, the simplest mathematical description (having light direction as a input parameter) of this problem includes a solution of one linear equation with 3 unknowns (surface normal) or a non-linear equation with 2 unknowns [60]. Usage of more complex lighting models can simplify the process for a material with known properties, but for a real-world case it only leads to even more unknown variables in these equations.



Img. 2.9 - A demonstration of different possible solutions of a very simple image using a shape-from-shading algorithm [21]

Generally the depth recovery approaches can be divided into four categories differencing by constraints implied on the resulting shape and the algorithm used for solving the describing equations. Minimization approaches obtain the solution by minimizing an energy function, propagation approaches propagate the shape information from a set of singular surface points to the rest of the image, local approaches derive shape based on the assumption of surface type and linear approaches

compute the solution based on the linearisation of the reflectance map [60].

A nice demonstration of a number of possible mathematical solutions describing even a very simple input image with unknown light direction (as common for real world images) and a Lambertian surface was shown in [21] (image 2.9). This paper shown a possibility of a perceptually-guided probabilistic model allowing a selection of the most likely solution of light direction for an input image, preparing ground for even more perceptually-based approaches described in the next section.

Because all algorithms described in this section would use all information available in the input image in order to make the shape estimation more robust, a connection of this field with HDR can prove to be very beneficial. The lighting model-driven estimation would then be capable of deriving the light direction more precisely, because no overexposed regions exist in HDR image, as well as more accuracy in shaded regions (together with no quantisation errors), may make these algorithms more effective [29]. Another important property of HDR images it she fact, that diffuse and specular reflections are expressed in luminance values directly as a sum of these two components, allowing an algorithmical separation of these compounds and their separated usage [29].

As well shown and described in the Shape from shading: A Survey [60] paper, every one of these approaches perform well for some specific type of input images, but because none of them is generally usable for this thesis, please refer to this paper for more complete description of each group of methods. The mathematical description of multiple solutions problem and bass-relief ambiguity can be found in [8].

2.2.2 Psychophysics research on shape perception

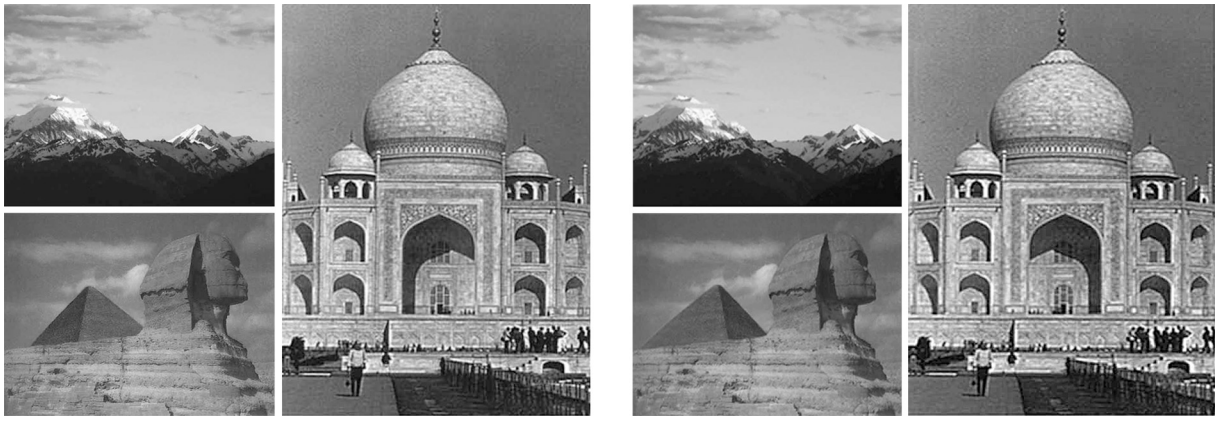
Psychophysics and perception research are for a long time regarded as important evidence in computer graphics, evaluating results of new algorithms and helping to focus on the most important issues in order to make images “look nicer”. Even though there can be no discussion about their importance, there are just a few methods that are actually directly built on top of perceptual evidence. One of these is RGB colour model used in most of the display devices, exploiting the fact that human eye can only perceive three different wavelengths of light to distinguish colours and therefore there is no need to simulate / display real white light with infinite number of wavelength.

The shape perception research results show, that the same problems of shape recovery as faced in SFS algorithms are addressed in human visual system. Possible solution for an image is not unique [8], and results for different participants can differ rapidly leading to very different resulting shapes, even though the actual semantic meaning is recognized correctly [30], [31].

This shows the fact, that human brain has to use more cues than just the shading in the image, like connection of perceived object with an already known object with similar shaping, leading to the question of how much the information in the input image actually need to be preserved in order to keep the same impression of the shape. For a complex lighting conditions this task becomes even more obvious – the complexity of the determination of lighting together with object shaping becomes so high, that the visual system actually has to provide most of the information about the shape, and the actual meaning of the shading in the image can be limited to a very simple “dark-is-deep” paradigm [32].

As to the lighting conditions in the scene, which are supposed to be directly related to the perception of object shape, for real-world lighting and a topologically complex object the errors in reflection from surface are not obvious [44], and the whole consistency of lighting can be well broken

without any noticeable results [41] (as illustrated on image 2.10). All these findings can make the shape extraction from an image a much simpler task.



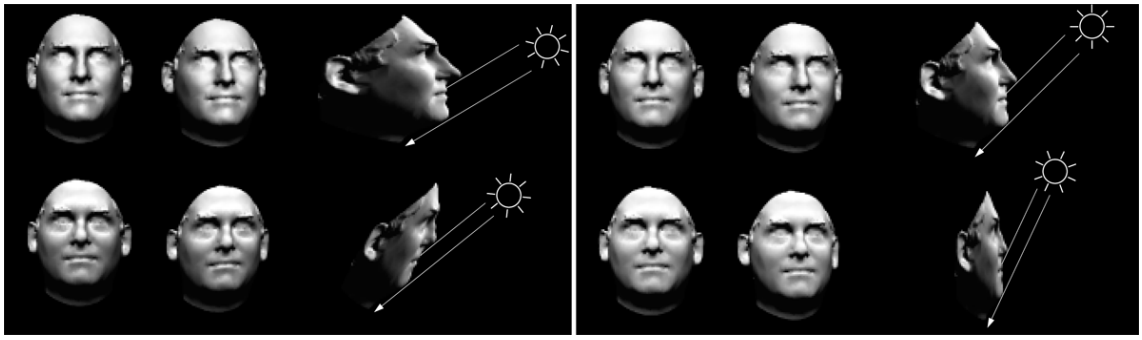
Img. 2.10 - An illustration of the visual masking of illumination inconsistencies as presented in [41] (left – images edited in order to contain a strong illumination inconsistency, right – original images)

2.2.3 Simplifications of shape reconstruction for material editing techniques

Image-based material editing (IBME) techniques draw heavily from previous chapter, and use the findings of psychophysics research in a practical application for automatic / semi-automatic change of materials on objects in a photograph or a video clip. The first step of such change is to recover a 3D shape of the original object, which can be then used for texture mapping, BRDF (bi-directional reflectance function, a description of light properties of a material) alteration, or even more complex material edits, and further rendering. The perceptual evidence justifying the heavy simplifications can be very extensive, this short chapter only mentions the most relevant research and its application.

The shape reconstruction simplifications are mainly influenced by three facts already described above. First – the solution is ambiguous [8] even in perceptual experiments [30], second – human visual system is not capable to see inconsistencies of lighting in real scenes [41], as demonstrated on image 2.10 (directly influencing the shape recovery algorithm), and third – in the IBME techniques the recovered shape is always rendered from the same viewpoint as was used for extraction. All these facts can help with simplification of the shape recovery, and more perceptual evidence can ease the masking of necessary inaccuracies, coming from the estimation algorithm, in the rendering process. This evidence contains again the inability to perceive illumination inconsistencies [41] (this time for the rendered result), the masking of surface reflection, which also does not have to be accurate [44] and the masking properties of high-frequency content (e.g. newly applied texture) [18].

To shine more light on the same-viewpoint property a demonstration picture (2.11) can be used. Because we can assume, that extracted values on x and y axis of the recovered 2.5D image are similar as the input coordinates in source data, only z axis values determined by depth-extraction algorithm may exhibit distortions and inaccuracies. And because the z axis is also similar to the view vector, these problems can be well masked in rendering process with proper lighting the same way as on the picture. And, again, this “proper” lighting is not a strong requirement [41], if we are not going to compare resulting images side-by-side.



Img. 2.11 - A model of human head with different transformations applied, leading to (with proper lighting) perceptually identical renderings [8] (every triple contains the rendering from the viewpoint of the applied distortion (left), the same rendering rotated by 5° (centre) and its side view (right))

A practical use of mentioned techniques was shown in Textureshop [17]. This software, which also included practically usable GUI and provided very strong evidence for correctness of IBME methods (more details in chapter 2.4.1), exploited the same-viewpoint property in order to discard the continuous shape constraint of SFS techniques. Normals of the surface of processed object were recovered using a lighting model and gradient orientation in the same fashion as in linear SFS methods, but instead of forcing the continuity constraint, these normals were clustered only into continuous patches (by directions of normals) and the texture was then applied in the same way as for polygons, using a simple form of mapping guaranteeing its continuity (see image 2.16). The shape recovered by this method was not consistent and did not fulfil the continuity constraint, but thanks to same-viewpoint property it has proven to be perceptually correct.

Another method demonstrated on glass-like materials in [29] was using the “dark-is-deep” paradigm [32] on its full extent, in combination with same-viewpoint property. The shape recovered by directly transforming the pixel luminance value into z axis value, was filtered using bilateral filter [49] (detailed description in section 4.2.3) to discard the texture information, and shaped using a simple analytically determined polynomial shaping function. Even though the recovered shape was only in very vague relation to the original shape, the results has yet again proven to be perceptually correct (see section 2.4.3).

2.3 Techniques using image-based approach

The common thing, and the source of the name, for all image-based techniques is the usage of images instead of other means for both rendering of the results and representing the geometry (to some extent), creating speed-ups, simplifications and sometimes even new effects, which would be very difficult to achieve using other means. The most well-known technique based on images is the texturing, replacing the fine geometry of the surface with an image, which then radically improve rendering quality with only a fraction of resources required for a full-scale detail modelling.

In a more general way the image-based techniques can be divided into two groups. The name of the first group is image-based rendering (IBR), and it is defined by usage of images as rendering primitives instead of polygons or other explicit or implicit 3D descriptions. In practice it is usually employed in some sort of image-geometry hybrid methods. Second group called image-based modelling uses images to drive the reconstruction of 3D geometric models, in an automatic or semi-automatic way.

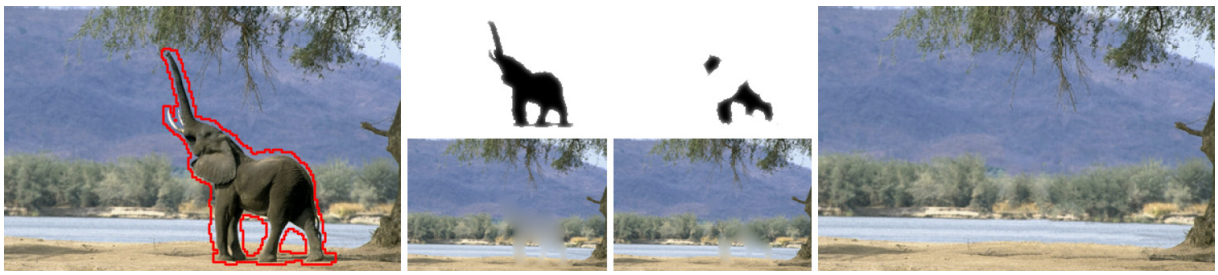
This chapter contains a list of basic principles involved in these fields, for more detailed descriptions please refer to [39].

2.3.1 Image composition and completion

Image composition and completion, although not directly considered as part of IBM or IBR, are two very related sets of methods dealing with integration of rendering results together or into other images, and handling the missing pieces of information after an image manipulation or removal of some objects from an image.

The groundwork of image composition was laid down in year 1984 by introducing the alpha channel and alpha blending operators [43]. These revolutionary principles changed the computer graphics and are still used without many changes even today in almost every domain of computer graphics. The image completion on the other hand is a growing field of methods allowing to fill missing parts of images with information consistent with their surrounding. For this reason they are often used in IBMR methods for filling the parts of scene, where image information is not sufficient or completely missing.

While image composition methods with alpha blending are well known, the image completion approaches deserve a more detailed explanation. Inputs of such algorithms consist usually of an colour image and a mask specifying the region of the image to be filled, optionally a user specified set of guiding lines or curves to make the completion more effective. By using a careful analysis of surrounding regions of the "hole" in combination with an iterative probability-based approach, the hole is filled with the pieces of the rest of the image (or synthesized fragments) fulfilling textural and orientation properties of the surrounding regions, leading to a perceptually and structurally correct image [46], [14].



Img. 2.12 - A fragment-based image completion process [14] (left – source image with mask for object removal, middle – two iterations of completion with confidence masks, right – completed image)

The relevance of composition methods to IBME is obvious – to insert the rendered result back into the original object background without noticeable artifacts, while image completion methods are used in changing object material to a refractive one (e.g. glass) to fill the hole after removing the original object from the source image with the same environment structure, which would be used as source of refraction information in process of rendering of the result.

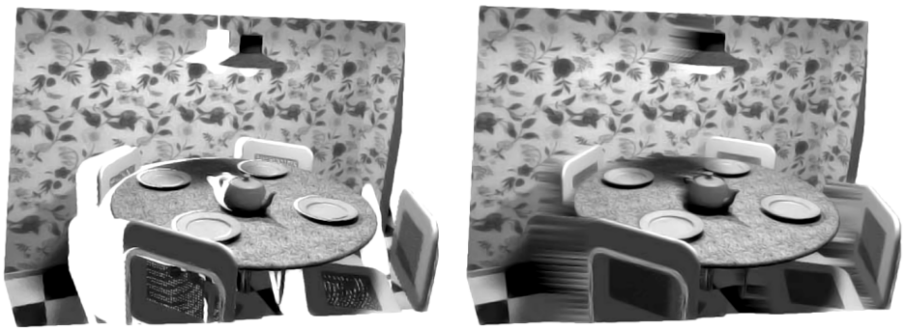
2.3.2 Image-based rendering techniques

Image-based rendering (IBR) techniques, as it was already mentioned above, use images as basic rendering primitives instead of other classical representations like polygon or implicit surfaces. Pure IBR methods use only images encoding light intensities for rendering novel views of the captured scene, and their main advantage compared to usual means is the fact, that scene complexity has no

effect on the rendering speed [39].

The most well-knows example of such approaches are panoramas and panoramatic viewers. Using a relatively simple photographic technique it is possible to capture a set of multiple exposures covering the camera surrounding within 360° viewing angle, stitch them together, and using a simple scrolling (for cylindrical panoramas) or sphere projection (for spherical panoramas) display only fraction of original image with the possibility of rotation, providing a very realistic representation of reality.

Hybrid IBR techniques employ also depth information stored together with intensities (results from rendering process can provide this information very easily) to enhance the panoramatic effects and also to introduce completely new ones, like image 3D warping [37]. Such techniques can prove to be usable for speed-ups of animation rendering processes, because they can offer a possibility to reuse information from previous frame in a new one and limit the rendering requirements only to differences between these frames.



Img. 2.13 - An example of 3D image warping with different reconstruction methods [37] (left – Gaussian cloud reconstruction, right – bilinear patch method)

By following the thought of images with depth, a more general representation involving multiple samples of both colour and depth information per one sampling ray / pixel would provide even stronger tool for object descriptions. Such method is called layered depth image (LDI) and allows a representation of complex geometry using only a form of raster image. But even LDI cannot contain enough information for a consistent object description, and a set of LDI acquired from different viewpoints is required for this purpose, forming together with a set of registration algorithms a image-based object approach [39].

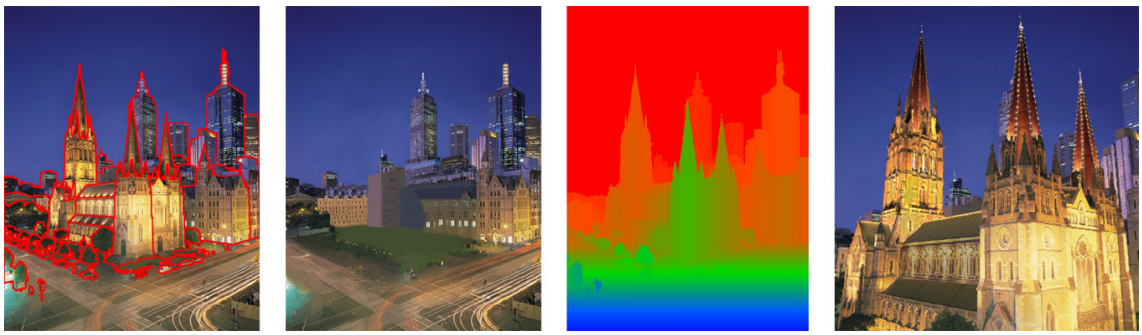
For a more complete guide of approaches mentioned in this chapter please refer to [39].

2.3.3 Image-based modelling techniques

The basic thought behind image-based modelling techniques is the possibility of shape reconstruction of objects in scene from multiple images and their later use in rendering. The mathematical models for such processes are very close to stereo vision and shape reconstruction from animations [20], whilst some methods may also include user interaction to allow more control over the reconstruction procedure.

Most of the successful applications of IBM techniques are designed for architecture modelling, using the possible simplifications yielded by their mostly orthogonal nature, and requiring additional user input (e.g. Image-based modelling and photo-editing [38]). Commercially successful applications in this field are PhotoModeler and PhotoBuilder, which both use a semi-automatic way for object

reconstruction from photographs, utilizing a user interface for specification of corresponding geometry parts of different photos together with orthogonal lines guiding the reconstruction process [39].



Img. 2.14 - An image-based modelling technique as presented in [38] (from left – image segmented into layers, clone-brushed hidden parts, false-colour rendering of pixel depth, rendering of a new viewport)

2.3.4 Image-based lighting

In the same way as the IBMR techniques are using the ability of images to describe complex geometry of an object, the image-based lighting (IBL) use the colour and luminance information to extrapolate and re-use light conditions of the scene, or to determine a set of probable light sources lighting the scene. The complex information included in an image can significantly improve the realism of rendering results.

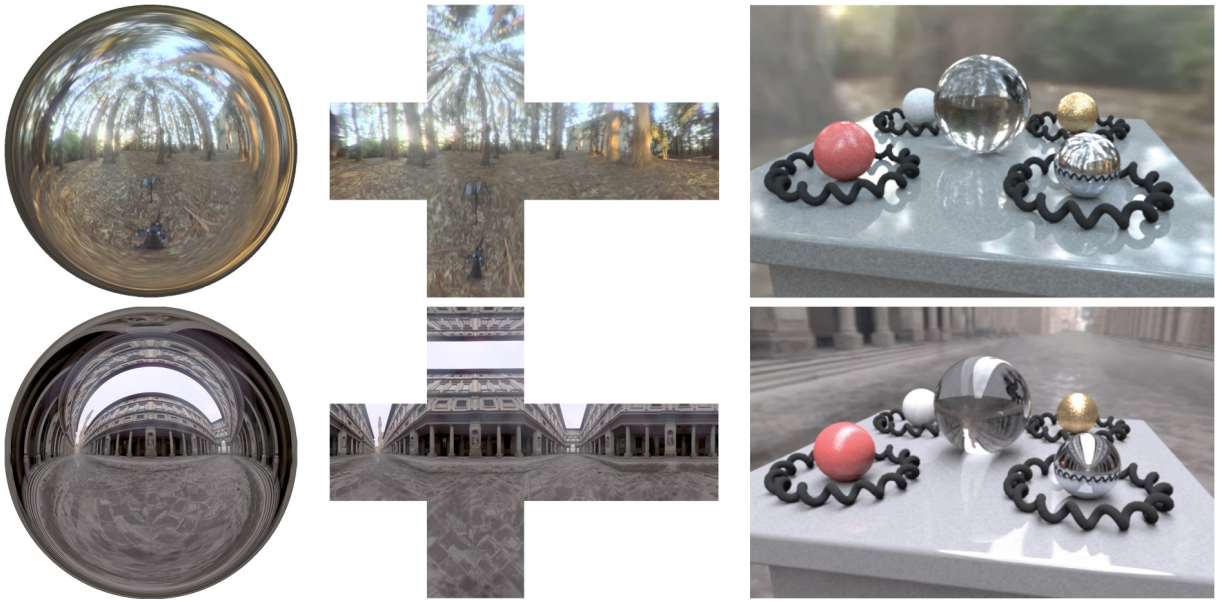
This information can be used in many ways. In a pure image-based way this information can be transferred to another image using a photometric approach, allowing for example to enhance unnatural and "flat" lighting of a flash photograph using a long-exposure (possibly blurred and noisy) photograph with original scene lighting [16], or by colour-transfer operator allowing to completely change the look of a scene (e.g. change a daylight photo into a moonlit one) [45].

A whole new dimension of IBL is opened by the high-dynamic-range (HDR) methods. The ability of HDR images to capture the whole lighting range in a scene or environment allows to actually use them directly as light sources and this way, using appropriate techniques, render an artificial scene with exactly the same lighting as the original real one.

At the bottom of these rendering methods stands a light probe – a special HDR photograph of a reflective sphere (image 2.15) in the position of the future artificial object, mapping completely the lighting condition in this particular point of the environment. The hypothesis is, that if the rendered into the scene object is small enough, and the distance of all objects of environment is large enough, the spherical reflective shape would reflect the incoming light to the camera, and the position of a point on sphere surface then corresponds to one direction of incoming light [45]. For the rendering purposes is this 2D image usually remapped using the computed incoming light vector to a more suitable description, like spherical or cubical environment map.

