Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

*Volume 38* (*2019*), *Number 7*

# Selecting texture resolution using a task-specific visibility metric

K. Wolski[1], D. Giunchi[2], S. Kinuwaki, P. Didyk[3], K. Myszkowski[1] and A. Steed[2], R. K. Mantiuk[4]

[1]Max Planck Institut für Informatik, Germany
[2]University College London, England
[3]UniversitÃă della Svizzera italiana, Switzerland
[3]University of Cambridge, England

**Abstract**

*In real-time rendering, the appearance of scenes is greatly affected by the quality and resolution of the textures used for image synthesis. At the same time, the size of textures determines the performance and the memory requirements of rendering. As a result, finding the optimal texture resolution is critical, but also a non-trivial task since the visibility of texture imperfections depends on underlying geometry, illumination, interactions between several texture maps, and viewing positions. Ideally, we would like to automate the task with a visibility metric, which could predict the optimal texture resolution. To maximize the performance of such a metric, it should be trained on a given task. This, however, requires sufficient user data which is often difficult to obtain. To address this problem, we develop a procedure for training an image visibility metric for a specific task while reducing the effort required to collect new data. The procedure involves generating a large dataset using an existing visibility metric followed by refining that dataset with the help of an efficient perceptual experiment. Then, such a refined dataset is used to retune the metric. This way, we augment sparse perceptual data to a large number of per-pixel annotated visibility maps which serve as the training data for application-specific visibility metrics. While our approach is general and can be potentially applied for different image distortions, we demonstrate an application in a game-engine where we optimize the resolution of various textures, such as albedo and normal maps.*

**CCS Concepts**
• ***Computing methodologies*** → ***Perception;*** *Image manipulation; Image processing;*

## 1. Introduction

Since the beginning of computer graphics, the complexity of rendered scenes evolved from small environments with basic shading to realistic-looking large-scale virtual worlds that often require days or weeks to fully explore. One of the critical components of these scenes is textures storing local color information in addition to other attributes such as surface normals, geometry displacement or reflectance properties. Since all of them significantly contribute to the faithful reproduction of real-world appearance, their complexity as well as quality is crucial for the final experience provided to a user. To address these quality demands, graphics vendors develop more complex tools, such as physically based texture painter [DOS16] or terrain generation software [Car18], which simplify the process of enriching textures with details, and therefore, increase the photo-realism of produced scenes.

Since one object may consist of many different parts and materials, the size of all textures can quickly increase with the complexity of the scene. As a consequence, many games released in recent years require a lot of memory space to store all the material attributes, e.g., Final Fantasy XV (170 GB) or Gears of War 4 (250 GB). Large textures also impact the overall performance by drastically increasing the loading time and memory bandwidth to GPU, and therefore, contributing to the reduced efficiency of a rendering engine. Textures size becomes an even more significant problem in the context of new ultra-resolution displays or virtual reality (VR) devices, where the quality demands with respect to spatial and temporal resolutions are pushed to the limits, or in the context of mobile devices, where the computational and memory budgets are very tight [Epi18].

Texture compression and resampling are critical preprocessing steps which address the above problem. They can balance the quality and memory requirements by trying to keep the scene description in the GPU memory as much as possible while minimizing quality loss [GO15, LLPP18]. However, finding a good set of parameters for these operations is a non-trivial task. The reason for this is twofold. First, the visibility of artifacts in one texture (e.g., albedo texture) is affected by the information in other textures (e.g., normal texture), mesh geometry, illumination, and viewing direction. Therefore, all the information should be taken into account. Second, there is significant variability in platforms to which renderings are ported. The differences include display size, resolution, and brightness, but also memory and performance limits [Epi18]. Therefore, it is critical to facilitate the process of compressing and resampling textures using an automatic process. This goal can be achieved using image metrics which are used to predict the visibility of distortion for different parameters sets. Unfortunately, as demonstrated in this as well as previous work [GO15, LLPP18], the performance of existing metrics is suboptimal with a significant room for improvements.

In this work, we consider the problem of automatic prediction of an optimal texture resolution for storing albedo and normal information during real-time rendering. The key challenge we address is designing a reliable and precise image metric for detecting visible distortions. While several general-purpose metrics have been proposed [MKRH11, WGY$^*$18], we demonstrate that their prediction can be significantly improved when they are trained on a task-specific dataset. The problem, however, is that retraining such metrics from scratch involves collecting a massive amount of experimental data from labor-intensive user experiments such as marking experiments [WGY$^*$18]. To address this issue, we demonstrate that one can retune existing metrics using the data from much simpler and shorter user experiments. The main contribution of this paper is a strategy for retuning an existing CNN-based visibility metric [WGY$^*$18] (CNN-GP) with a new dataset for improving the accuracy of prediction. CNN-GP metric was trained on a large LocVis dataset[†] with locally marked distortions [WGY$^*$18], and shown to outperform existing VMs, even when retrained on the same dataset. We demonstrate the data required for retuning CNN-GP can be collected in a modest-in-scope perceptual experiment so that it becomes a reliable metric for selecting texture resolution. While this paper demonstrates the performance of such procedure in the context of texture optimization for game engines, our experiment design and retuning can be potentially used also in different applications.