The derived map can then be either directly used as an emitting surface in radiosity-based rendering methods [45] (e.g. in Radiance [52]), or sampled using some form of importance sampling method (e.g. hierarchical sampling [40]) and transformed into a set of directional light sources [45]. This type of lighting became very popular in practical applications, because it allows to create a very complex scene lighting in an intuitive way, or to capture real lighting of a serving as background for a

composition involving both real and 3D animated elements. For this reason it is implemented in practically all modern renderers.



Img. 2.15 - Image-based lighting illustration [45] (from left – light probe photograph, remapping to a cubic environment map and rendering of an artificial scene with corresponding lighting)

Recent psychophysics research gives convincing evidence, that the conditions for correctness of the light probe based object size and environment distance may not be so strong. It actually shows, that the lighting can be very inaccurate or even inconsistent for different objects in the scene, but as long as it holds its very basic colour and structure (like the sunlight always coming from somewhere on sky), the inaccuracies are very hard to perceive [41] (as demonstrated on image 2.10). Another research in this field shows, that human visual system is also not capable of perceiving errors in patterns on reflective surfaces (as long as the surface has a more complex shaping) [44], which may be used to create a simplified approach for a scene lighting.

The simplifications directly taking advantage of perceptual results can be huge. Even though the condition of the scene lighting coherency still remains, for most real-world images captured as HDR we can use the image data directly as the light source for composited artificial object (as shown in [29]). To convert the 2D image to a 3D light directions a very simple half-spherical projection can be used, leading to an approximation of probable lighting, which holds the most important horizontal structure of the scene lighting. For more information about this particular topic please refer to [29], or section 4.5.1 of this thesis.

2.4 Image-base material editing

Even though material is (together with shape) the most important property of an object [18] and the modification can completely change its meaning, only little attention is paid to it in the perception field, as well as in computer vision field the material texture and light-interaction properties are only rarely used in terms of material recognition [2]. One of the causes of this fact is the large number of possible combinations of material structures together with specific lighting conditions implying the requirement of a very complex analysis method. The opposite can be said about computer graphics,

where material description forms a very large field of research covering many different techniques for the whole variety of possible material properties.

Building on the fact, that the shape of an object cannot be precisely determined from a single image (see section 2.2 or [8]) and that this determination is only vaguely influenced by its texture and lighting properties, we can change the material of this object by rerendering the extracted approximate shape with a new artificial material and compositing it back to the original image [43] without introducing any visible artifacts into the result [17]. The method, however, has to fulfil a set of basic condition, like continuity of the final texture or local gradient preservation, which are still a subject of ongoing perceptual research and do not have a complete and consistent specification available to date [29].

The state of the art of this relatively new field is directly connected to psychophysics and computer vision research on materials recognition, which both are still only beginning to provide practically useful results. For this reason the image-based material editing (IBME) methods include only handful of published papers, barely touching the possibilities of such approaches. Content of this chapter introduces the results of research related to this subject.

2.4.1 Change of texture

First and the most successful application of IBME is the texture change, which is directly based on its geometry masking properties [18] and this way hides possible extensive inaccuracies in the shape recovery algorithm. A short list of papers addressing this issue is described below, and includes applications to both static images and animations.

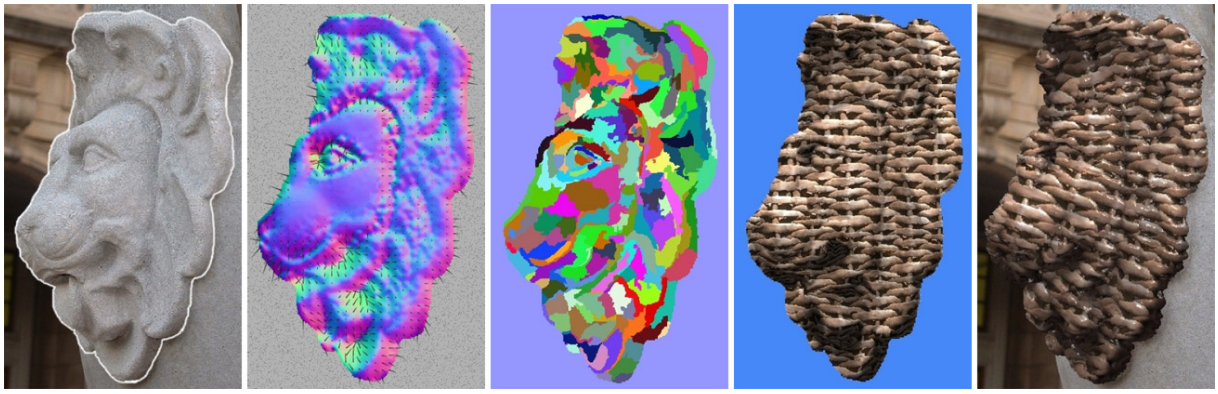
The usual approach utilized in most algorithms in this group is to use the Lambertian lighting model with direct connection of lighting gradients with surface gradients in order to extract surface normals, and to apply this information in a few different ways in order to achieve the retexturing goal.

The oldest paper, which was also the first demonstration of the effectivity the of IBME methods, described a tool called Textureshop [17]. The algorithm used in this paper consisted of a few steps (as illustrated on image 2.16). In the first step the object is isolated from its background using an user-defined mask and then the surface normals are determined in the fashion described above. To overcome the problem that the surface normals information is not a complete description of the object shape (as described in section 2.2), the normals information is clustered and the resulting set of patches is used for the texture mapping. It is important to note that such approach does not lead to a constructable shape, because it is not possible to connect the patches into a consistent geometric representation of an object, but because this information is not evaluated by human visual system, it does not lead to any noticeable shape distortions [17].

An extension to this algorithm was published by Zelinka et al. [59], introducing a combination of advanced object selection method with Gaussian mixture model (GMM), used for both object selection and for guiding the normal recovery. The GMM allowed to evaluate colour clusters in the input image, associated them with their reliability in terms of description of a full shape, and this information was then used to drive the normal recovery algorithm giving the possibility to use textured input images. This was not possible in previous approach, because the additional shading information introduced by the texture of the input object would lead to distortions of the extracted shape.

The usability of such methods for animations was demonstrated by Guo et al. [23], employing a Poisson-equation-driven algorithm for deformation simulation of the source objects with the

possibility to hold the structural information by its propagation to multiple frames of the animation.



Img. 2.16 - An illustration of texture change procedure as introduced by Khan et al. [17] (from left – source image, normals recovered using shape-from-shading algorithm, set of patches formed by normals clustering, a texture mapped to these patches and the final result with texture orientation distortion, displacement mapping and environment mapping)

A very different approach with similar results was introduced by Khan et al. [29] together with a whole set of new IBME methods described in many different parts of this thesis. Their algorithm incorporated the perceptual "dark-is-deep" paradigm (for more information see section 2.2.2 and mathematical explanation in section 4.2) with bilateral filter for removal of object texture (explained in section 4.2.3) and an analytically derived shaping function (see section 4.2.2). An HDR image was used as the input, because it was able to provide more lighting accuracy and more information in highlight regions leading to a smooth recovered shape. The depth function determined this way was then used to compute the texture projection offset for each pixel (exactly the same way as for transparency / translucency changes – equations 2.1 and 2.2).

2.4.2 BRDF change

The alteration of BRDF function describing the light-interaction properties of a material is more challenging problem than the texture change, because it results in smooth object surface without the textural masking of underlying shape. The only shape-inaccuracies masking involved in this case is the bass-relief ambiguity [8] and the fact that the result is viewed from the same viewpoint as the source image.

The described masking is hardly enough for a simple shape, where the shaping issues can be spotted much easier, but for more complex objects the high-frequency masking [44] is still present, giving the shape-recovery algorithm some space for imperfections. The first attempt to achieve the BRDF change was introduced by Khan et al. in [29], but the resulting algorithm had strong limitations and proved to be sufficient only for relatively small set of input images (results shown on image 2.17).

2.4.3 Transparency and translucency

The last successfully applied IBME transformation to date is the possibility to change a material into a transparent / translucent one, as demonstrated by Khan et al. [29]. The basic idea behind object transparency is the fact that human visual system is not capable of exact reconstruction of objects and environment refracted by a non-uniform shape, judging the correctness of the refracted pattern only by very vague structural criteria [29]. The refraction also introduce more high-frequency content into the

resulting image, leading to a successful masking of the shape imperfections [18]. Translucency represent a similar case in terms of the structural conditions, the only difference is the extensive light scattering inside the object body, which can be well simulated by using a blurred texture for determination of the refraction result [29] (can be also well demonstrated on reflection patterns [44]).

To mask the strongest shape recovery issues, which appeared at the highlight areas of the original shape, the reflection property of all refractive materials can be well used, pasting the original (and possibly enhanced) highlights from input directly to output, covering the problems by overexposed regions without any information. Because the reflected pattern errors are also extremely hard to perceive [44] and they again introduce more high-frequency content into image, can this processing step again improve the resulting visual experience.

The first step of the transparency / translucency material change (after locating the object boundaries) is to create the base of the refraction pattern, which needs to have the same structure as the surrounding of the considered object. For this purpose it is necessary to remove the original object from the image and fill this "hole" by a content constructed from the remaining regions of the object background. The image-completion algorithms (as described in section 2.3.1) can be well used for this purpose providing very accurate reconstruction of the background, but as Khan et al. have also proven, a very simple method mirroring the pixel values surrounding the shape on horizontal axis and blending the left and right mirrored region together using a very simple alpha blending can serve for this purpose as well, providing visually indistinguishable results in the resulting image [29].

The shape of the object is then determined using a "dark-is-deep" paradigm (as explained in section 2.2.2 and mathematically described in section 4.2) leading to a depth value for every pixel described as a function $D(x, y)$. From this value a spatial gradient field ∇d can be computed by differencing the depth values in neighbouring pixels, and the texture coordinates t_x and t_y are then computed using a formula:

$$t_x = x + s\nabla_x d \tag{2.1}$$

$$t_y = y + s\nabla_y d \tag{2.2}$$

where x and y are coordinates of the processed pixel, s is a linear scaling factor for adjusting the amount of distortions and $\nabla_x d$ and $\nabla_y d$ are x and y components of the gradient field value at the pixel position respectively.

Img. 2.17 - A demonstration of transparency / translucency and BRDF change [29] (from left – original image, transparency and aluminium-bronze BRDF function, translucency and nickel BRDF function)

To improve the accuracy of the result in the way discussed above, the highlights detected using a histogram method can be added to the resulting image as the last processing step [29]. This

mathematically and algorithmically very simple method, which is fully based and supported only by perceptual evidence, yields very convincing results, as shown on image 2.17.

2.5 Photorealistic rendering and Renderman system

Computer graphics is a large member of the pantheon of computer-based sciences. It can be itself divided into many large groups, some serving as the necessary parts of modern hardware and software, and some only developed as methods usable specifically in bustling entertainment industry. The 3D rendering techniques, used as a mean to transform some sort of 3D description of a scene into its 2D image equivalent, could be by large extent set into the second mentioned group. But this does not make them any simpler or less important, because the light simulation, on which most of those techniques are based, involves much of the physics research and very advanced optimisation algorithms.

Photorealistic rendering is usually considered as the main goal and the most realistic result achievable in the rendering process. To be more accurate a photograph is just one possible description of the light scattering inside an environment, and as a method is directly dependent on the physical properties of the capturing device, has very limited illumination range, and represents the wavelength spectrum as a mix of just three different primary colours. However, all these assumptions are usually based on human perception and therefore the photorealistic rendering creates results very close to human experience of the observed scene.

The extensive group of different approaches used to achieve a (to some extent) photorealistic result can be divided into three sets of methods varying by their computational complexity, level of physical simulation involved and the perceived accuracy of the result. The simplest and fastest is the object-based approach, rendering each object independently with only limited correspondence to physical process of light simulation. More physically based approach is a set of local methods. Local in this meaning can be described as the fact, that each pixel (fragment) of the resulting image is computed independently of all others. And the last and most realistic set can be called as global approaches, usually involving evaluation of the whole scene at once, simulating the light scattering the most physically-based way.

The explained division is very coarse and also has more relevance to the basis of the method than to the actual level of visual “realness” of the results. Especially the local methods involve large number of approaches enhancing (or even “cheating”) the results by visually important effects, like shadows, complex light scattering, reflections or refractions, which, although not directly based on simulation of light, can provide the perceptual experience similar to much more computationally expensive techniques.

For practical use the requirement of physical accuracy is of almost no importance, because the way we perceive the world around us is not by directly evaluating mathematical equations. The perceptual plausibility is the most relevant criteria for judging the rendering result, as well as the possibility to control the resulting appearance to very high extent (or at least up to director's wishes) for the applications in film and gaming industry.

2.5.1 The practical face of the rendering approaches

As it was already mentioned in the introduction to this chapter, the most realistic and also computationally slowest approaches are directly based on the simulation of light behaviour in the

given environment. They can be explained as the approximate solutions to the rendering equation [27], which formed a mathematical framework for description of the rendering process. But some visually important effects are very hard and time consuming to be realized in this way directly, e.g. the complex process behind penumbras as solved in radiosity methods [52].

Another completely different question is the controllability of the light simulation. Even though the results can be very physically accurate, it is not always desired, as we can see in most of the recent production of American film industry. The possibility to alter and customize the rendering process was obvious from the very beginning of realistic computer graphics, as well as the strong constraint of the rendering speed, which can prove to be very limiting for a large number of rendered frames. This is why the Renderman system [4] and the concept of shaders, developed by Pixar studios, came into the rendering field and soon after its first public release dominated the film-making enterprise.

The shader model (as described below) is an extremely strong tool for creation of the specialized content and "faking" the expensive physical simulations, but sometimes it could be simpler and faster to use a natural property of a more physically-based method in order to achieve the desired result. For this reason new versions of Renderman and new implementations of its interface also include more physically-based methods as raytracing or HDR lighting in order to make the development process of a special effect easier, faster and mainly cheaper. A good example of this approach is 3Delight [1], a professional and extended implementation of the Renderman interface, which is also available for free for personal use.

2.5.2 Shaders

The principle of shaders, an extremely usable way of describing visual effects, was introduced in the Renderman standard and in time became base of many modern approaches and hardware realisations used in all modern graphics card. The simplest description of a shader would show it as a small piece of programming code placed in an spot of the rendering pipeline defined by its type, influencing the rendering in an user-specified way. The original design was, however, stronger than most of the modern implementations, because its design was purely software-based and therefore not limited by any hardware-imposed restrictions.

Shaders usually are usually coded in a very specific programming language, which does not contain some of the usual programming structures, but on the other hand provides a large set of graphics-specific constructs, operators and data structures, allowing a simplified programming approach to graphics methods together with the high possibility of specialized optimizations not available in more general languages [4].

The specific points of interest for an offline (non-realtime) shader-based rendering system would be the strong focus on code optimizations together with the possibility of parallel processing without the any requirements on a specific algorithm design approach, and the focus on possible preprocessing, saving computation time during the main rendering process of the final high-resolution image.

2.5.3 Renderman rendering system principles

As it was already mentioned in the introduction to this section, Renderman is a rendering system aimed on, and also developed by, film industry animation studios. At the very base of it stands a object-based rendering API with the possibility of binding to the C programming language, not much different from the classic OpenGL model, but with the huge advantage of the user-defined shaders.

The Renderman system itself is just an implementation of the Renderman Interface [48], which is defining the syntax and semantics of both the API interface and the shading language. In time many different implementation of this interface emerged, some with GPL and OpenSource licensing (Aqsis, Pixie) and some orienting to the same customers as the original Renderman (3Delight [1]). With the Renderman Interface Specification [48] getting older, many different (and usually not-compatible) enhancements appeared, implementing the more recent methods in a way usable in the customized shader programming.

The complete list of the advantages of the Renderman system, which made it the choice for film industry, would take a few pages. Just to mention the most important ones directly specified in its basic design – the flexible and highly customizable shader-based interface, good support for parallel rendering (designed for rendering "farms"), support for text / binary / compressed input files or the input generated directly using C API without the need of any file-based scene description, from this property leads the possibility of embedding the Renderman as a library into an application, its very strong shading language, and TCL scripting support for rendering job planning and administration of the rendering process.

There are many other rendering systems available today, having the same strength as the Renderman Interface, like Blender, Lightwave, V-Ray, etc., but this thesis is focusing only on the Renderman as it is the most common option in practical applications.

2.5.4 Renderman shaders and shading language basics

A shader is a small compiled program inserted into a particular place in the rendering pipeline, performing user-specified computations and thus influencing the appearance of the resulting image. The shader type is determined by the exact location of its integration inside the rendering process (e.g. Renderman shader structure on image 2.7), as well as its interface and mandatory input parameters provided by the system.

The Renderman interface allows a shader to be much more complex than it is common in modern hardware-based shaders. Its shading language, although still just a limiter version of more general languages in terms of its flexibility, provide the possibility to define unlimited number of supporting functions, and defines only mandatory interface of the entry function, connected to the shader type. The required set of input parameters can be also extended by more optional variables, with values assigned either by their specified default value, or by the input scene specification / API calls, where every such parameter is identified by its name.

The list of supported shader types covers all parts of the object-based rendering process, and the more advanced "global" features are either supported using a combination of multiple shaders, or by a preprocessing step performed on the same scene description (e.g. shadow maps). The shader types as described in the Renderman interface are [4]:

Light source shaders – are given the position of a light source and the direction of a surface point from the light, and return the colour of the light originating at the light source and striking that surface point. There can be several light sources (instances of a light source shader) present in the scene, with the possibility to turn them on and off selectively for any surface in the scene. The light source shaders are also handling the shadowing of the light by using a preprocessed shadow map / deep shadow map (shadow map approaches are described in section 2.7.2).

Volume shaders – describe effects of light passing through an environment, both inside and outside an objects of a scene (corresponding to interior / exterior volume shader type). They get the source and destination point in space as input parameters and return the attenuation "colour".

Transformation shaders – are part of the geometric process rather than shading. They can be described as a non-linear alternative to transformation matrices, changing the whole geometric space without any relation to objects. Their input parameter is a point in space and have another point in space as the output. Some implementation may not describe them as shaders due to their connection to the input geometry rather than rendering.

Displacement shaders – are similar as the transformation shaders in terms of moving a point in space to different position, but instead of transforming the whole geometric space, they are designed to perturb the surface of an object point by point. For this reason their input parameters are the original point position on the surface, the surface normal, and can be extended also using parametric derivatives of the original surface curvature, texturing coordinates etc. They output a position of the updated point in space.

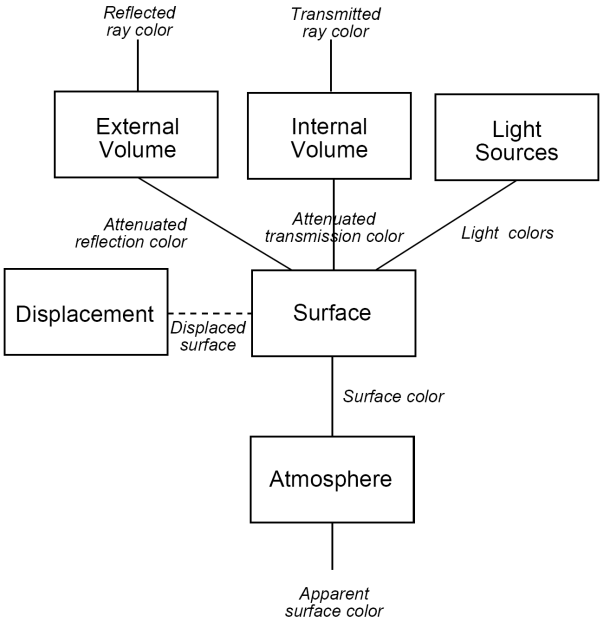
Surface shaders – describe the light scattering on a surface of an object by computing the reflected light intensity and colour in a particular direction. They receive the reflection direction, point on the surface, orientation of the surface on that point and the set of light sources in the scene on input, and return the integrated colour value considering all light sources (or possibly only a subset of them) as the output.

Imager shaders – complete the collection of Renderman shaders. Their job is to transform floating-point pixel colour and alpha values, as returned from the rendering process, using only a colour, alpha and pixel position information, into the final colour value written into the result buffer. They can be well used for composition of the rendered result into a background image or for a simple tonemapping of the result (limited because they do not have the information about the whole image at once).

The Renderman shaders structure together with a simplified information flow during rendering process is shown on image 2.18.

Because the shaders are extremely specialized only for the description of parts of the rendering process, the shader programming language is also designed in order to make the design and programming more intuitive and effective together with allowing more optimisations directly based on the processing flow. Its origins can be found in the popular C programming language, but the changes relating it to the computer graphics field are also quite extensive.

Almost the whole classical set of programming constructs as available in the C language than be used (e.g. *while*, *for*, *do*, *if-then-else*), but the graphics extension also bring new ones – like *illuminance* construct usable in surface shaders for looping through light sources (and allowing light source evaluation optimizations), or *solar* and *illuminate* used in light source shaders specifying the attenuation properties of the light emitted by the light source. For a complete list please refer to [48] and for more practically-aimed description to [4]. Some implementations may enhance the set of constructs to allow more advanced rendering techniques (e.g. raytracing), for their description in the 3Delight implementation please refer to [1].



Img. 2.18 - Simplified illustration of the structure and data flow of the Renderman shaders [48]

The variable types are also optimized for purposes of computer graphics algorithms. The standard includes usual *float* and *string* datatypes with the possibility to construct arrays and highly optimized *matrices*, but also extends the usual set by *color* datatype, with explicit and implicit colour space conversions, and a *point* / *vector* datatype in its usual unified 4D notation. The integer datatypes common in the usual programming languages are also present, but most of the implementations use a floating-point value instead of creating a special type for integer numbers. The variables provided as parameters for shaders can be further divided into two groups – *uniform* variables are set constant for an instance of a shader (e.g. surface shader assigned to an object), whilst *varying* can change for every surface parameters (e.g. texture coordinates assigned to vertices). This division is a form of high-level optimization, where the uniform variables are embedded into the compiled shader code in order to achieve a better computational performance (at the expense of used memory occupied by "copies" the shader code differing only in parameters).

Last basic property of a language, that needs to be described, is the set of operators and provided function library. The Renderman shading language in its basic form does not support any use of shared code in form of libraries, or object-oriented programming, but it provides a large set of graphics-related operators and functions natively, giving the programmer most of the graphics functions required in order to create any visual effect imaginable.

The list of all the supported functions would be very long, and most of the practical implementations of the interface are extending this list by even more functions, but to mention at least a few:

crossprod is the geometrical cross product operator,

dotprod returns the result of dot product of its parameters,

normalize returns the normalized value of a vector,

noise can be used for creation of parametric noise with different properties commonly used in parametric texture creation,

faceforward can be used to detect vectors facing the camera, often used in surface shaders to ensure the result is same for both sides of the surface

ambient, diffuse and *specular* functions summing the respective contributions of all lightsources applied to a surface without the need to loop through them

texture is a function for extraction (and interpolation) of texture information

The speciality of a shading language is its design narrowly aimed on the computer graphics field. For this reason such language do not need to include all means common in more general languages, as it may be on the expenses of the computational effectiveness or optimisation possibilities. The most important limits of the Renderman shading language are the inability to define new datatypes in any form (except fields), no direct or indirect recursion, no calls between different shaders (but passing of data is possible) and no code sharing between shaders.

The very short description given in this short section of this thesis is by no means complete, and had to leave many important aspects of the complex Renderman shading language unmentioned. Also the description was based only on the original standard, which has been improved many times in different implementations, although these improvements were never put back forming a new version of standard.

2.6 Representation and modelling of hair and fur

The realistic hair and fur is a very important feature required for realistic representation of human characters and animals, and is for this reason a topic of research from the beginning of realistic computer graphics. The field consists of many methods designed for different purposes, differencing in the means of level of physical accuracy and computational complexity. The more realistic methods are usually based on a complex physically-based simulation and therefore requires long rendering times, whilst the real-time methods, strictly constrained by processing time, usually represents hair properties in a phenomenological way not considering the physical base of their real basis.

Even though a large number of methods aim to simulate the physical properties of the hair at their most thinkable extent, a generally accepted solution still does not exist, which is also caused by the fact that global hair dynamics as well as exact lighting properties of hair are still unknown.

This thesis does not need a particularly complex model of the hair, and also no dynamic simulation is needed in order to render a still image. For this reason this short chapter only describes this field in a very short manner. For a more complete list of methods and approaches please refer to [53] and [35].

2.6.1 Hair representations

The first question in hair modelling is the representation of the hair, which consists of large number of individual strands, but also behave as a relatively coherent volume. From the rendering method point-of-view a suitable representation may allow more advanced acceleration techniques, and may also reduce the aliasing artifacts emerging from its high-detailed geometry. The same fine geometry also allows so use simplifications yielding from the fact, that from a larger distance it is not possible to distinguish each individual strand and therefore a statistically-based approximation of hair lighting properties may lead to the same perceptual experience even in the connection with very coarse geometry.

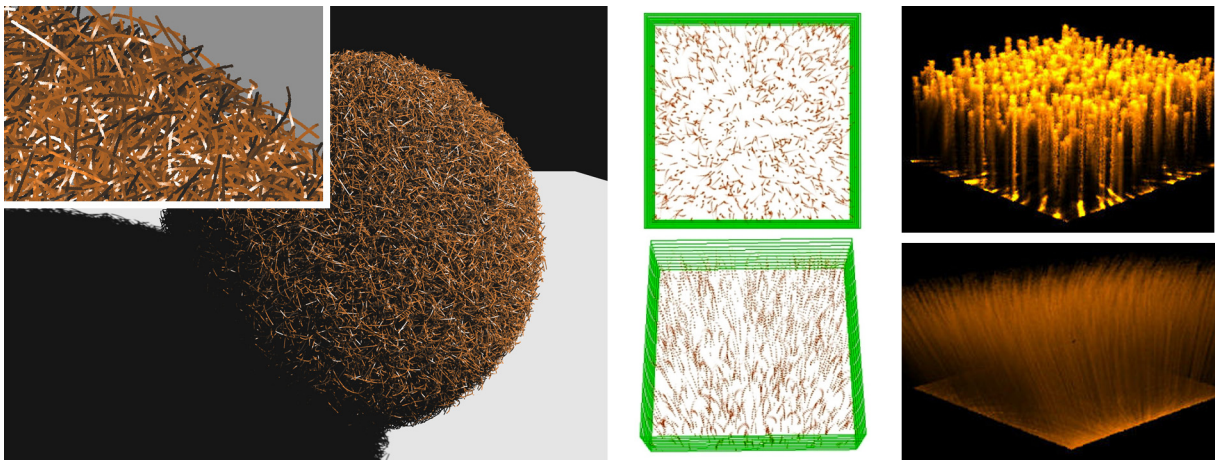
A method successfully exploiting this possibility is called Fake Fur [22]. The simplifications

yielded by this method are rather extreme – the approach does not use any form of geometry at all and represent all the visual properties of the hair by a statistical representation of reflectance function of a very short hair cover. The main limitation of this method is its limited application only to very short fine hair cover, and its inability to provide close-ups, but the extreme optimizations of the computational process led to its successful application in the film industry,

The second and more intuitive approach represents each strand of hair explicitly, leading to a high number of geometry primitives representing the whole volume. A naïve implementation of this method would use a curved cylinder with a general-purpose lighting model, as the real hair also has approximately circular cross-section, but this would lead to severe aliasing problems, especially with specular reflections. The solution is to use a flat curve representation, building on the fact that the detailed geometry of a hair strand is not visible even in a close-up, together with a lighting model accounting for all the light reflection and refraction along the cross-section in one computational step (e.g. approach used in [28]). The level-of-detail in this representation can be implemented via curve tesselation (algorithm is available for most of the cure representations) and by hair count multiplication in more close-up scenes [35].

Somewhere in the middle way between explicit and implicit representation is a shell-based method, which express the hair volume as a layered set of textures, with the high speed-up possibility yielded by modern graphic hardware. Indeed this approach was successfully implemented in real-time using a set of custom shaders processed on the graphics card [33], and is nowadays the most common approach to detailed fur rendering in real-time applications.

The last group introduced by the very first fur-rendering paper by Kajiya and Kay is implicit representation, storing the hair geometry in a volumetric texture (using a 3D texel in place of 2D pixel) together with all necessary data for rendering (e.g. hair tangent vector). Even though this method is not very effective in terms of required storage space, it allows a very practical way of reducing aliasing problems by using filter operators and mip-mapping.



Img. 2.19 - Illustration of different fur representations (left – flat-ribbon representation; middle – shell-based representation [33]; right – 3D texel representation, image courtesy of Michael Turitzin and Jared Jacobs, http://graphics.stanford.edu/courses/cs348b-competition/cs348b-03/fur)

2.6.2 Distribution of hair on a surface

The question of hair distribution on the underlying surface is addressed on two levels. The high-level

view relates to the hairstyle modelling, or to the distribution of different types and lengths of fur cover on an animal body, whilst the low-level view provides the distance relations between individual hair strands.

Holding this division the high level is described by an intensity map corresponding to the density of a particular region of the surface. In order to provide an intuitive method for describing such map, a few different approaches were developed, building either on a 2D mapping principles, direct drawing on a surface in 3D and wisps / clustering methods.

The first group of methods using the mentioned 2D mapping in order to create a density map are aimed to provide an interface between an usual 2D drawing program and the 3D representation of the shape. The problem is that there is no arbitrary way to perform such conversion, especially for more complex shapes, and therefore these methods can prove to be not intuitive. The most common method uses usual texture mapping built on a mapping of a set of 2D *(u, v)* coordinates to the corresponding vertices. Although it is the most common approach for texturing in computer graphics, it requires the manual specification of the unfolding. Another approach is using a usual mapping function as also known from texturing (e.g. spherical map), but its main limitation is the fact, that this form of mapping is only applicable to a specific class of object topology. The last method can be based on patches formed together to cover the surface, not far from an automatic uv-mapping system. As obvious, this method has the same limitations as the first mapping mentioned.

Second group of approaches specifying the fur cover density are based on a 3D drawing interface, allowing to “spray” the density map directly to the object. This method can prove to be very effective in terms of intuitiveness, and its only disadvantage is the necessity to rotate and move the viewport, which may employ a relatively complex user interface.

The last method is directly related to clustering approaches for hair shaping. Each cluster is placed on a particular spot on the surface and has its “guiding hair” - a 3D curve describing the shape of the whole cluster. The density can then be relative to the particular cluster and would correspond only to the hair strands inside it.

For more information about this topic please refer to [35].

The second group of methods, as described in the introductory paragraph, is aimed to more low-level hair distribution, directly relating the placement of hair to each other. As described in [35], the distribution of hair on the skin is very close to uniform Poisson-disc distribution, which is caused by the spatial requirements of the root cells allowing the hair growth. However this kind of distribution on a 3D mesh in computer graphics terms is a relatively complex problem, because for a complex geometry, where the solution cannot be determined using an analytical transformation of parametric space and a low-discrepancy quasirandom sequence generator, the solution requires a time-consuming iterative algorithm.

Research related to the Poisson-disc sampling of an arbitrary 3D shape has, apart from hair distribution, many different uses in computer graphics, e.g. remeshing of a complex polygonal representation, creation of a polygonal model from an implicit surface etc. The most targeted issue of the computation is its time-complexity, and for iterative algorithms the number of steps required in order to recover a good solution. An usual approach to this problem is based on some form of particle simulation, with particles movement restrained only to the surface of the processed shape.

The most intuitive would be a particle repulsion method [25], which first places the particles on

the surface in a random fashion and then iteratively applies a simple simulation based on a repulsion force between particles, until an approximation of an equilibrium state is achieved. The issue with this approach is, that each iteration requires evaluation of nearest-neighbour-set for each particle, which, even though well studied, still forms a computationally complex problem.

Another method would be based on starting from a random point on the object surface and adding new particles to the set by computing the best placement for the new particle from a subset representing the cover edges [55]. Another name for essentially same approach is a "packaging bubbles" algorithm, as described in [56]. The main issue with this approach is the fact, that a naïve implementation would lead to a completely even distribution, which is not exactly the point of this method. This problem can be solved by adding a level of randomness into the computation, changing the particle "forces" in a manner allowing a more randomly-looking final pattern.

2.6.3 Methods for creating global hair shape

A physically correct solution to the shape of the hair strands in hair volume is a complex problem, influenced by the extreme fine geometry of hair and closely connected to its dynamic properties, as obvious from a large set of possible hairstyles, even more stressed with long hair. The short hair, and especially short fur, is a simpler problem, because the strength of each short strand is enough to keep it from large bending, and therefore does not form such a complicated combinations. Because this thesis is aimed only on creation of static images, the whole area of dynamic simulations of hair behaviour is not part of this text, and for information about this topic please refer to [35] and [53].

The methods for static hairstyle modelling can be qualified into two large groups. The first group can be called "manual", because the methods in this group directly influence the shape of each strand, whilst "automatic" approaches employ some kind of helping structure influenced by user, which is later transformed into the actual volume of hair strands [53]. A complete list of methods in this field is very long, as the research in this field is very productive, and this text only explains some of the most basic principles. For a more complete list please refer to hair modelling and rendering surveys [35], [12] and [53].

The group of "manual" methods contains a few approaches, allowing different level of freedom in hairstyle modelling, which became common in today's graphic software. Parametric surface method defines the hair shape by using hair "strips" - flat guiding structures transformed in rendering process into a flat strip of hair [35]. Wisp models as well as generalized cylinders are aimed to reduce the number of manipulated shapes to hair groups with statistically-based generation of individual strands inside this helping volume [9], [35]. The definition of hair style is more complicated than in the previous case, but it also allows more power, even though the limit to hairstyles using some sort of strands is quite obvious. The last method of this group mentioned in this text is Multires [53]. Building on the base of generalized cylinders, this method offers a hierarchical approach, allowing to define dependencies between strands and giving more freedom to the user by the possibility to separate each cylinder recursively to as small groups as necessary.

"Automatic" methods differ from the previous group by the fact, that the modelling structure does not directly show the shape of the modelled hairstyle, but rather forms a virtual representation of it which needs to be later transformed in order to obtain the final hair volume with desired structure. The simplest approach from this group is based on the cantilever beam, which is a mathematical structure used in material strength field of physics described as a straight beam with support at one end

and with gravity as its main bending force. In order to receive a more complex hairstyle, extra forces (usually in form of a field) need to be applied. Another method in this group represents the hair shape as a fluid flow, creating the final shape by a set of user-defined obstacles, vortices and sources. The main disadvantage of this method is very "organized" solid look of final hairstyle. The last methods mentioned in this text are the styling vector and the motion field, which both defines the hair shape in terms of a field – a volumetric representation with orientation defined at every point of its body. Individual hair strands are then created by tracing the field lines of the vector field or by a simple particle simulation.

2.7 Hair and fur rendering

In the very same manner as the hair volume representations mentioned in the previous section, the rendering methods differ by the desired effects, constraints, and the research used as their base (either physically-based or simpler phenomenologically-based approaches). The main constraint of each rendering process is mainly the computational complexity, and for this reason every hair rendering method mentioned is trying to achieve the most realistic result under this strict constraint. This is also the reason why no generally accepted and usable solution exists so far.

Every hair rendering method is designed for a particular rendering approach (object / local / global rendering approaches) and a particular hair representation, and therefore it is most effective in this combination. This applies mainly to the shadowing and self-shadowing effects required by hair-volume rendering, because these methods are strongly connected to a particular rendering approach, however a weak connection of the same type can be also found in the lighting models, where each particular model performs the best with the proper scattering function it was designed for. All these relations are described in more detail below.

The simplest, fast and for some cases sufficient approach is called "fake fur" [22]. This method does not use any form of hair representation, as all the lighting, shadowing and inter-hair-scattering properties are described in terms of a parametrized reflectance function of a "mean hair" assigned to each region, which statistically represents the properties of all hairs in this region. Although this method is extremely computationally effective, it is strongly limited to short dense hair and has no possibility to render close-ups, because no fine geometry actually exists.

For a real-time constrained rendering, where an actual computation of the full shadowing would be too expensive, a statistical approach to the hair shadowing can be used, adding an additional dimension to the lighting function with the meaning of the distance from the hair tip. The assumption is, that statistically the amount of shadowing of a hair strand by other hairs can be determined from its tangent direction, the angle of incoming light and the tip-distance parameter, assuming that the hair strand is more likely to be in a shadow close to its root. The statistical base is close to a part of previous fake-fur approach, but this model also incorporates the fine geometry and is therefore capable of rendering realistic close-up views [33].

More sophisticated physically-based approaches involve usage of both lighting model and shadowing / self-shadowing algorithm, leading to a description very similar to real world. Next two subsections describe these two aspects separately for a small subset of methods relevant to the topic of this thesis. For a more complete list please refer to [35] and [53].

2.7.1 Hair lighting models

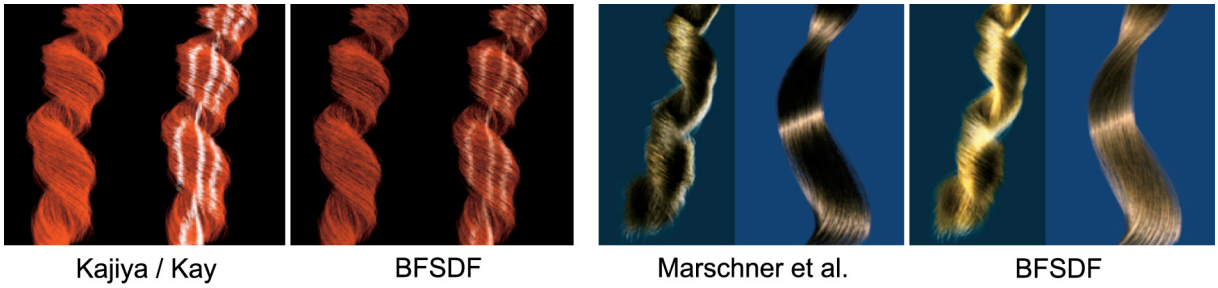
Although some of the usual lighting models used for hair are directly based on standard models used also for other phenomena, hair needs a special lighting model to mimic the effect of light scattering in the hair volume, which is extremely computationally expensive to simulate at its full extent. Very fine geometry also leads to severe aliasing issues, but the same fine geometry and cylindriv nature of hair allows to use a generalized lighting model accounting for all reflection, refraction and light scattering effects at once.

The first successfully applied model of hair lighting was proposed by Kajiya and Kay in 1989. Their approach was based on analysis of light reflection of a cylinder with Phong-type surface, not accounting for any internal cross-section light scattering. Even though this method does not consider also some other very important hair properties stressed in later work, it is still the most-used hair lighting model for its simplicity and realistic appearance based on diffuse term, which actually quite well interpolates the large set of different physical properties at once.

A new model, directly based on measured light-scattering function a hair strand, was introduced by Marschner et al. [36] and was also very successfully used in some of the new films (e.g. King-Kong). The theoretical base of their model is described by an analytic solution of physical simulation of light inside filament, whose properties are matched with the measured function. However their mathematical approach fails in a case of second internal reflection, which forms the most-important coloured “glinches” perceived in real hair, because the determined function has a singularity at this point leading to infinite light intensity in the very similar manner to caustics in analytical solution of the light refraction in a glass object. Their solution of this abnormality incorporates a specially-designed blurring algorithm, which changes the intensity distribution of the peak in a more realistic manner.

The most general description of the light scattering function of a filament, with the same base and terminology as BRDF function (bi-directional reflectance function) describing the surface light distribution of an opaque object is the BFSDF function (bi-directional fibre scattering distribution function), as introduced by Zinke et al. in [61]. Their description builds on another similar function called BSSRDF (bi-directional surface scattering reflection distribution function), which is capable of describing also general internal scattering in a material, and changes this complex function to a simplified version designed specially for filaments. Even though this description is the most general and add the possibility of statistical approximation of inter-hair scattering to the Marschner's model, its very general nature makes its parameters extremely hard to set properly in order to get the desired result.

Because the problem of hair lighting is very complex, all of the presented models provide just a different level of approximation of the real state, which itself is still a subject of active research.



Img. 2.20 - A comparison of different lighting models as presented in [61]

2.7.2 Shadowing and self-shadowing of hairs

Shadowing and self-shadowing is one of the most important properties of hair volume, because it creates visual effects severely changing the hair lighting models and also forms visual patterns crucial for distinguishing between different hairstyles [53]. The issue connected with it is the fact, that most of the hairs observed in real world are to some extent semi-transparent and that the shadows cast by very thin hair strands are very weak, leading to a strong requirement of proper shadow opacity handling during the rendering process (as demonstrated on image 2.21).

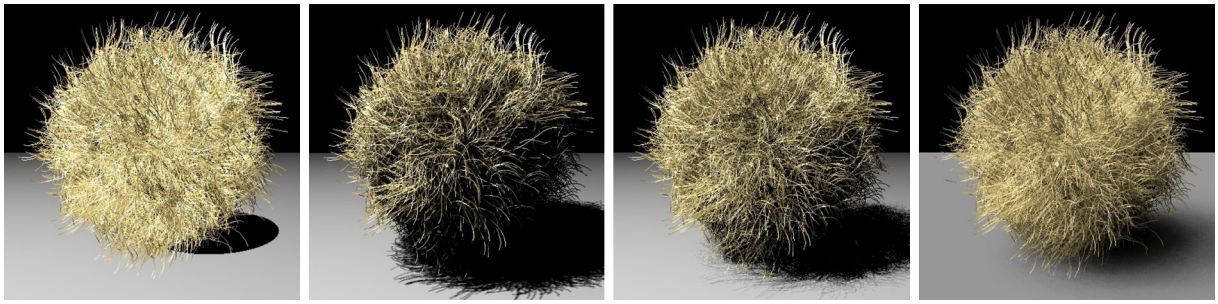
The first solution described in this text has its base in volumetric representation of the fur data, where texels, among other information, also store the density value, which can be gathered using a relative simple raytracing algorithm in the matter usual for rendering any sort of volumetric data [28]. The same method can be with a few changes used also for explicit data representation, where the coverage can be computed using a supersampling approach in order to avoid strong aliasing artifacts. Main disadvantage of this method is the fact, that the traced ray has to progress through the hair volume with very small step in order to capture all desired effects, and from every such step another shadowing ray has to be evaluated for every lightsource, which both together can make the computation very expensive.

The object-based rendering methods, represented by popular real-time implementations in hardware as well as in the Renderman system, are not capable to represent the global shadowing natively, because the very base of the method itself do not cover any sort of global relations between objects, except the occlusion. And the depth-buffer method used for global occlusion can be well used in order to achieve a shadow effect, by creating a depth map in a preprocessing step, related to the point of view of a light, storing the distances of the first object creating the shadow of this light in a particular direction. This information is actually representing a virtual shadow volume, determining the shadowed regions of the scene, which can be used in a simple manner in second rendering step to evaluate the shadows. The most obvious disadvantages of this approach are aliasing issues, caused by a limited resolution of the shadow map, and its inability to represent semi-transparent objects, because the shadow is (except of a smoothing function used to reduce aliasing effects) evaluated only in boolean true / false manner.

An extension of shadowmaps described above is aimed to allow semi-transparent and volumetric objects, defining each pixel of the shadowmap as a function connecting depth value with the shadowing opacity [34]. In order to represent this function effectively in a limited storage space, a compression algorithm was proposed, approximating this function as well as possible in the limited pixel structure. This method leads to more accurate shadows, especially with semi-transparent

volumetric objects as fur, even though the aliasing issue is still present.

The fur rendering with HDR lighting is a complex issue, because the high number of lightsources created by sampling of the HDR map, together with very complex hair geometry, lead a very computationally expensive problem. A solution aimed to furry objects is suggested by Yuksel et al. [57] [58], introducing an "igloo" data structure storing the information about rays traced through the fur volume in a very similar manner as the deep-shadow-map structure. The environment map is then used for the rendering directly, without the requirement of any sort of sampling. The main issue with this approach is its computational complexity, which is large due to mapping of all possible light directions, and its integration in existing rendering software, caused by its very different nature from most commonly used methods.



Img. 2.21 - The object-based fur shadowing methods as described in [58] (from left – no hair shadows, classical shadowmap, deep shadowmap, the "igloo" illumination method with an area light)

3 Critical analysis of the current state

Image-based material editing (IBME) is a promising new field contributing to the set of photo-editing methods mainly by a completely new point of view to the image data. Even though its aim focuses exclusively on changing the content of classical 2D images, the processing itself takes advantage of methods originally developed for description and simulation material properties of 3D objects. This novelty opens a large space for further research, because not long time ago such methods were considered as impossible as the mathematical proof of insolvability of shape-from-shading destroyed the dream of getting perfect a 3D shape data from a single image.

Possible uses of these methods are obvious. Photo editing software still forms one of the most broadly used applications of computer graphics, and the number of functions and plugins included in professional packages such as Adobe Photoshop demonstrates the need of more semi-automatic approaches to complex tasks, moving some of the painful "manual" work from the user to the computer. As the processing power of modern computers increases, more sophisticated methods can make their way to these packages, giving user the power to change the image in any way he wants.

Not long time ago the IBME methods included only changes to texture of objects pictured on an image. The reason is quite apparent – even though our visual system is capable of processing the texture information, the perceived conjunction of a region with is neighbourhood is quite limited, and the distracting effect of the applied texture also helped to hide the problems with shape extraction. Although the shape-extraction problems still remain, and the shape recovery techniques did not get any better recently, the connection of perceptual research allows to explore new possible applications beside the textural changes.

And this is where the revolutionary work of Khan et al. [29] came on stage. Their straightforward application of psychophysics results leading to a little "wild" combination of computer vision, perception and computer graphics, showed the possibility of more material modifications than ever before. Their approach is not fully automatic, because the number of parameters is quite high and a user intervention is still needed in order to receive a visually correct result, but this can also be regarded as a positive of the whole method, because it allows the user to influence the result the way he wants.

This thesis draws heavily from their work in the means of shape recovery, usage of HDR for both input data and lighting of the resulting geometry and the postprocessing of the extracted shape. But it is not sticking on a new implementation of already published methods, as it is aimed to prove that the similar approach can be used even for much more complex edits, introducing new geometry into the original image and compositing it in a way not distinguishable from the rest of the picture. Or at least this is the starting hypothesis. The first target for these methods is the addition of hair / fur into the image.

A possible problem connected with the very nature of hair cover can be the fact that it does not only cover the underlying surface, but also follows its structure, revealing the smoothness of the shape and its orientation. For this reason one of the main goals of this thesis is to evaluate the extent of this particular issue on the plausibility of the resulting images. The proper experimental evaluation of the results is however not included in this thesis and it is planned as a future study.

Previous chapter introduced methods and algorithms, which are relevant for achievement of this

goal to some extent. The reason was to introduce a broad range of approaches considered as applicable during the development period, even though only a fraction of them was finally implemented. This chapter focuses on the justification of every choice of a particular method by describing the requirements, desired properties and expected results necessary to achieve the goal of this thesis.

The main purpose of this thesis is the "proof of concept" of the proposed method. Therefore, the development process involved implementation and testing of more approaches than described in this section, because the main purpose was to find a usable combination providing visually plausible results, while another condition was to keep the final set of algorithms as simple as possible.

3.1 Shape from shading vs. psychophysically based approaches

The classical problem of computer vision, trying to recover the shape of an object from a single image using the shading associated with it, is a long-studied and well understood problem. As described in much more detail in chapter 2.2, the specification of the task is highly underconstrained by its nature, and the usual approach to find a unique solution is to introduce more constraints (e.g. probability constraint, one directional light with known direction etc.) or use some sort of additional information.

Classical algorithms usually perform well for certain types of images they were designed for and tested on, but a general solution, providing a good solution for a "large enough" group of input images, still does not exist [60]. And as it has been proven many times, it may not exist at all, leading to development of more perceptually-based algorithm promising at least a solution, which is correct in the visual point of view [21].

The perception of shape in images behaves the same way. Even for a very simplified case of a single light, our visual experience does not provide enough information, and the missing bits have to be filled in a later processing step involving a combination of memory and some sort of creative transformations. More perceptual details about this process can be found in [30]. The complex real-world lighting, usually very scattered and diffused by environment surrounding the examined object renders the task even more complicated and therefore completely unsolvable by analytical means. As in many other cases related to human perception, our visual system is capable of providing full experience even under these conditions.

The specific aim of this thesis is to change the object appearance. And as it suggests, the only thing changed in the scene is the surface of the object, leaving the background, lighting conditions and, most importantly, the viewpoint intact. Without the change of viewpoint, a strong perceptual shape masking is in the game, based on the fact that our visual system assumes that there is nothing wrong with the shape or the viewpoint [17]. Using this fact, the percepted object still appears consistent, even though its actual shape is not constructable, build from a set of not-connected patches or wrong in another way (see image 2.16). The inaccuracies would be very strongly perceivable if the shape was rotated, or the viewpoint was changed in any way, but nothing like this will happen in this thesis.

All the mentioned facts and assumptions are used in IBME shape recovery (e.g. [17], [29]), and are very successful with more complex objects (very simple shape would reveal the inaccuracies, because the analytic part of the visual system is then able to reconstruct the lighting fully), and with a simple filtering step (usually just a simple blurring) even for textured materials. A nice example of this fact can be seen on image 2.17)

The shape itself is not a result of the processing in IBME methods. The purpose is to rerender it

with different material, which usually contains texture of some form, or a definition of more complex light scattering function. This shows another possibility to use even more masking based on high-frequency content of the image (texture, highlights etc.), which can also hide a large set of shaping problems, as proved in [44] and [18].

All these facts connected together lead to a conclusion, that there is no point of using a very complex shape-from-shading algorithm, which would at the end only limit the set of possible applications to a small group of images it actually works on. The Textureshop approach leading into not-interconnected set of patches is also not usable in our case, because fur needs a constructable geometry as its base in order to model its shaping, covering of different hair clusters (appearing on more complex geometry), and hair (self)shadowing, which is also an extremely important feature of the hair and fur. For this reason, the approach of Khan et al. [29] (with small adjustments) was used, exploiting the perceptual evidence at its full extent, leading to a constructable shape, which still appears correct when composited back into the original image, even though has just a little correspondence to the original object.

The main issue was, that it was not sure at the beginning of the development of this work if the fur with its flow consistent with underlying shape would give perceptually enough masking, and if the (self)shadowing necessary in order to render realistic fur would not expose the inaccuracies in the shape too much.

3.2 Fur and hair modelling

Many different approaches to hair modelling and its distribution on object surface were introduced in the chapter 2.6 and the question of choosing one of them is discussed in this short section of this work.

Because the aim of this thesis is just a proof of concept, there is no need to use a complex styling method, long hair with its complicated simulation, or any sort of complex hairstyles generation. On the other hand, some control of the fur shaping and its distribution have to be allowed, even though it is not even close to complexity of any of the advanced methods introduced in section 2.6.3 .

Therefore, the choice of simulation method used as base for hair shaping led to a very simple particle simulation, which is closely related to cantilever beam known from material physics, with definition of a simple styling vector field allowing to change the global direction of the hair “growth”. In order to increase the correspondence to real fur both physically and visually, a simple randomising algorithm is necessary to enhance the furry appearance.

The parameters of such algorithms, however, have to be set carefully, because not enough randomness leads to a “fluidy” look and reveal the underlying shape too much, whilst too much of randomness leads either to too much “scruffiness”, or with fine fur to large problems with aliasing (otherwise masked by the high amount of hair with the same direction) leading to a random-pattern look.

For exact mathematical and algorithmical description of the final algorithm please refer to section 4.3.2 .

Another important part of the fur generation is the hair density control on the surface of the underlying shape. Because the viewpoint of the rendered result is exactly the same as the viewpoint of the source image, a simple 2D projection of a map with grey level corresponding with the fur density

in a particular region can be used. This map cannot be the same as the mask used for isolation of a shape from the rest of the image, because it is used for shape recovery and its change (e.g. holes meant as parts without the fur cover) would lead to artifacts in the final image.

A distribution of hair on the surface is another question that needs to be answered. As concluded in most of the work on hair [35], the distribution of hair on a surface is very close to a Poisson-disk-driven distribution function. However, evaluation of such function involves either a solution of a global set of equations describing every hair in terms of its position, or an iterative algorithm solving this set approximately. Even though many methods were developed for this purpose (see section 2.6.2 for more details), the extremely high number of hairs on a furry surface leads to very high requirements of computational power.

By using the same conclusion, that the number of hair on a surface is very large, it is possible to use a random distribution, which actually for the limit case of infinite number of samples leads to the Poisson-disk distribution as well. The random distribution was also used in the final form of this thesis (see section 4.3.1).

Last possible simplification of the hair distribution is based on the fact, that most of the fur on the surface, which is leaning away from camera, is covered by the fur closer to the viewpoint. Again, building on the fact that the viewpoint remains the same from the first shape recovery until the last rendering step, the density of hair coverage of a region can be modified by using the *cos* of camera view vector and the normal vector of the surface point, leading to significant reduction of hair strands rendered into the image whilst the visual experience remains the same. For practical purposes a dotproduct of the view vector and normal vector of the surface is used, leading by its mathematical definition to the same results.

3.3 Rendering of fur and hair

The issue of hair rendering was discussed many times in both previous work and many chapters of this thesis. The correct answer for every particular case lies in a compromise between physical accuracy and computational complexity of each method. Because the hair is necessary for creation of almost any virtual character, the support of its creation is added to all practically used rendering systems and there is no need to reimplement an existing method or to try to reinvent the wheel by developing a new one for purposes of this thesis.

To some extent the Renderman renderer supports most of the methods described in the chapter 2.7 by the implementation of a set of shaders performing the computation, but the most intuitive way and the simplest approach natively supported by Renderman is to represent the hair strands by flat ribbons facing the camera. The ribbons are described as curves with automatic subdivision, based on the details necessary for rendering, and its colour is determined by a hair lighting model, which is also simplifying the cylindrical hair shape to a flat representation [35].

One of the most important features of hair is its shadowing and self-shadowing, providing the actual experience of the shape of underlying object. First approaches with 3D textures used a relatively complex light scattering simulation in order to obtain the most accurate results of light scattering in the hair volume, whilst later approaches used raytracing-based methods together with flat curve representation of hair strands.

The most optimal approaches, providing good results with just a fraction of the computational complexity of the raytracing methods, are based on shadow maps [3] (for more details see section

2.7.2). To represent the volumetric nature of the hair in a more suitable way, the deep shadow maps approach introduced a depth-density function, which can well approximate the light scattering as traced through the volume [34]. The original Renderman implementation included shadow maps, applicable using a relatively simple manner in light shaders code, and as the 3Delight implementation enhanced the supported set of features by deep shadow maps [1], they also became the option used in this thesis.

New methods of hair shadowing can provide even better results than deep shadow maps, with even less computational complexity, as described in [57] and [58], but the heavy optimizations of Renderman system allow to achieve results very close to those methods in terms of both computational time and resulting effects. Another problem connected to the new methods is also the fact that their different view on the scene data makes them hard to implement in existing rendering frameworks.

To solve the issue of complex lighting of the scene based on a real-world photograph as its source, usage of the HDR information seems to be the best option. The unavailability of more input data except the original 2D image leaves us with the only option of Khan et al. [29], which uses an approximation of the light probe by mapping the original image onto a half-sphere. For more information about the final implementation of the lighting please refer to section 4.5.1 .

Last question that needs to be answered is the choice of a hair lighting model, as described in section 2.7.1 . The models differ mainly in the extent of approximations used in computation, and their base in analytical solution of light scattering or in the light distribution data obtained by observation or direct measurement [53] (more details in sections 2.7.1 and 4.4).

The first practically used model of light properties of a hair strand was proposed by Kajiya and Kay [26]. Their analytical solution of the Phong lighting model applied on a cone, with integration along its radius in order to avoid the aliasing in specular part of the model, has proved to be very simple and relatively accurate, and is still the most common used model for hair lighting. For this reason it was the first choice of this thesis. Its main disadvantage is that it is not considering the cross-section scattering in the hair strand, and therefore it is not incapable to properly represent the light hair with all its highlights, and has also problems with more complex lighting conditions.

This problem was partly solved by Marschner et al. [36] by using the data collected in direct measurement of the hair scattering function as the base for analytical model of almost all the effects of hair structure on light distribution. For this reason it was the second model implemented in this thesis. The problem of this model is its inability to account for inter-hair scattering as the measuring method used to obtain the source data was not able to include them.

Last possible method, which could be used for the hair lighting is the filaments light scattering model [61]. Even though this model is the most general and should be capable to capture all effects of light interaction with hair volume, the number of parameters, which would be required to be set properly in order to obtain a usable model for a particular hair colour is very high, and therefore impractical.

4 Description of the work

Previous section has drawn the conclusions about the feasibility of a particular set of techniques to the solution of the proposed problem. This section extends these thoughts by more detailed description of every one of them with enough details to understand their implementation. It is not aimed to provide a commentary for the actual code, but rather to give a more concise explanation of each method in means of mathematics, algorithmic description and its results.

Because the concept of the whole thesis is a “proof of concept”, even this chapter will continue in this manner. The main difference from a development project is the missing system design part, because as a research project the final design was not known at the beginning of the work and the whole final software is not meant to be a final version of a usable application. For this reason, the implementation does not include a usable user-friendly interface of any sort, and the final design does not have the ambition of a properly developed piece of software, even though all the object oriented paradigms are used and held.

The implementation consists of three programs sharing parts of their source code. The first part is an OpenGL and wxWidgets tool for testing of the properties of the depth-extraction algorithm. It is not meant to produce any results directly presentable as the output of the system, but on the other hand provides a tool for the evaluation of the proposed depth-extraction chain. Two other programs are command-line-based applications without any graphic interface and produce a set of RIB (Renderman Interface Bytestream) files describing the scene and another set of TIFF images serving as inputs for the rendering process. The rendering itself is realized in a Renderman-compliant renderer with deep-shadow-maps support (tested and implemented for with 3Delight). The first of the two remaining programs is an reimplementation of the Khan's et al. [29] IBME technique changing the object appearance into glass, and the second one performs the actual algorithm described in this thesis.

All three pieces utilize the same configuration file describing the parameters for all the algorithms, even though the actually used subset of the values is different for every one of them. The processing and usage of these variables is described further in the text with the explanation of the method they are involved in, and a complete list of the parameter names with their description can be found in appendix 7.2 .

4.1 Input information and file formats

As it was already mentioned many times in the previous chapters, this thesis is a proof-of-concept of the fur rendering hypothesis. For this reason also input files of the programs are not meant to be user friendly, but rather to provide the largest possible flexibility to test the used algorithms and methods.

There are only two compulsory input files for all the algorithms. The first and the most important input file is the source HDR image in the Radiance .hdr format, and the second one file is a greyscale map used for separating the object of attention from its background. The object-isolation step could have been realized using an automatic algorithm with a simple GUI, but this was not the main goal of this thesis and therefore it is not implemented. The remaining set of input files contains only optional data that are not necessary for the basic algorithm itself, but they can provide more control over the process of generation of the files serving as inputs of the rendering process. These files are used for modulation of the fur parameters, namely its length, colour, density and possibly

providing alternative LDR background for the final composition of the result. A more detailed description is provided below.

4.1.1 Algorithm inputs and temporary files

This short chapter explains the meaning and format of all the used input files and the temporary files generated by the command-line implementation of the proposed algorithm. The "temporary" in this context means that these files are used as inputs for the rendering process itself and therefore do not represent a complete result of the whole processing.

The set of the input files for the fur program is provided in the following list (the sets of input files for the depth-extraction preview and the glass-material program are subsets of the described index):

HDR input file – provides the input information for the depth extraction, and therefore guides the whole process. If no alternative LDR background is provided, a simple global tonemapping operator is used on this input file to produce a background for the resulting rendered file. The supported file formats are 32-bit per channel version of HDR TIFF and the Radiance HDR format.

Object isolation mask – is a grayscale map used to separate the object of interest from its background. The accuracy of this map is essential in order to receive a good approximation of the resulting shape and this way directly influences the quality of the resulting image. This file can be provided in 8-bit grayscale TIFF or PGM file format.

Fur density mask – is another grayscale map used to modulate the density of fur in a particular area by projecting the map to the 3D shape determined in the depth-estimation step of computation. The values in the map are evaluated as a fraction of the maximum density value set in the configuration file, where white corresponds to 100% of the maximum density and black to no fur cover. The supported formats are again 8-bit grayscale TIFF and PGM file format.

Fur length map – is behaving the same way as the density map, modulating the length of hair in different areas of the recovered shape. The possible file formats are the same as in previous case.

Alternative LDR background – can be provided in order to bypass the simple tonemapping algorithm of the implemented system, giving the possibility to use an external tonemapper with better results than the simple exponential mapping used internally by this implementation. The only supported format for this file is 8-bit per channel RGB version of the TIFF format.

Fur colour modulation image – can be used to modulate the diffuse colour / internal fur attenuation term (corresponding to the lighting model used). The expected file format is the same as in case of alternative LDR background.

Configuration file – is the last possible input file, which is usually used to provide all the information necessary in order to get the processing results. Even though it is still possible to specify all the parameters directly on the command line, the high number of them would render this approach very hard to handle.

An example of the most of the described files (except the configuration file due to its extent) can be

found in the appendix Error: Reference source not found together with the rendered result.

The temporary files, as mentioned above, are actually results of most of the processing described in this text, and their only purpose is to serve as the input for the rendering of the final image using Renderman system. These files are:

Depth map – described as an HDR (32-bit) TIFF grayscale map, serving as the displacement texture for the rendering process. The HDR representation is necessary in order to aviod quantisation errors, because the usual 256 levels when representing a shape of an object have proven to be not accurate enough, leading to "staircase" look (see section 4.2).

LDR background image – represented as 8-bit RGB TIFF image server as the background texture for composition of the rendering result. It can be either provided externally, or generated using a simple tonemapping algorithm included in the implementation of HDR to LDR conversion.

Global RIB file – is a small auxiliary file only meant to connect the two main processing steps into one by including the scene RIB file and shadowmap RIB file.

Shadowmap RIB file – is used as the description of the preprocessing step generating the deep-shadow map for every lightsource in the scene. It sets up the rendering details (antialiasing, turns off the pixel smoothing and sets one sample per pixel as required by the deep-shadow-map algorithm) and renders one "frame" of the map for every lightsource by setting the camera settings to the viewpoint of the processed lightsource. The result of this step is a set of .dsm files describing the light shadow volumes of all lights in the scene.

Scene description RIB file – sets up the rendering settings for the final image (antialiasing level, pixel smoothing filter, number of samples per pixel), the projection, adds light sources together with (already processed) shadowmaps, initialize the imager compositing shader and includes the world RIB file to allow the rendering of the resulting image.

World RIB – is the last of the temporary files required for rendering. It contains the description of the underlying polygon together with the definition of its displacement map and connected shader, creates the instance of the fur-lighting-model shader and ad last the description of every hair as a Bezier curve primitive. Because the total number of hair strands in this file can be very large (up to 500 000), the binary format is used to describe their parameters in order to save disk space.

The actual content of the generated RIB files is not described in this thesis, as it is just a set of statements not much different from an usual scripting language. For exact description of every command found in these files please refer to [48] and [1].

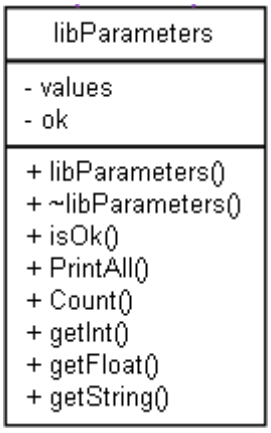
4.1.2 Configuration file

The configuration file is common for all three programs presented in this thesis and its only purpose is to substitute the large set of possible command-line arguments with a more intuitive form. The necessity of a simple configuration file became obvious in one particular moment of the development process, when the number of parameters grew rapidly with the implementation of the fur lighting models. The configuration file is in a very simple line-based format with one parameter allowed per one line. The parsing process also allows to use comments in order to give the possibility to make the

file more readable.

Even though most of the modern programs would employ some form of XML-based configuration file in this place, for the particular purposes of a research project the possibility to supply all the parameters in a configuration file as well as using command-line arguments was a stronger requirement, leaving intact the possibility to use a script to change input parameters without the need to generate multiple configuration files.

The class structure holding the information about parameters is shown on image 4.1. The constructor of the class takes arguments *argc* and *argv* in the same format as the usual entry function of a C program, determines from them if the input should be read from a configuration file or if the parameters are provided on the command line itself, and creates an associative array connecting the parameter name with its value. It does not try to test if all the necessary parameters are supplied and if they are in the correct format in this stage. Second step of the processing is the extraction of the parameter values from this structure by multiple calls of g*et** functions, providing the name of expected parameter, default value for optional parameters and its type (by the version of the function called). The testing of correct types and existence of all required parameters is tested in this stage, and can be tested by calling the *isOK* function after the attempt of extraction of a parameter or simply at the end of extracting all of them.



Img. 4.1 - The structure of the configuration file object

The complete list of possible parameters of the algorithms is relatively extensive, and therefore can be found in the appendix 7.2 .

4.1.3 Information representation in HDR file formats

HDR file formats are specifically designed to effectively represent the characteristic large luminance interval of high dynamic range images. As it was already mentioned in chapter 2.1.2 , the direct floating-point representation would be the most intuitive, but the problems with compression of such representation make it not very suitable for this purpose. For this reason a set of different methods was developed, varying in the basic thought behind possible compression of data, the amount of information lost by the very principle the representation method and also the compression ratio.

This thesis is not trying to describe all the commonly used formats, because due to the ongoing research in this field none of them was generally accepted as a standard representation. For a general description of some of them please refer to section 2.1.2 of this thesis and for a more complete list to

the HDR imaging book [45].

The following two subsections describe the two HDR formats supported by this work in details necessary in order to understand the way of data processing involved, and also provide references with the actual implementation details. These two formats are the original Radiance HDR format, today usually described by the file extension ".hdr" and mostly considered as a standard (original) representation, and the TIFF format, whose flexibility to represent almost any kind of image data together with an open-source library for loading and saving makes it one of the most common options today.

The actual implementation of the image structures is based on a relative complex inheritance graph, and its complete description would be very extensive. The relatively self-explaining class diagram of these objects is presented in appendix 7.3 .

4.1.4 Radiance file format

The storage format of the Radiance file description is based on separating the common exponent of the three channels (RGB or XYZ, in the further text only RGB is mentioned, but the same principles are used for the XYZ version), leading to RGBE representation as shown on image 2.3. The base of this simplification is the fact, that in most real-world images the value range of each of the RGB channels for a particular pixel is very close, allowing to separate the common exponent value with just a small loss of data. The practical implementation employs an 8-bit mantissa for every channel and 8-bit exponent value, leading to 32-bit data per pixel, luminance range 10^{-38} to 10^{38} and final representation-based error less than 1% on this range [5]. The actual value-conversion algorithm can be described using following equations (equation 4.1 for conversion from floating-point RGB values to RGBE representation and equation 4.2 for the backward conversion):

$$E_n = \left\lfloor \log_2\left(max(R_o, G_o, B_o)\right)\right\rfloor + 128$$

$$R_n = \left\lfloor \frac{255\ R_o}{2^{(E_n - 128)}} \right\rfloor \quad G_n = \left\lfloor \frac{255\ G_o}{2^{(E_n - 128)}} \right\rfloor \quad B_n = \left\lfloor \frac{255\ B_o}{2^{(E_n - 128)}} \right\rfloor \tag{4.1}$$

$$R_n = \frac{R_o + 0.5}{256}\ 2^{(E_o - 128)} \quad G_n = \frac{G_o + 0.5}{256}\ 2^{(E_o - 128)} \quad B_n = \frac{B_o + 0.5}{256}\ 2^{(E_o - 128)} \tag{4.2}$$

where R_n, G_n, B_n and E_n describe the result parameters (integer values in 4.1 and floating-point numbers in 4.2), while R_o, G_o, B_o and E_o are the input values for the computation (of floating-point type in 4.1 and integer type in 4.2). The original conversion was described only algorithmically [5], and the presented mathematical explanation can be found in [45].

This particular representation allows another simple extension by using well-known run-length-encoding (RLE) compression algorithm. Because spatially close pixels usually determine the same object with very close luminance values, there is a relatively high possibility of same component values computed by previous set of equations (or at least for exponent value). To make the encoding simpler, for every scanline all values of pixels are reordered into a component-wise sequence, leading to *n* bytes of red channel followed by *n* bytes of green channel etc., with *n* as number of pixels in a row [5]. Efficiency of this encoding is about *1:1.3*, which makes the resulting image approximately the same size as an uncompressed 24-bit image [45].

The file itself consists of two parts – an ASCII header and the binary-encoded RLE compressed scanline data. This text does not provide all the details necessary for the implementation of a loading function, because such description would involve unnecessary amount of technical data that can be found elsewhere as well. For the exact description please refer to [5] and [51].

4.1.5 Support for HDR data in the TIFF file format

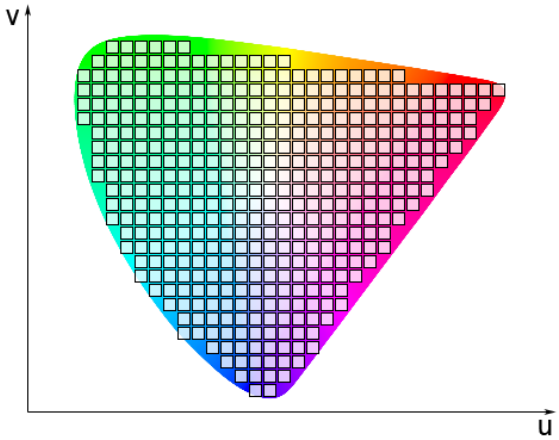
The second common approach for storing the HDR information, as described in more detail in section 2.1.2 , is to separate the luminance information using a suitable colour model (e.g. Yxy, Luv) and then compress only the luminance information, which contains all the HDR information of the image [45]. This is also the method used in the TIFF format, the second HDR file format supported by this work.

Colour version of HDR TIFF file description comes in two variants – 24-bit and 32-bit, both built on the same basis of the Luv colour model, differing only in the number of bits per each channel and the colour information representation. The 24-bit representation offers 10 bits for luminance information (L_{10}) and 14 bits for colour information (C_{14}) stored as colour index to the colour table covering the colour space in a very effective way (see image 4.2), while the 32-bit format stores 16-bit signed luminance value (L_{15}) and separated 8-bit *u* and *v* channels. An illustration of the bitwise layout of data stored for every pixel can be found on image 2.4, while the colour-quantisation space is shown on image 4.2. The non-linear perceptually-based compression of the luminance channel is described in following set of equations [45]:

$$L_{10}=\lfloor 64\,(\log_2 L+12)\rfloor \qquad L_{15}=\lfloor 256\,(\log_2 L+64)\rfloor \tag{4.3}$$

$$L=2^{\frac{(L_{10}+0.5)}{64}-12} \qquad L=2^{\frac{(L_{10}+0.5)}{256}-64} \tag{4.4}$$

where L represents original (or reconstructed) luminance value and L_{10} with L_{15} represent the luminance channel of 24-bit and 32-bit format respectively.



Img. 4.2 - The quantisation of the uv colour space in 24-bit version of TIFF image format [45]

The description of the LDR versions of the TIFF format would use the same rules as it is common in all other classical image file formats, and therefore it is not included in this text. The grayscale HDR version of the tiff image (used in this thesis as the temporary file describing the depth information as input for the rendering process, as described in section 4.1.1) uses only the 16-bit luminance value

compressed the same way as above, just not including the colour information.

For the programming realisation of the loading of this relatively complex file format it is not necessary to reimplement the whole reading function, because an open-source C library is available for this purpose [50], and it was also used in the development process of this work.

4.2 Extraction of the 2.5D base shape

The extraction of the 2.5D base shape is a crucial part of this work, providing a base for the rest of the processing as well as for the rendering process itself. For this reason a lots of space in the previous chapters (namely chapters 2.2 , 3.1 and their subchapters) was devoted to give enough detail about the state of the art in this particular field and this chapter only focuses on the actual mathematical explanation of the chosen approach.

The chosen approach draws heavily from the psychophysics research on the shape perception, because the shape-from-shading techniques have proven to be insufficient for this purpose (see chapter 3.1 for more details), and is similar to the method used in the image-based material editing paper by Khan et al. [29]. Main differences can be found in a new implementation of the bilateral filter for the texture extraction, and in more focus on the smoothness of the resulting shape, as required for the fur cover consistency. For this reason the bilateral filter application in this work uses much larger intensity kernel size than reccomended in previous work, and the resulting shape therefore looses some of its details, which would be masked by the fur cover anyway. The inaccuracies of the shape detection are, as in the previous IBME work, masked by the high-frequency nature of the fur cover and the complex lighting extracted from the HDR image used as the light source.

4.2.1 HDR highlight reduction

The first computational step connected with the depth extraction is the reduction of highlights in the input image. The possibility of this step is related to the fact, that in an HDR image no information is lost during the acquisition steps, and to cropping of the high luminance values occurs. For this reason it is possible to suppress (or possibly enhance) values above a certain detected threshold using an exponential mapping in order to reduce (or increase) the effect of highlights on the surface of the object. The importance of this step is given by the fact, that the sigmoidal compression and shaping function introduced later are not capable to reduce the effect of strong luminance spikes at the places of highlights.

The highlight compression / enhancement process can be mathematically explained using the following set of equations:

$$L' = \begin{cases} L_{min} + (L_{max} - L_{min}) \left(\alpha \dfrac{L - L_{min}}{L_{max} - L_{min}} \right)^{\beta} & L > L_{min} \\ L & L \leq L_{min} \end{cases} \tag{4.5}$$

$$L' = \begin{cases} L_{hmax} + (L_{min} - L_{hmax}) \left(\dfrac{L - L_{hmax}}{L_{max} - L_{hmax}} \right)^{\beta} & L > L_{hmax} \\ L & L \leq L_{hmax} \end{cases} \tag{4.6}$$

where equation 4.5 describes the highlight enhancement process and equation 4.6 shows the

compression process, with L_{max} as the maximum luminance value in the input image, L_{min} as the minimum luminance and L_{hmax} describing the starting luminance value of the highlight, which can be either user-specified or computed from the derivation of the image histogram (as illustrated on the left part of the image 4.3). Parameters α and β adjusts the amount of the compression / reduction, with $\alpha = 0.05$ and $\beta = 20$ as the default values for the highlight enhancement equation 4.5 and $\beta = 0.05$ for the compression equation 4.6. The actual effects of the described processes on the resulting images are illustrated on image 4.3.

Img. 4.3 - An illustration of the highlight reduction / enhancement algorithm[29] (left – histogram of an HDR image and its derivative, with the detected start of highlight, right – images demonstrating the luminance remapping, from left – original image, enhanced specular component, enhanced diffuse component)

4.2.2 Sigmoidal compression and gradient reshaping

The perceptual experiments serving as the very base of the used method of depth extraction define the relation between luminance and perceived depth as "dark-means-deep" paradigm [32]. Even though this definition might be accurate enough for the conclusion of a perceptual report, in order to be able to use it in an algorithm, a mathematical description of it has to be formed.

The first step is to relate "depth" value to the perception of luminance in the image. As it was shown before in many different fields of perception research, the human visual system mostly provide the experience of a particular property in the logarithmic domain (e.g. see chapter 2.1.1), as well as the recommendations of the authors of bilateral filter (as described in next section) also conclude, that their filter is best to be applied after some sort of logarithmic compression of the input data. The logarithm, however, is an unlimited function, which may cause problems in the computer vision way of data processing. A function with very close properties to logarithm, but limited to interval *0..1* is called *sigmoid function*. Its origins lie in statistics, but nowadays it is often used in models of early human vision, and its description in the language of mathematics is [29]:

$$L_s(x,y) = \frac{L^n(x,y)}{L^n(x,y) + \sigma^n} \tag{4.7}$$

$$\sigma = \left(\prod_{x,y} L(x,y) \right)^{\frac{1}{N}} = \exp\left(\frac{1}{N} \sum_{x,y} \log\left(L(x,y) \right) \right) \tag{4.8}$$

with L_s describing the compressed luminance value at the pixel position (x, y), L as the input luminance value on the same position, σ as the semi-saturation constant computed in a preprocessing

step, N as the total number of pixels in the image and n as an compression-steering constant, with default value *0.73* common in the modelling of the instance photoreceptor responses [29].

Due to sigmoidal compression applied before the bilateral filtering, the range of transformed values d is strictly limited with the gradients near silhouette too small. For this reason an inversion of the sigmoidal compression is applied, resulting in a more suitable values range for gradient extraction. The actual equation is:

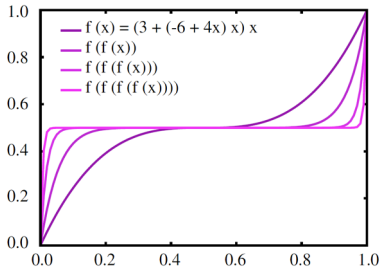
$$d'(x,y)=\left(-\frac{\sigma^n\, d(x,y)}{d(x,y)-1}\right)^{\frac{1}{n}} \tag{4.9}$$

with d' as the transformed value of the pixel depth, d as input depth value and the rest of the parameters with the same meaning as in the equations mentioned above.

The last function of the reshaping set of methods is an analytically-determined function, which is an estimation of the relation between depth and luminance information. In the program itself is this function referred as the "magic" function, because it has no direct relation to any measurable physical unit. In the mathematical point of view it is a simple polynom [29]:

$$f(x)=(3+(-6+4\mathrm{x})x)\,x \tag{4.10}$$

with x as the input value. The amount of shaping of the final result can be easily adjusted by its recursive application, as shown on image 4.4.



Img. 4.4 - The graph showing the "magic" gradient-shaping function [29]

4.2.3 Fast bilateral filter implementation

One of the main disadvantages of the first IBME algorithms was the fast, that they could be applied only to a non-textured surface, because the texture would cause perturbations of the recovered shape and therefore lead to artifacts in the rendered result, as can be seen on the processing in the Textureshop software [17]. Pinpointing this issue Zelinka et al. [59] used a gausian-mixture-based model to statistically determine the texture of the processed shape and this way eliminate its influence on the result. Another approach was introduced in the Khan's et al. paper [29], where a special filter was used to remove the texture from the surface. This approach is also the method followed in this work.

The desired properties of such filter would need to include two main constraints – the filter has to be able to remove the texture, which is usually determined by smaller local differences of the luminance of the object surface, and has to keep the strong edges, usually describing the real object

edges, intact [29]. The computer-vision description of such filter would be based on anisotropic diffuse [42], where the small variances are propagated through the spatial dimensions of the image, while strong edges form barriers for this "intensity flow". Anisotropic diffuse is, however, an iterative process, and thus not very suitable for the image processing due to its computational requirements.

A solution of this issue was provided by the bilateral filter [49], which is essentially an approximate solution to the anisotropic diffuse with a particular parameters after a large number of integration steps. The approximation is caused by the fact, that the anisotropic diffuse would stop the "flow" even on a thin edge, whilst the bilateral filter due its base in low-frequency spatial filtering would not. Even though it can be seen as a drawback, because it cannot be mathematically linked with the diffuse, its final properties proven to be very usable for many applications (e.g. tonemapping [42], edge-aware smoothing [42] etc.) and often even better than its mathematical origins in the diffusion function.

Img. 4.5 - Result of the bilateral filter processing showing the edge-aware blurring effect (for both pairs of the images – left shows the original image and right is the processing result)

The bilateral filter consists of a combination of two Gaussian functions – 2D spatial one and another 1D in the intensity domain, giving the filter its particular properties. The basic equations showing its relation to both anisotropic diffusion and the Gaussian-blur follows [42]:

$$L_r = \frac{1}{k(s)} \sum_{p \in \Omega} f(p-s)\, g(L_p - L_s)\, L_p \tag{4.11}$$

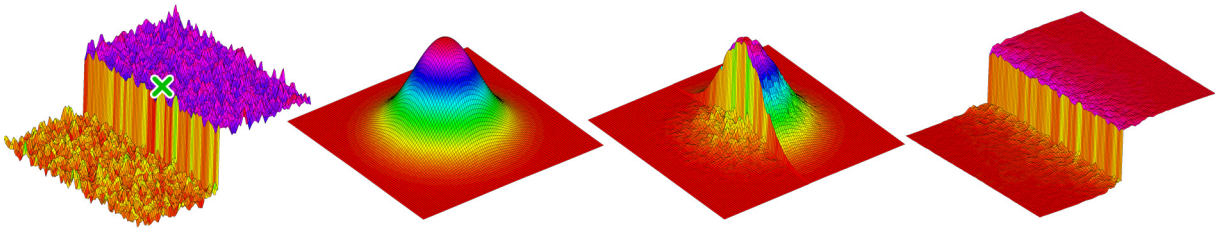
$$k(s) = \sum_{p \in \Omega} f(p-s)\, g(L_p - L_s) \tag{4.12}$$

where L_r is the resulting luminance, $k(s)$ is the normalisation term, p describes the position of the processed pixel, s is the position of a neighbouring pixel from the neighbourhood set Ω, L_p and L_s are their respective luminances, term f describes the spatial Gaussian function and finally term g is used for intensity function.

The description itself consists of two terms. Equation 4.11 is the main description of the filter, whilst the equation 4.12 forms a normalization term with different value for every pixel, based on the neighbouring luminances in the same manner as the main term. From the description it could be obvious, that the result is a non-linear filter, with nonlinearities present in two places – first in the normalisation term and second in the multiplication of two different Gaussian functions with different parameters. For this reason the only optimisation obvious from this description is the simultaneous computation of two maps with main and normalisation values with division performed at the end of

the computation. This is, however, not enough, because the complexity of this term is the same as pure convolution, i.e. $O(N^4)$, leading to extreme computational requirements especially for large images. Please note, that due to the nonlinearity caused by the multiplication of two Gaussian functions inside the processing, this filter cannot be implemented using usual FFT speedup approach used for most of the confolution-based filters in computer vision field.

The effects of such processing on a grayscale image are illustrated on image 4.5. The blurring effects are obvious in the regions with lower contrast, whilst strong edges remain unchanged. The processing of one pixel close to an intensity edge can be described in the means of image IMAGE. Again, it is obvious, that the amount of noise in the flat regions is significantly reduced, while the edge with strong contrast is kept.



Img. 4.6 - Demonstration of bilateral filter processing on one pixel close to an intensity edge (from left – original image shown as 3D graph with z as the intensity value, "x" denotes the processed pixel; the spatial kernel size illustration; the kernel modified by the intensity function showing the high weight value for the surrounding of the processed pixel with the same intensity; smoothed result of the processing of all pixels in the original image)

Based on the low-frequency filtering properties of the filter, a different acceleration technique was proposed, making the updated computation practically usable. The simplification builds on the fact, that for the low-frequency filters an undersampled input together with smaller Gaussian kernel can be well used, leading to approximately same results as the full-scale one. The first implementation of this technique exploited this property in the intensity domain, and in order to make the computation even faster, the "layers" built by quantisation of the intensity range were separately processed using the FFT-based convolution. This led to speed improvement by order of approximately two magnitudes (depending on the roughness of the quantisation). For more details about this technique please refer to [42].

An even more optimized approach, reaching the real-time processing constraints, was recently introduced by Chen et al. [11]. Their method is based on the undersampling of the input as well, but the implementation includes undersampling in both intensity and spatial domain.

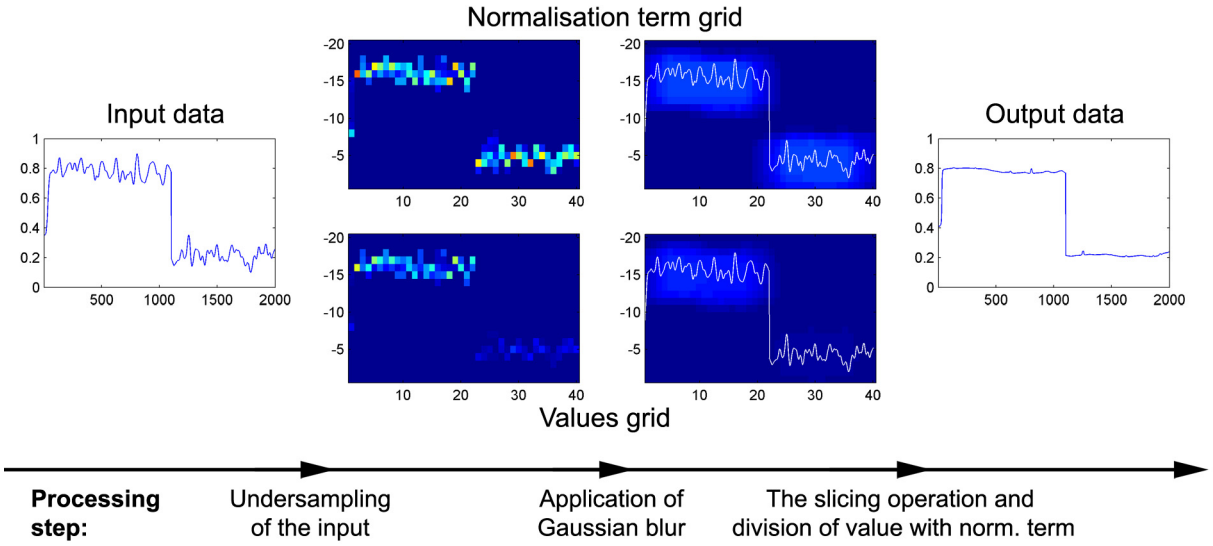
In the first preprocessing step the input data are quantized and stored in a 3D "bilateral grid" structure, which has two dimensions the same as the source image and the third one is defined by the intensity value in the particular pixel. Every cell of the new structure includes two values – first represents the accumulated "normalization" term from the equation 4.12 and second represents the accumulated "main" term as described in equation 4.11. The accumulation in this step is formed by summing values of the undersampled versions of the input intensities (for "normalization" term) and their multiplications with pixel intensities (for the "main" term).

In the next step the convolution with a 3D Gaussian kernel is applied separately on both terms,

leading to a smoothed grid structure, which contains the undersampled data now fully describing the sums from the equations 4.11 and 4.12, and the last remaining operation is the data extraction together with the normalisation from equation 4.11.

This step is called “slicing” by the authors and involves extraction of the resulting data from the coarse structure by determining each value on the coordinates of the original high-resolution data and their interpolation using trilinear filtering.

The whole processing algorithm is illustrated using simplified case of 1D input data and 2D bilateral grid structure on image 4.7. Because a full explanation of this algorithm in a more understandable way would occupy too much space in this text, for more details about the basis of this method together with a more extensive description please refer to [11].



Img. 4.7 - Illustration of the bilateral filter evaluation using bilateral grid, as introduced in [11]

The problem with this implementation is the fact, that the undersampling and later applied linear interpolation can lead to visible artifacts. Even though this problem is not apparent in the usual applications of the bilateral filter, for the extraction of the shape used in this thesis this problem becomes much more stressed (see image 4.8). Luckily the fur cover masking properties are strong enough to hide even this underlying-geometry issue.

4.2.4 Base shape as a shader

Previous sections of the text introduced all necessary steps required in order to recover an approximate depth map from the source image. This section focuses on the transfer of this depth map into the rendering process via an HDR luminance map and a displacement shader guided by this information.

A naïve approach for the transfer of the depth information to the renderer would use a polygon representation, which has to be recovered for the generation of the hair distribution anyway (as described in the next section). The main problem with this method is the fact that without a complex geometry-simplifying algorithm, such interpretation would need two polygons per every pixel of the source image in order to guarantee the coherence of the geometry, leading to unnecessary large description file containing at least six floating-point values per every pixel.

The Renderman interface provides a different mechanism to achieve the same result. The

displacement shader is a piece of code changing the object geometry by moving points on the surface of an object. By using the texture to guide the displacement of a plain polygon facing the camera and underlying the whole rendering window, together with orthogonal projection and an imager shader providing the necessary integration with the background image (as described in section 4.5.2), which should overwrite the segments of the polygon which were not displaced, it is possible to achieve the same final result as using a geometrical representation of the shape.

The only remaining issue is the quantisation applied when storing data in form of an usual grayscale image. The minimal step used to covert the floating point data into a 256-level grayscale image is small enough to provide a perceptually smooth result, but in this case it leads to a "staircase" appearance and incorrect gradients of the surface (e.g. image 4.8). The solution would be to use an 16-bit floating-point luminance map, which would allow more accuracy for every pixel. Such a file format is only provided by a non-standard extension to the TIFF format, not supported by most of the image editing software, but luckily implemented in the recent version of the libTIFF library and in the 3Delight implementation of the Renderman interface.

Img. 4.8 - A visualisation of the depth-estimation algorithm results as displayed using OpenGL (left – the extracted shape from the same viewpoint as the source image, middle – a rotated version of the same shape showing only vague correspondence to the original object, right – the effect of the quantisation of the result with non-quantized version on the left side and quantized on the right side)

4.3 Generation of hair description

The last step of the input processing, performed just before the rendering itself, is to generate the hair cover for the recovered surface. This process is divided into three steps – in the first step the hair bases are distributed in a random manner on the underlying surface (section 4.3.1), the second step performs a simple Newtonian particle simulation in order to create the shape of each hair (section 4.3.2) and in the last step is this information stored in a Renderman RIB file as a Bezier curve "flat ribbon" primitive with rounded root, thinning towards the tip and having additional transparency, creating an impression of a thick fur cover (see section 4.3.3).

Because the number of hair covering the shape has to be very high (for tested cases even up to 500 000 separated hair strands), it is necessary to use an optimized representation as well as a simplified generation process. The main performance issues in this step are the stress on generating the lowest number of hair with keeping the impression of full covered shape, and aliasing problems

connected with the final rendering process of such fine geometry. Both issues are also addressed in the next chapters.

4.3.1 Hair distribution on the underlying surface

The hair distribution algorithm is based on the statements from section 3.2 with conclusion, that a random-based algorithm would be the most suitable for the case of the high number of strands.

The generation method uses the depth map information in a polygonal format, created by connecting each four neighbouring depth samples into two triangles, and the normal information computed by crossproduct of two edges of each polygon. To ensure a more even distribution of the samples over the whole surface, a low-discrepancy quasirandom generator based on radical inverse function is used [24], with base 2 for the first dimension and base 3 for the second one. The number of samples D generated for a particular polygon is determined by the density parameter D_i, modified by the optional density map value d_m in range $0..1$, the polygon area a_p, a random number r in range $0..1$ and the dotproduct between camera view vector v and polygon normal n to account for the surface direction in relation to camera:

$$D = \left| (a_p \; D_i + r) \; d_m \; (\vec{v} \cdot \vec{n}) \right| \tag{4.13}$$

The position of each sample inside the polygon area is determined using a set of equations:

$$\xi_1 = \begin{cases} r_1 & r_1 + r_2 \leq 1 \\ 1 - r_1 & r_1 + r_2 > 1 \end{cases} \tag{4.14}$$

$$\xi_2 = \begin{cases} r_2 & r_1 + r_2 \leq 1 \\ 1 - r_2 & r_1 + r_2 > 1 \end{cases} \tag{4.15}$$

$$\vec{h} = \xi_1 \vec{p}_0 + \xi_2 \vec{p}_1 + (1 - \xi_1 - \xi_2) \vec{p}_3 \tag{4.16}$$

The equation 4.16 forms the well-known barycentric-coordinate formula for determination of a point h in a triangle described as three point p_1, p_2 and p_3, based on two parameters ξ_1 and ξ_2. The sum of these parameters has to be lower than 1, therefore a simple transformation of two random variables r_1 and r_2 (from interval $0..1$) needs to be performed in order to fulfil this condition, as described in equations 4.15 and 4.14. An illustration of the final hair distribution, with very low density value in order to allow each strand to be visible, is demonstrated on image 4.9.

4.3.2 Hair shape and particle simulation

Section 2.6 provides relative extensive description of many modern methods used in the field of hair simulation and shape modelling, but for this work it is not necessary to use such complex means. The very simple method finally used is related to the cantilever beam representation and the vector-field shaping method. The shape of the cantilever beam can be well approximated by using a Newtonian particle simulation model, starting from the point of surface with initial velocity direction determined by the surface normal, and influenced by a “gravity” vector describing the flow of the final fur, as shown in the set of equations:

$$\vec{a} = \frac{\vec{g}}{m} \tag{4.17}$$

$$\vec{a} = \frac{d\,\vec{v}}{dt}\ ,\quad \vec{v}_0 = \vec{n} + \vec{r} \tag{4.18}$$

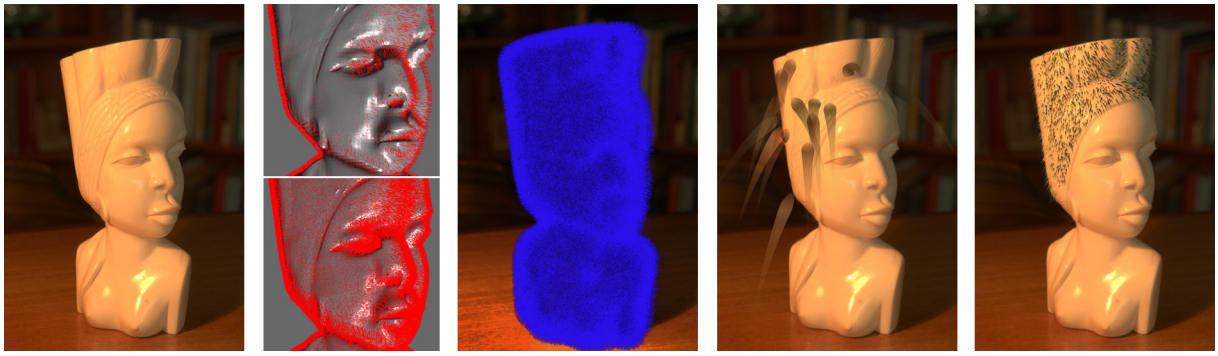
$$\vec{v} = \frac{d\,\vec{p}}{dt}\ ,\quad \vec{p}_0 = \vec{s} \tag{4.19}$$

where a describes the vector-field influence to the shape by using a user-specified vector g and a biasing constant m, v forms the velocity of the particle initialized by the normal vector n of the underlying polygon and a random vector r providing the shape with randomness as observed on real fur, and finally the term p forms the position of the particle creating the final strand shape, initialized by the position vector s of the hair root on the surface. The resulting shape in a large scale is shown on image 4.9.

4.3.3 Hair rendered as curve primitive

The most-effective way to represent fur in the Renderman interface is to use curve primitive, making use of extensive optimizations included in the 3Delight implementation. Curves are rendered as flat ribbons always facing the camera, with the possibility to use interpolated direction information for their colouring in the assigned surface shader.

Real animal fur has a number of distinct properties, which were to some extent also considered in the final representation. First, the fur strand is thicker at the root and is thinning towards the tip, having a conical shape, and second, as the hair gets thinner towards the tip, the coloured internal part providing the fur its colour gets thinner too, allowing less internal scattering and therefore more transparent look. Both of these properties were implemented, as illustrated on image 4.9.



Img. 4.9 - The fur cover representation in more detail (from left – original image; fur distribution and direction vectors with (bottom) and without (top) the randomisation of direction; fur cover rendered without a lighting model; details of strand representation and the shape simulation; a low-density fur cover stressing the fur distribution)

4.4 Hair lighting models

The lighting properties of hair and fur are still not well known, for a volume of short reflective and refractive fibres, which a hair or fur cover really is, expresses extremely complex light interaction behaviour. Even a single strand of hair, as viewed under the microscope, has a very complicated structure formed by thin overlapping cells on surface (called cortex) and a completely different

internal structure (called medulla) providing the hair with its colour.

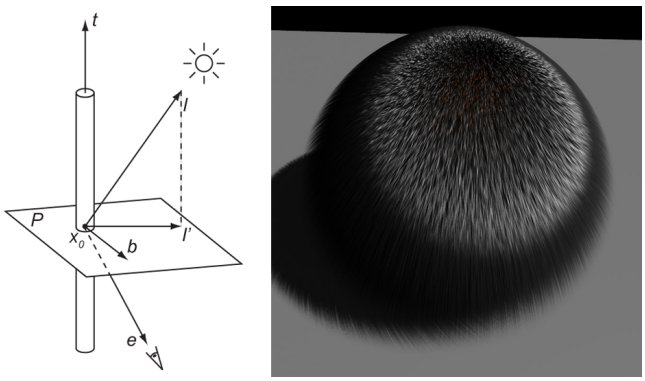
This thesis uses two implementations of hair lighting models. The first model is the original Kajiya-Kay model [28], which is based on observation and mathematically described as a generalization of light scattering on a cylinder. Even though this model was the very first and oldest, it is still the most commonly used for its simplicity together with very realistic appearance. Apart from its simplicity, its main advantage is also a very intuitive set of parameters, closely related to Phong's lighting model.

The second implemented model builds on physical measurement of the hair scattering function, and in order to achieve the same results provides an analytic solution of light scattering inside hair cross-section [36]. Although physically very realistic, the high number of parameters not intuitively related to any lighting properties visible on the final rendering, make it quite impractical.

As described above and also quite extensively in previous chapters, both models have their advantages and disadvantages, and therefore are well supplementing each other when one of them fails to produce a visually plausible result.

4.4.1 Generalization of Phong's lighting model to filaments

The very first practically-usable hair lighting model, as presented by Kajiya and Kay on Siggraph conference in the year 1989 [28], is based on analysis of the raytracing of light interacting with a solid cylinder and was strongly inspired by the Phong's lighting model. A simplified description of thoughts behind the model follows, drawing the notation from image 4.10.



Img. 4.10 - The notation used in the Kajiya-Kay lighting model description (left) [28], a sphere covered by fur rendered using this model (right)

One linear segment of a hair strand in this model is represented by its position x_0 and its tangent vector t. The light vector l points from the light segment to the light source and vector l' is a projection of this vector to a plane P perpendicular to the tangent vector. The vector b is perpendicular to both l and l', and the final vector e represents the view vector. The analytic solution first determines the scattering on the plane P by integrating the Phong's diffuse and specular component around the surface of the cylinder, and then relates this to the angle between the original vectors l and e and their respective projections to the plane P. The actual process of the derivation of these equations is relatively extensive, for more details please refer to [28]. Their description of the of the lighting process led to following set of equations:

$$\Psi_{diffuse} = K_d \sin(\vec{t}, \vec{l}) \tag{4.20}$$

$$\Psi_{specular} = K_s \left(\vec{t} \cdot \vec{l}\ \ \vec{t} \cdot \vec{e} + \sin(\vec{t}, \vec{l}) \sin(\vec{t}, \vec{l})\right)^{P_s} \tag{4.21}$$

where $\Psi_{diffuse}$ and $\Psi_{specular}$ denote the reflected amount of light as related to a corresponding element of the Phong's lighting model, K_d and K_s represent the reflection coefficients determining the strength of each component, P_s is the specular exponent again with the same meaning as in the Phong's model and the rest of the notation can be determined from image 4.10.

Even though this model is already usable, Banks [7] provided a simplification of the described terms, which is also the form used in this work. His description included additional vector *h* computed as a half-way vector between the eye and light direction, and an exponent P_d allowing to "sharpen" the diffuse reflection. The equation in a form of sum of both diffuse and specular terms Ψ, which corresponds to its usual practical usage, follows [33]:

$$\psi = K_d \left(1 - (\vec{t} \cdot \vec{l})^2\right)^{P_d/2} + K_s \left(1 - (\vec{t} \cdot \vec{h})\right)^{P_s/2} \tag{4.22}$$

4.4.2 Cross-section scattering function

In an attempt to represent the scattering inside hair in a more accurate way, Marschner et al. measured the light-distribution function of hair and created an analytical model with corresponding properties. The model is based on an analytic solution of the reflection on the hair surface as well as the light travelling through the hair, and provides solution for three different reflection cases R, TT and TRT (see image IMAGE for the explanation). As obvious from this explanation, the model does not contain any form of integration along the hair cross-section, but rather solves the attenuation term for every particular angle of incoming light and view vector directly.

Although the solution is very close to the measured physical properties of hair, and the model itself is built on physical simulation of light, the very complex and unintuitive parametrisation makes it rather impractical. Another issue is connected to the measurement method, where one strip of hairs was measured in a special device. For this reason data did not contain the inter-hair scattering effects so important for light hair colours, and therefore the model is usable only for dark hair (as objected in [61]).

Because the full description of the model in both its basis and its mathematical representation is extremely complex, this text only introduces the most important equations and sketches illustrating the thoughts behind, and then continues to show the results of the model. For full explanation please refer to [36].

The very base of the scattering function can be described by using a set of equations:

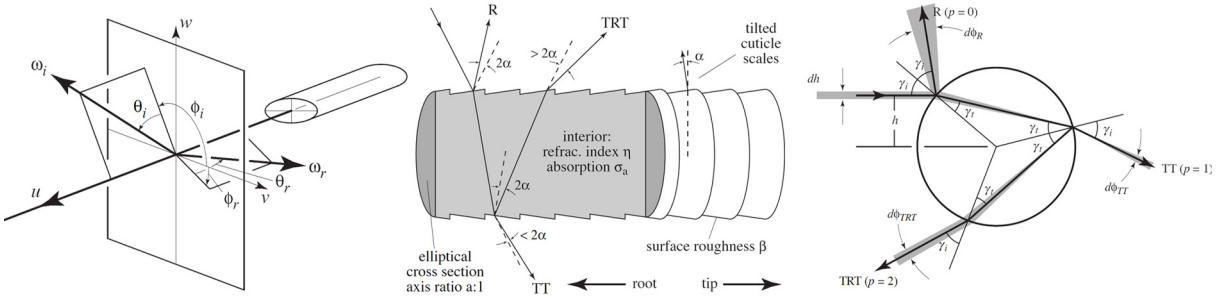
$$S(\phi_i, \theta_i, \phi_r, \theta_r) = M(\theta_h)\ N(\eta'(\eta, \theta_d), \phi_i, \phi_r) / \cos^2 \theta_d \tag{4.23}$$

$$\theta_d = (\theta_r - \theta_i) / 2 \tag{4.24}$$

$$\eta'(\eta, \theta_d) = \sqrt{\eta_2 - \sin^2 \theta_d} / \cos \theta_d \tag{4.25}$$

$$\theta_h = (\theta_i + \theta_r) / 2 \tag{4.26}$$

where S determines the final scattering function, M is the longitudinal scattering term, N represents azimuthal scattering, η is the hair index of refraction (usually *1.55*), η' is the index of refraction corrected by the θ_d value in order to separate the cross-section scattering function N from the longitudinal function M completely, θ_d represents the difference angle between the view and light vector, θ_h is the half-way longitudinal angle, and the rest of the notation can be determined from image 4.11. The general form of the S function in equation 4.23 actually differs for every component, and for this reason the further equations are written in three versions (one for each component).



Img. 4.11 - The notation used in the Marschner's et al. model [36] (left – the notation of the incoming and outgoing vectors and angles, middle – the side-view of a hair with components description, right – the notation used for cross-section scattering computation)

The first complex term to deal with is the longitudinal part of the S function:

$$M_R(\theta_h) = g(\beta_R, \theta_h - \alpha_R) \tag{4.27}$$

$$M_{TT}(\theta_h) = g(\beta_{TT}, \theta_h - \alpha_{TT}) \qquad where\ \alpha_{TT} = -\alpha_r/2\ and\ \beta_{TT} = \beta_r/2 \tag{4.28}$$

$$M_{TRT}(\theta_h) = g(\beta_{TRT}, \theta_h - \alpha_{TRT}) \qquad where\ \alpha_{TRT} = -3\alpha_r/2\ and\ \beta_{TT} = 2\beta_r \tag{4.29}$$

$$g(\sigma, x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{4.30}$$

where β determines the longitudinal width (in interval *5°* to *10°*) and α determines the longitudinal shift (in interval *-10°* to *-5°*). As it is obvious from the equations above, the longitudinal function corresponds to a Gaussian-shaped function formed around the ideal reflection / refraction angle.

The last term to describe is the azimuthal scattering part, which is a little bit more complicated:

$$N_p(p, \phi) = \sum_r A(p, h(p, r, \phi)) \left| 2\frac{d\phi}{dh}(p, h(p, r, \phi)) \right|^{-1} \tag{4.31}$$

where p is the number of internal path segments (*0* for R component, *1* for TT and *2* for TRT), r represents the number of solutions of the h function (*1* for R, TT and up to *3* for TRT, which solves the exit parameter h for a ray with Φ (used as both value and a derivation of its description function) as the difference angle between incoming and outgoing vector (projected onto normal plane of the hair) and A as the attenuation of light as being reflected, refracted and absorbed in the hair body. The exact

description of the h function and its solution is beyond the scope of this work, for its explanation please refer to [36].

The solution of this function for the TRT component produces two singularities with infinite intensities, which is mathematically and even physically correct, because they correspond to glinches observed in real hair. However the problem with mathematical equations is, that they do not represent the real inaccuracies from the real hair, which smooth the resulting TRT values into a more realistic interval, in the same manner as caustics of glass would have infinite intensity if behaving exactly according the mathematical equations. For this reason Marschner et al. introduced a smoothing algorithm, which is supposed to mimic the effect of small inaccuracies on the hair surface. However the description of this algorithm is again beyond the scope of this thesis an in can be found in [36].

The next last term to describe is the attenuation function A. It again consists of multiple versions determined with parameter p (as explained above):

$$A(0,h) = F(\eta', \eta'', \gamma_i) \tag{4.32}$$

$$A(p,h) = \left(1 - F(\eta', \eta'', \gamma_i)\right)\ F\left(\frac{1}{\eta'}, \frac{1}{\eta''}, \gamma_t\right)^{p-1}\ T(\sigma_a', h)^p \tag{4.33}$$

$$\eta'(\theta_t) = \frac{\sqrt{\eta^2 - \sin^2\theta_t}}{\cos\theta_t} \tag{4.34}$$

$$\eta''(\theta_t) = \frac{\eta^2 \cos\theta_t}{\sqrt{\eta^2 - \sin^2\theta_t}} \tag{4.35}$$

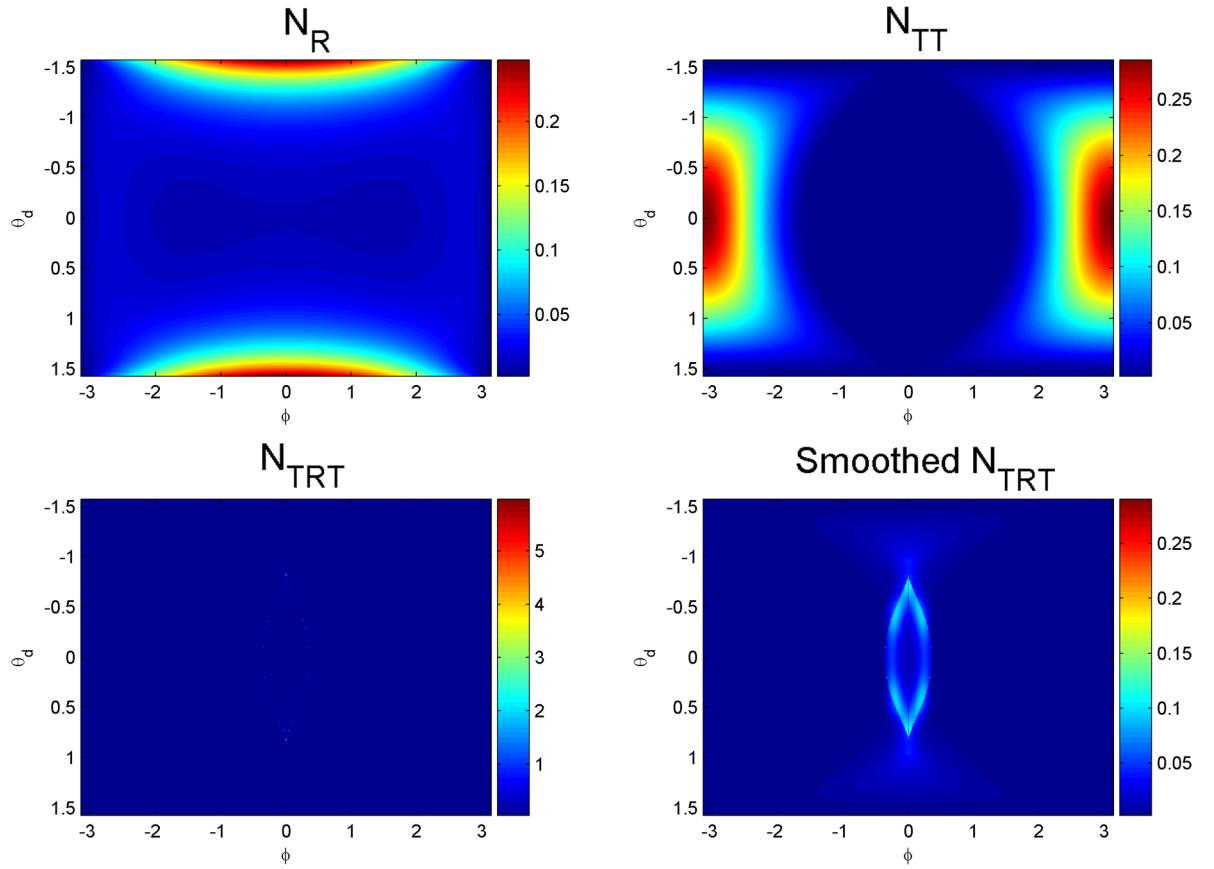
$$\sigma_a'(\theta_t) = \frac{\sigma_a}{\cos\theta_t} \tag{4.36}$$

$$T(\sigma_a', h) = \exp\left(-2\sigma_a'\left(1 + \cos(2\gamma_t)\right)\right) \tag{4.37}$$

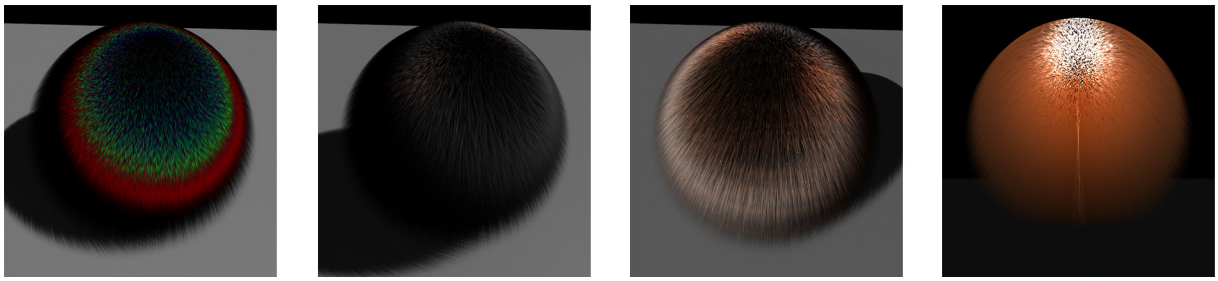
where η' and η'' are indices of refraction accounting for the projection of the input vector to the normal plane of the hair, γ_i and γ_t are just different descriptions of the input and output angle (see image 4.11), F are the physical Fresnel's functions describing the amount of reflected and refracted light on an edge of two different environments, T is the attenuation in one pass through the hair with its length determined by previously mentioned parameter h (transformed to its γ_t alternative), σ_a is the absorption coefficient responsible for hair colour (different for red, green and blue channel, in interval *0.2* ... ∞) and σ_a' is its value fixed with the inclination θ_t.

As it is now obvious from the large set of equations above, which are only a simplified representation of the mathematics behind this model, it is very complex indeed. The whole implementation can be well described in terms of graphs, as shown on image 4.12. The actual full function is four-dimensional, so the graphs represent only slices of the result, but they can nicely represent the most important properties of the model.

The N_R component shows the strongest reflection with Φ close to zero, meaning that the light is reflected in a mirror-like fashion, while N_{TT} has the opposite behaviour, having its highest values near $\Phi = \pm\pi$ suggesting its representation of light passing through the hair. The most interesting “glinches” component N_{TRT} takes effect only is a very limited angular interval, as it is also expected to behave.



Img. 4.12 - A set of graphs representing 2D slices of each component of the cross-section scattering lighting model. Note the smoothing effect on TRT component in both terms of graph shape and values scale.



Img. 4.13 - Results of the cross-section scattering model (from left – longitudinal components of the model with N_R in green, N_{TT} in red and N_{TRT} in blue; the fur sphere lit from side; the same sphere lit from front; and the same sphere lit from up behind exposing mainly the TT component)

4.5 Rendering of the resulting image

The rendering process itself consists of two main steps, which are in the implementation separated into two RIB files (Renderman interface bytestream, the description of the rendering parameters and of the scene for the Renderman system).

The DSM files are generated in the first step, describing the shadow volume of each lightsource in the means of deep shadow map method, as explained in more details in section 2.7.2 . The file-

based approach to shadow generation is one of the specialities of Renderman system, because the rendering method is object-based and not global, as usual in most modern renderers. This allows a very simple possibility of paralelization of the rendering process without any need to synchronize different computing machines on a lower level. Each shadow volume is in the source file represented as a new “frame”, the only difference from the standard frame rendering is in the “display” plugin which determines a “rendering target” - the format and information stored as a result of rendering process.

The second step is the rendering itself, which makes use of the surface shaders implementing the lighting models of the hair, lighting shaders providing the sampled representation of the HDR map used for lighting the scene, and the combination of displacement and imager shader used to generate underlying surface and composite the rendered result back into the original background.

Most of the mentioned steps were described in the previous sections, this section is aimed to fill the two missing bits – the light source generation via sampling of the HDR map and the composition of the rendered result into the original background.

4.5.1 Light sources and lighting

The lightsource types supported by the program are two – the first possibility is to use one user-specified directional light which, although originally implemented only for testing purposes, proved to be useful for some images, where the fur lighting model (especially the Marschner's et al. version) performed poorly together with the HDR environment mapping. The second possibility is the lighting based on half-spherical mapping, as described in [29]. Because the first type is very simple, this section focuses on the description of the second case.

The half-sphere mapping can be used as an approximation of the lighting probe in cases, where this information is not available. IBME methods can be such case, because the only information about the surrounding environment available is the source image. The construction of this approximation consists of two steps – in the first step a circular part the original image is cut, aligned with the original image and with maximal possible radius, and in second step is this cut mapped onto spherical surface by linear projection.

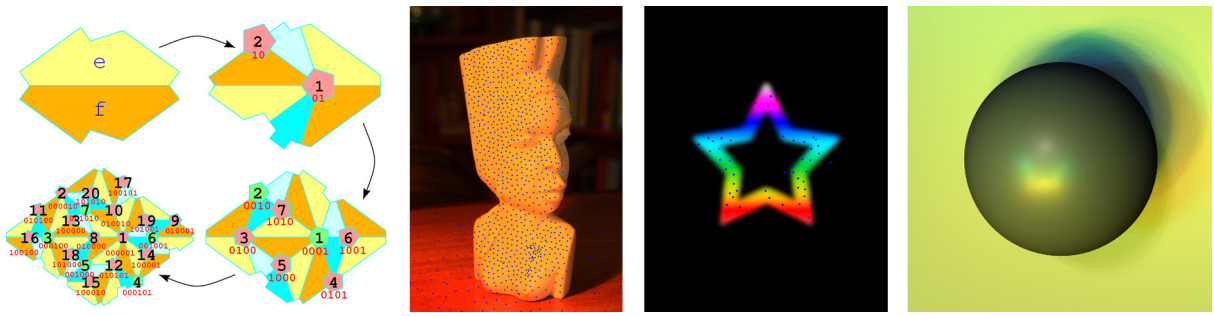
Mathematically the coordinate system *(x, y)* of the original image is remapped into a normalized coordinate system (x_n, y_n) using the information about the image centre (x_c, y_c) and the maximal possible radius R as half of the shorter image edge length (equation 4.38). The normalized coordinate system can then be used directly for the projection (represented in equation 4.39 as vector d):

$$(x_n, y_n) = \left(\frac{x - x_c}{R}, \frac{y - y_c}{R} \right) \tag{4.38}$$

$$\vec{d} = \left(x_n, y_n, 1 - \sqrt{(x_n)^2 + (y_n)^2} \right) \tag{4.39}$$

The direct usage of an HDR image as a light source is possible only in global illumination rendering methods, for all other cases the environment map has to be transformed into a set of directional lightsources. In order to represent the map well, these lights have to be placed in such parts of the image, which contain the strongest luminances and thus are the most important in the lighting process. The process of determination of such parts is called importance sampling.

With the fast progress of the HDR field a relatively large number of importance-sampling methods for images was developed, usually providing the exchange between the computational complexity of the search and the quality of result. One recent method based on a recursive geometrical structure has performed exceptionally well in both terms, changing the usual sampling interval from tens of seconds to real-time [40].



Img. 4.14 - The Penrose tiling, importance sampling based on it and its usage for HDR lighting (from left – the recursive division in the Penrose tiling [40], an HDR image sampled using this technique, an HDR star image also sampled with this technique and the rendering result of a sphere lit by the half-sphere method using the sampled star image.

The mentioned sampling algorithm is based on Penrose tiling, a geometrical structure with subdivision properties very close to a fractal in terms of recursive division. The pattern generated by the tiling of the image based on the importance of each segment is then processed using a simple relaxation process, also based on the relations determined by the tiling structure, and the resulting distribution then exhibits properties of the uncorrelated "blue noise".
The implementation of this sampling is available of the author's website as a general-use library, which was also used in the implementation of the process in this work.

4.5.2 Composition of the result

The very last step of the rendering process is the integration of the rendered result into the original image. Although the image composition using the alpha channel is well known process, the particular application in this thesis is complicated by the underlying polygon representing the recovered shape of the original object.

The solution is based on a combination of the displacement shader creating the shape and the imager shader performing the composition. A value of the background pixel is used only in the case, when the displacement shader did not change the z value of the polygon and when the alpha channel was not changed, otherwise the blending with background occurs, leading to a proper result even for a partly-transparent strands of fur.

5 Conclusion

The aim of this work was to prove that high-level image edits, such as adding of fur to a surface of an object depicted on a photograph, are possible; producing visually plausible results, even though the whole processing is constrained by imperfect depth and lighting information derived directly from the input image. As demonstrated on the set of resulting images in Appendix 7.1 , this aim was fulfilled.

The progress of the work consisted of many steps, and, as it is common in research projects, not all of them led to the desired results. This thesis is aimed to describe the successful way; the details of many other attempts and algorithms that did not provide the required solution, are left unmentioned. However, the following paragraphs will mention at least a few of them, because the wrong ways of a research can be often as important as the right ones.

First of all, the rendering engine itself – the first impression was, that the fur rendering with global illumination, as required by the method, would need to use a new rendering engine designed specially for this purpose, as it was shown in the fur and global illumination papers before. This impression was strengthened even more by the new approaches using the "igloo" method [57] [58], which is only vaguely related to the radiosity rendering methods and very different from all the other methods. However, as the renderer development went on, it has proved to be extremely difficult to solve all the usual rendering issues and global illumination details as well as process the HDR information and also implement the mentioned "igloo" methods mentioned above, which unfortunately required more information than was available in the published papers. For this reason, after almost two months of development of the rendering engine, it was decided to switch from this approach to a classical and well documented Renderman system.

The issue of the fur together with the HDR lighting proved to be rather intriguing too, as also mentioned by the team of researchers behind the HDR fur lighting algorithm as well. The lighting models of fur accounted either for too much light scattering in the hair volume, providing nice results under one lightsource, but proving to be very inaccurate under more complex lighting; or did not account for the scattering at all, providing exactly the opposite results with very strong nonlinearities in the reflectance function. This was the reason behind the implementation of the two different models, for when one of them failed for a particular input image the other one proved to be useful and vice versa.

The other dead ends included the hair distribution model, originally based on poisson-disc distribution, the hair shaping, based on spring and mass-points, volumetric fur representation, and many other small issues encountered on the way.

The precise answer to every point of the original assignment can be offered in the form of the following list:

1. The realistic fur rendering and related shadow algorithms form a large part of this thesis, particularly chapters 2.6 , 2.7 , 3.3 , 4.3 , 4.4 and their subchapters.
2. The same can be written about the HDR imaging methods and image-based illumination techniques, forming chapters 2.1 , 2.3.4 , 4.5.1 and their respective subchapters.
3. Image-based material editing is the very base on which this thesis is built, its relation to HDR is also included, as described in chapters 2.2.3 , 2.4 and 3.1 .

4. The discussion of their combinations is mainly included in the critical-analysis in section 3 ; the description of combinations used in the previous work can be found also in section 2.4 .
5. The description and evaluation of the methods used in implementation constitutes the whole section 4 , and the final assessment of the work is included in this conclusion. The actual results of the processing can be found in appendix 7.1 .
6. The final conclusions and suggestions for the possible future work are placed in this chapter.

The set of final renders, as included in the appendix 7.1 , form a nice example of the capabilities of the resulting method. Even though this thesis lacks a proper perceptual evaluation of the results, which would require a relatively large experimental setup with a number of participants together with real and synthesized versions of the furry objects depicted on images, it clearly shows the possibility of the high-level semi-automatic edits and their potential for future studies.

The future research of this topic can be quite extensive. The first step would be to perceptually evaluate the results not only with the presented method, but also with the rest of the IBME (image-based material editing) techniques, which, although usually built on the perceptual basis, still lack the evaluation of the visual correctness. The extensions of this method would involve implementation of new approaches to the shape extraction which still offers a lot of space for new methods and algorithms, more advanced hair simulations, and tests of new high-level material edits, as observed in the real world and on the large group of superheroes from American film industry, such as stone carving, wood bark (with the geometrical displacement of surface), fire etc. Another very interesting extension could based on the IBME in animations, shown only for texture change and for very consistent movement so far, allowing to change the material of an object (or actor) for the film studios.

6 References

[1] 3Delight 7.0 User's Manual. 2007. Available online on http://radsite.lbl.gov/radiance/refer/filefmts.pdf (27.4.2008)

[2] Adelson, E. H. 2001. On Seeing Stuff: The Perception of Materials by Humans and Machines. *Proceedings of the SPIE*, volume 4299, 2001, 1-12

[3] Ahokas, T. 2002. Shadow Maps. *Helsinky University of technology.*

[4] Apodaca, A. A. and Gritz, L. 1999 *Advanced Renderman: Creating CGI for Motion Picture*. 1st. Morgan Kaufmann Publishers Inc.

[5] Arvo, J. 1991. Real Pixels. Graphics Gems II, *Academic Press*, 80-84

[6] Aslantas, V. 2007. A depth estimation algorithm with a single image. *Optics Express*, volume 15, issue 8, 5024-5029

[7] Banks, D. C. 1994. Illumination in diverse codimensions. In *Proceedings of the 21st Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '94. ACM, New York, NY, 327-334.

[8] Belhumeur, P. N., Kriegman, D. J., and Yuille, A. L. 1999. The Bas-Relief Ambiguity. *Int. J. Comput. Vision* 35, 1 (Nov. 1999), 33-44.

[9] Choe, B. and Ko, H. 2005. A Statistical Wisp Model and Pseudophysical Approaches for Interactive Hairstyle Generation. *IEEE Transactions on Visualization and Computer Graphics* 11, 2 (Mar. 2005), 160-170.

[10] Chang, J. T., Jin, J., and Yu, Y. 2002. A practical model for hair mutual interactions. In *Proceedings of the 2002 ACM Siggraph/Eurographics Symposium on Computer Animation* (San Antonio, Texas, July 21 - 22, 2002). SCA '02. ACM, New York, NY, 73-80.

[11] Chen, J., Paris, S., and Durand, F. 2007. Real-time edge-aware image processing with the bilateral grid. In *ACM SIGGRAPH 2007 Papers* (San Diego, California, August 05 - 09, 2007). SIGGRAPH '07. ACM, New York, NY, 103.

[12] Daldegan, A., Magnenat-Thalmann, N. 1993. Creating virtual fur and hair styles for synthetic actors. *Communicating with Virtual Worlds*, N. Magnenat-Thalmann and D. Thalmann, Eds. Springer-Verlag, 358-370.

[13] Debevec, P, 1997. Recovering High Dynamic Range Radiance Maps from Photographs. Available online on http://www.debevec.org/Research/HDR (11.5.2008)

[14] Drori, I., Cohen-Or, D., and Yeshurun, H. 2003. Fragment-based image completion. *ACM Trans. Graph.* 22, 3 (Jul. 2003), 303-312.

[15] Durand, F. and Dorsey, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. In *Proceedings of the 29th Annual Conference on Computer Graphics and interactive Techniques* (San Antonio, Texas, July 23 - 26, 2002). SIGGRAPH '02. ACM, New York, NY, 257-266.

[16] Eisemann, E. and Durand, F. 2004. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 673-678

[17] Fang, H. and Hart, J. C. 2004. Textureshop: texture synthesis as a photograph editing tool. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 354-359.

[18] Ferwerda, J. A., Shirley, P., Pattanaik, S. N., and Greenberg, D. P. 1997. A model of visual

masking for computer graphics. In *Proceedings of the 24th Annual Conference on Computer Graphics and interactive Techniques* International Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., New York, NY, 143-152.

[19] Fished, B., Perkins, S., Walter, A., Wolfart, E.: Spatial Filters – Gaussian Smoothing. *Department of Artificial Inteligence, University of Edinburgh*, Available online on http://www.cee.hw.ac.uk/hipr/html/gsmooth.html (24.5.2007)

[20] Forsyth, D. A., Ponce, J. 2003. Computer Vision: A Modern Approach. *Pretince Hall.*

[21] Freeman, W. T. 1996. Exploiting the generic viewpoint assumption. *Int. J. Comput. Vision* 20, 3 (Dec. 1996), 243-261.

[22] Goldman, D. B. 1997. Fake fur rendering. In *Proceedings of the 24th Annual Conference on Computer Graphics and interactive Techniques* International Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., New York, NY, 127-134.

[23] Guo, Y., Wang, J., Zeng, X., Xie, Z., Sun, H., and Peng, Q. 2005. Image and video retexturing: Image, Colour and Illumination in Animation. *Comput. Animat. Virtual Worlds* 16, 3-4 (Jul. 2005), 451-461.

[24] Halton, J. H. 1964. Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM* 7, 12 (Dec. 1964), 701-702.

[25] Heckbert, P. S. 1997. Fast Surface Particle Repulsion. *In SIGGRAPH '97, New Frontiers in Modeling and Texturing Course*, ACM Press, 95-114.

[26] Jobson, D. J. Rahman, Z., and Woodell, G. A. 1997. A multiscale Retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing* 6, 965-76.

[27] Kajiya, J. T. 1986. The rendering equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 143-150.

[28] Kajiya, J. T. and Kay, T. L. 1989. Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph.* 23, 3 (Jul. 1989), 271-280.

[29] Khan, E. A., Reinhard, E., Fleming, R. W., and Bülthoff, H. H. 2006. Image-based material editing. *ACM Trans. Graph.* 25, 3 (Jul. 2006), 654-663.

[30] Koenderink, J. J., van Doorn, A. J., Kappers, A. M., and Todd, J. T. 2001. Ambiguity and the 'mental eye' in pictorial relief. *Perception* 30, 431-448.

[31] Koenderink, J. and Van Doorn, A. 1979. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211-216.

[32] Langer, M. S., and Bülthoff, H. H. 2000. Depth discrimination from shading under diffuse lighting. *Perception* 29, 6, 649-660.

[33] Lengyel, J., Praun, E., Finkelstein, A., and Hoppe, H. 2001. Real-time fur over arbitrary surfaces. In *Proceedings of the 2001 Symposium on interactive 3D Graphics* I3D '01. ACM, New York, NY, 227-232.

[34] Lokovic, T. and Veach, E. 2000. Deep shadow maps. In *Proceedings of the 27th Annual Conference on Computer Graphics and interactive Techniques* International Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., New York, NY, 385-392.

[35] Magnenat-Thalmann, N., Hadap, S., and Kalra, P. 2000. State of the Art in Hair Simulation. *International Workshop on Human Modeling and Animation*, 3-9.

[36] Marschner, S. R., Jensen, H. W., Cammarano, M., Worley, S., and Hanrahan, P. 2003. Light scattering from human hair fibers. *ACM Trans. Graph.* 22, 3 (Jul. 2003), 780-791.

[37] McMillan, L. 1997 *An Image-Based Approach to Three-Dimensional Computer Graphics*. Doctoral Thesis. UMI Order Number: UMI Order No. GAX97-30561., University of North Carolina at Chapel Hill.

[38] Oh, B. M., Chen, M., Dorsey, J., and Durand, F. 2001. Image-based modeling and photo editing. In *Proceedings of the 28th Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '01. ACM, New York, NY, 433-442.

[39] Oliviera, M. M. 2002. Image-Based Modelling and Rendering Techniques: A Survey. *RITA - Revista de Informática Teórica e Aplicada*, volume 9, number 2, 37-66

[40] Ostromoukhov, V., Donohue, C., and Jodoin, P. 2004. Fast hierarchical importance sampling with blue noise properties. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 488-495.

[41] Ostrovsky, Y., Cavanagh, P., and Sinha, P. 2005. Perceiving illumination inconsistencies in scenes. *Perception 34*, 11, 1301-1314.

[42] Paris, S., Durand, F. 2006. A fast approximation of the bilateral filter using a signal processing approach. In *Proceedings of the European Conference on Computer Vision* 2006..

[43] Porter, T. and Duff, T. 1984. Compositing digital images. *SIGGRAPH Comput. Graph.* 18, 3 (Jul. 1984), 253-259.

[44] Ramanarayanan, G., Ferwerda, J., Walter, B., and Bala, K. 2007. Visual equivalence: towards a new standard for image fidelity. *ACM Trans. Graph.* 26, 3 (Jul. 2007), 76.

[45] Reinhard, E., Ward, G., Pattanaik, S., Debevec, P. 2006. High Dynamic Range Imaging – Acquisition, Display and Image-Based Lighting. *Morgan Kaufmann*.

[46] Sun, J., Yuan, L., Jia, J., and Shum, H. 2005. Image completion with structure propagation. In *ACM SIGGRAPH 2005 Papers* (Los Angeles, California, July 31 - August 04, 2005). J. Marks, Ed. SIGGRAPH '05. ACM, New York, NY, 861-868.

[47] Tchou, C., Debevec, P. 2001. Technical Sketch on HDR Shop. *SIGGRAPH 2001 Conference Abstracts and Applications*, 195-196

[48] The Renderman Interface version 3.2.1. Available online on https://renderman.pixar.com/products/rispec/rispec_pdf/RISpec3_2.pdf (17.5.2008).

[49] Tomasi, C. and Manduchi, R. 1998. Bilateral Filtering for Gray and Color Images. In *Proceedings of the Sixth international Conference on Computer Vision* (January 04 - 07, 1998). ICCV. IEEE Computer Society, Washington, DC, 839.

[50] Using the TIFF library. Available online on http://www.libtiff.org/libtiff.html (17.5.2008)

[51] Ward, G. J.: Radiance File Formats. Available online on http://radsite.lbl.gov/radiance/refer/filefmts.pdf (24.5.2007)

[52] Ward, G. J. 1994. The RADIANCE lighting simulation and rendering system. In *Proceedings of the 21st Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '94. ACM, New York, NY, 459-472.

[53] Ward, K., Bertails, F., Kim, T., Marschner, S. R., and Cani, M. 2007. A Survey on Hair Modeling: Styling, Simulation, and Rendering. *IEEE Transactions on Visualization and Computer Graphics*

13, 2 (Mar. 2007), 213-234.

[54] Wikipedia, the free encyklopedia - sRGB color space. *Wikipedia, the free encyklopedia.* Available online on http://en.wikipedia.org/wiki/Srgb (7.6.2007)

[55] Witkin, A. P. and Heckbert, P. S. 1994. Using particles to sample and control implicit surfaces. In *Proceedings of the 21st Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '94. ACM, New York, NY, 269-277.

[56] Yamakawa S., Shimada, K. 2004. Triangular / Quadrilateral Remeshing of an Arbitrary Polygonal Surface via Packing Bubbles. In *Proceedings of the Geometric Modeling and Processing* 2004, 153-162.

[57] Yuksel, C., Akleman, E.: Rendering Hair-Like Objects with Indirect Illumination. In *ACM SIGGRAPH 2005 Sketches*, 2005

[58] Yuksel, C., Akleman, E.: Hair Occlusion: Rendering Hair-Like Objects with Global Illumination. *ACM SIGGRAPH'2006 Poster*, 2006

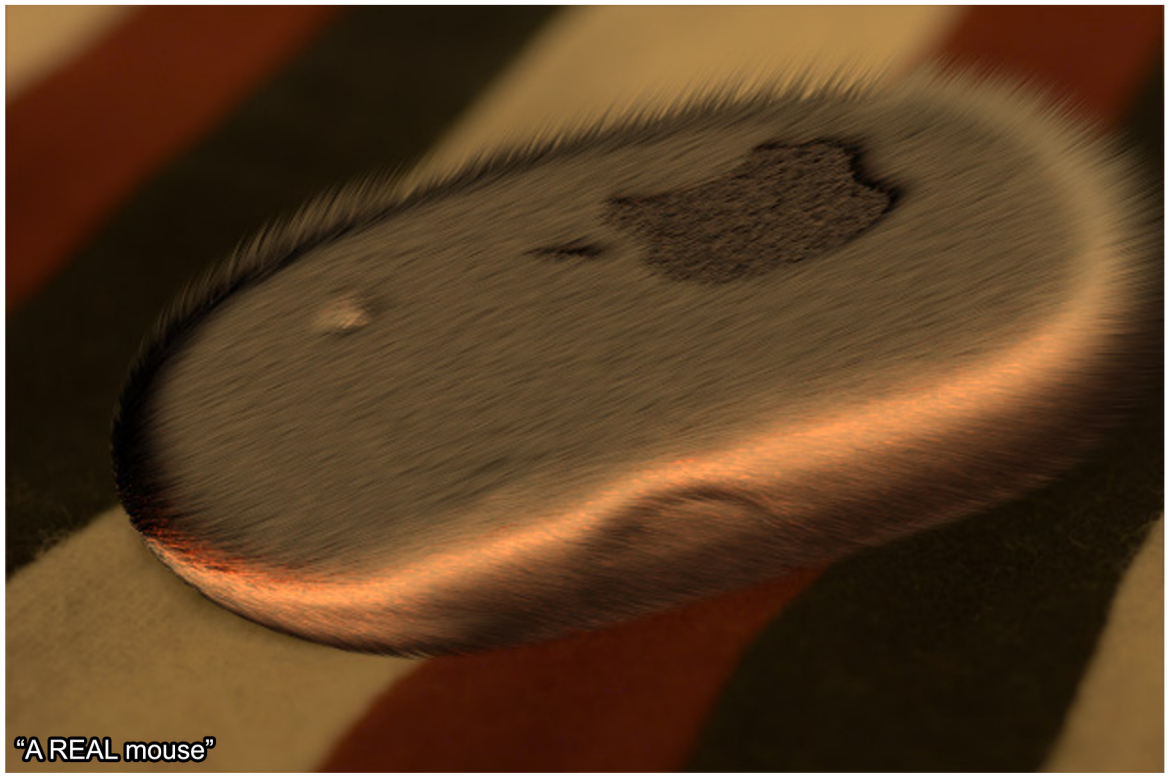
[59] Zelinka, S., Fang, H., Garland, M., and Hart, J. C. 2005. Interactive material replacement in photographs. In *Proceedings of Graphics interface 2005* (Victoria, British Columbia, May 09 - 11, 2005). ACM International Conference Proceeding Series, vol. 112. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, 227-232.

[60] Zhang, R., Tsai, P., Cryer, J. E., and Shah, M. 1999. Shape from Shading: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 21, 8 (Aug. 1999), 690-706.

[61] Zinke, A. 2007. Light Scattering from Filaments. *IEEE Transactions on Visualization and Computer Graphics* 13, 2 (Mar. 2007), 342-356.

7 Appendices

7.1 Resulting images


“A REAL mouse”


Original image


Without the length map

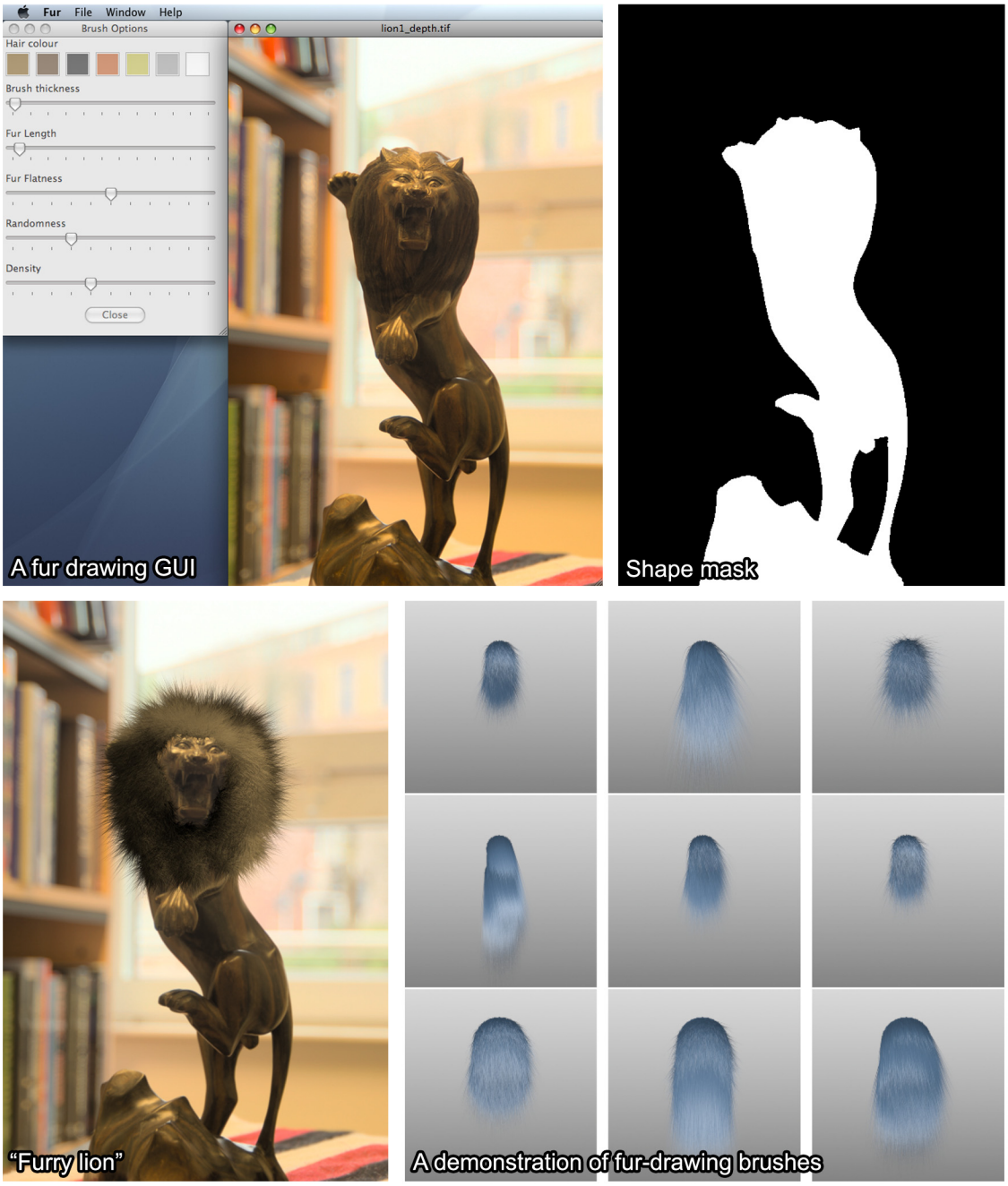

Shape mask


Less direction randomness


More direction randomness


Fur length mask 1


Fur length mask 2



The fur-drawing GUI is a project currently being developed by Tania Pouli as a continuing research on the topic of this thesis. The relevancy to this thesis is based on the fact, that this project makes use of the base code and rendering process developed in this work.

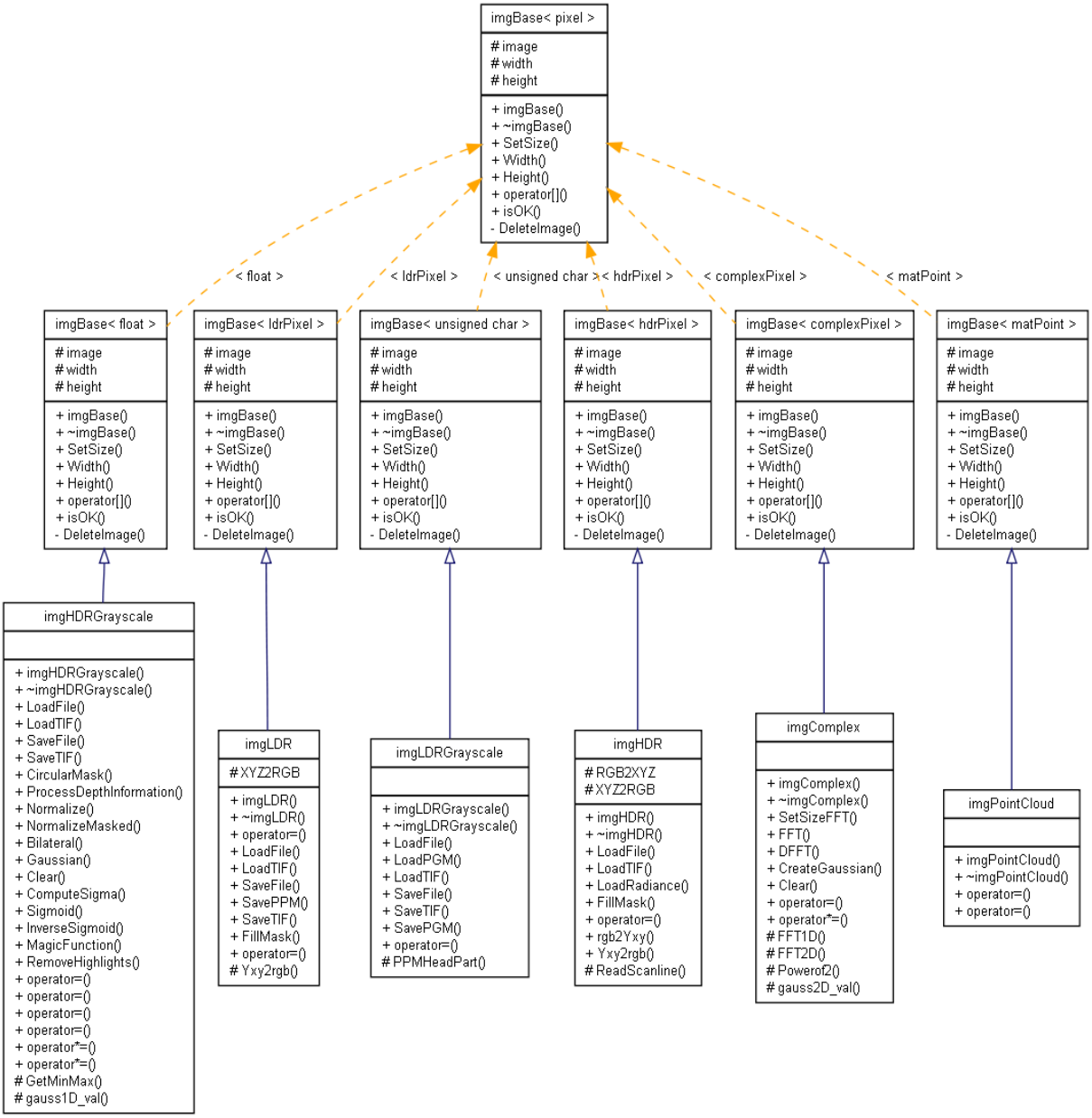
For more information please contact Tania Pouli at pouli@compsci.bristol.ac.uk

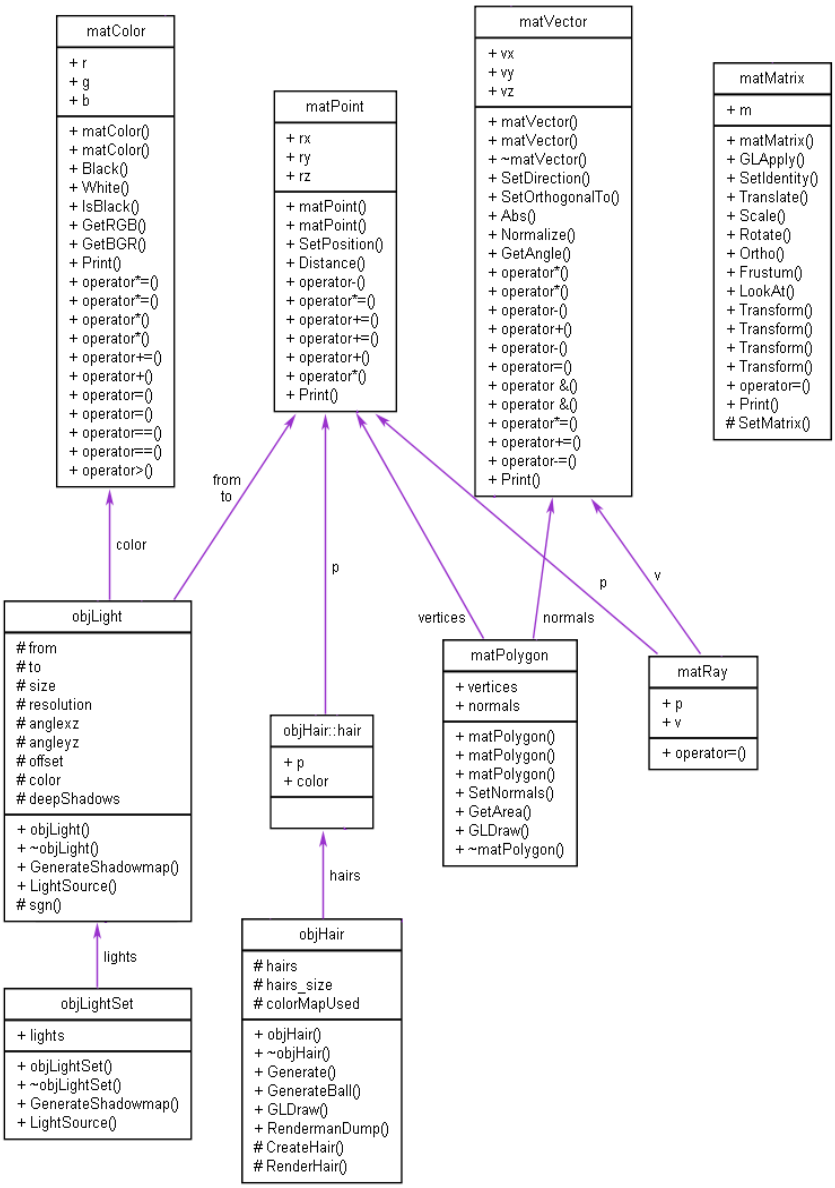
7.2 Configuration file parameters description

Parameter name	Description	Default value	Value interval
Filename-related parameters			
file	filename of the input image	(required)	-
mask	grayscale mask separating the object of interest	(required)	-
fur_mask	grayscale mask changing the density value	none	-
length_map	grayscale mask modifying the fur length	none	-
color_map	RGB color image specifying the hair diffuse component	none	-
background_image	an alternative LDR background image	tonemapping	-
output_prefix	the prefix of the temporary and output filenames	"renderman_fur"	-
Depth-recovery parameters			
l_hmax	lower highlight edge for highlight removal	1.0	0.0 – 1.0
l_beta	exponent in highlight removal equation	20	1 – 50
spat_kernel	spatial kernel size for bilateral filter, related to shorter edge of the image	(required)	0.001 – 1.0
int_kernel	intensity kernel size for bilateral filter, related to full intensity interval	(required)	0.001 – 1.0
magic_iterations	number of iterations of "magic" gradient-shaping funtion	(required)	0 – 5
depth_multiply	depth scaling factor applied to the result of depth recovery	1.0	0.0 – 1.0
Depth-image sampling and shadowing parameters			
magnitude	magnitude of the sampling algorithm influencing the number of sampled points (lightsources), depends on dynamic range of the input image	(required)	0 – 100 000
intensity	light sources intensity correction (for proper LDR rendering)	(required)	0.1 – 50
shadowmap_resolution	resolution of shadowmap buffers, should be only values formed as powers of 2	512	256, 512, 1024, 2048, 4096
deep_shadows	set to 1 if deep shadow algorithm is to be used, 0 for standard shadowmaps	(required)	0, 1
Additional lightsource parameters			
test_light	1 = add a distant test light	0	0, 1
test_light_x **test_light_y** **test_light_z**	direction vector of the distant test light, normalized if not unit	[0, 0, -1]	any nonzero vector
test_light_r **test_light_g** **test_light_b**	the RGB colour of the test light source	[1, 1, 1]	0 – 1 for every channel
Hair shape parameters			
z_coef	multiplier of *z* axis for density computation, 0 - ignore *z* axis, 1 – use z axis from recovered shape	1	0 – 1
hair_density	hair density per unit area	(required)	0 – 1 000 000
hair_length	length of each hair strand	(required)	0 – 0.1
hair_width	width of each hair strand	(required)	0 – 0.02
g_strength	strength of the shaping vector force	2	0 – 5
g_x **g_y** **g_z**	direction vector of the shaping force, normalized if not unit	[0, 1, 0]	any nonzero vector

Parameter name	Description	Default value	Value interval
Common lighting model parameters			
fur_opacity	fur opacity at the root	0.5	0 – 1
lighting model	lighting model switch, 0 – Kajiya / Kay, 1 – Marschner et al.	0	0, 1
Kajiya-Kay lighting model			
fur_diffuse_r **fur_diffuse_g** **fur_diffuse_b**	RGB colour of the diffuse component	[0.3, 0.3, 0.3]	0 – 1 for every channel
fur_diffuse_power	power value for diffuse component computation	10	1 – 10 000
fur_specular_r **fur_specular_g** **fur_specular_b**	RGB colour of the specular component	[1, 1, 1]	0 – 1 for every channel
fur_specular_power	power value for specular component computation	1000	1 – 10 000
Marschner et al. lighting model			
alpha_R	longitudinal shift of R lobe	-5°	-10° - -5°
beta_R	longitudinal width of R lobe	10°	5° - 10°
eta	hair index of refraction	1.55	1.0 – 1.9
sigma_a_R **sigma_a_G** **sigma_a_B**	colour attenuation for R, G and B component	[0.44, 0.64, 0.9]	0.2 – infinity for every channel
k_G	glint scale factor	2	0.5 – 5.0
w_c	azimuthal width of caustics	25°	10° - 25°
deta	caustics fade range	0.4	0.2 – 0.4
dh_M	caustics intensity limit	0.5	0.1 – 1.0
a	eccentricity of hair cross-section	0.85	0.85 – 1.0
rotation_frequency	frequency of random rotations along the hair	3	0.0 – 10.0
rotation amplitude	maximal amount of random rotation changes	3.14	0.0 – 6.28
multiply_constant	multiplier of computed light values	2	1.0 – 100.0
Underlying surface colour parameters			
paint_over_underlying	switch for paining the underlying surface, 0 – paint the surface with plastic shader, 1 – leave the original colours	0	0, 1
underlying_r **underlying_g** **underlying_b**	RGB colour of the underlying surface	[1, 1, 1]	0 – 1 for every channel
underlying_ks	specularity constant for underlying surface colour	1	0.0 – 1.0
underlying_kd	diffuse constant for underlying surface colour	0.3	0.0 – 1.0
underlying_ka	ambient constant for underlying surface colour	0	0.0 – 1.0

7.3 Class diagrams of the base structures





7.4 Content of the attached CD

The CD contains the final version of the software (without a large number of temporal testing programs, whose outputs were used for many images in the fourth chapter) in the *Source code* directory, together with a readme file *__readme__.txt* contatining information and requirements for compiling and running the software. Directory *Program documentation* contains HTML documentation generated using Doxygen tool; directory *Examples* contains a set of input files (*Ivory Statue*, *MAC Mouse*, *Tania Coat*), corresponding configuration files and a set of resulting images for every source. Last directory *Thesis text* contains the OpenOffice (version 3.0) source file and PDF version of this text.