## 2. Related work

Image metrics have been employed in many graphics applications to improve their performance without affecting the visual quality [OHM$^*$04, WSR$^*$17]. In this section, we discuss the problem of image quality evaluation (Section 2.1) with an emphasis on texture compression for 3D textured models (Section 2.2) and complex texture sets (Section 2.3) that are used in modern interactive rendering systems to achieve complex object appearance. While here we focus on full reference (FR) image metrics commonly used in computer graphics, a complete survey including also non-reference (NR) metrics can be found in [Cha13].

### 2.1. Image quality and visibility metrics

It is important for our considerations to make the distinction between image *quality* and *visibility* metrics. Image *quality* metrics (IQMs) such as PSNR, CIELAB, SSIM and MS-SSIM [WB06] produce a single score that corresponds to the distortion magnitude. IQMs are typically trained on the mean-opinion score (MOS) data [SSB06], which reflects the level of viewer annoyance by such distortions. *Visibility* metrics (VMs) predict the perceptibility of distortions for every pixel, and typically rely on models of early human vision (VDM [Lub95], VDP [Dal93], HDR-VDP-2 [MKRH11]), or differences in custom feature maps (Butteraugli [AOS$^*$17]). VMs are a better choice on estimating texture compression distortion as they are precise for near-threshold distortions as opposed to IQMs, which are specialized in estimating the supra-threshold distortion magnitude. As reported by Cadik et al. [ČHM$^*$12], such general-purpose VMs might not be suitable for reliable detection of rendering artifacts in complex images. However, their performance can significantly be improved when retrained on artifact-specific and per-pixel-accurate perceptual data [ČHM$^*$12, AVS$^*$17, WGY$^*$18].

Machine-learning based image metrics, which often rely on convolutional neural networks (CNN) [KSH12], are capable of learning important characteristics of the human vision in a “black-box” style, and were shown to achieve the best performance [BMM$^*$18, WGY$^*$18, ZIE$^*$18]. However, their ability to generalize over artifacts that are not present in the training data is limited. Also, the prediction accuracy typically depends on the volume of training data. Collecting such data for VMs is a very labor-intensive task as they require per-pixel labeling of the distortion visibility in complex images. The key contribution of this work is a method for improving the performance of such data-driven metrics for a particular application.

### 2.2. Level-of-details for textured 3D models

Controlling level-of-detail (LoD) for textured objects poses similar challenges as our application. Mesh simplification introduces complex interactions between texture and geometry, which influence the visibility of distortions [LM15]. Initial solutions relied on simple geometric error measures such as *texture deviation*, which accounts for a given texel motion in the object space as a result of mesh simplification [COM98]. Following a general observation that image-space metrics often lead to more robust quality predictions [LT00], Tian and AlRegib proposed an image-space approximation of texture deviation [TA04].

The visual masking of geometry details by texture has been widely investigated [FSPG97, QM08, HER00], and complex perceptual image-space models inspired by VDM and VDP have been proposed. Walter et al. consider using rendered textures as a masker and create a predictor based on the quantization matrices found in JPEG image compression [WPG02]. Computationally efficient solutions have been proposed, where image-space visual masking processing was moved to vertex space [MG10], or just a simple contrast sensitivity function (CSF) was considered [WLC$^*$03].

Early data-driven approaches relied on perceptual experiments that systematically measured the impact of mesh simplification and texture resolution reduction on the perceived quality of 3D models. A quantitative model that predicts the MOS data that were obtained this way has been proposed in [PCB05]. Guo et al. [GVC$^*$16] used slowly moving animation in a 2-alternative-forced-choice (2AFC) experiment to account for multiple views. Based on the collected data they propose a quality metric that linearly combines object-geometry (e.g., the standard deviation of curvature differences) and

[†] https://doi.org/10.17863/CAM.21484

image-space (MS-SSIM) components to account for geometric and texture distortions.

As found by Larkin and O'Sullivan [LO11] silhouette simplification is the dominant artifact that should drive mesh simplification. In this work, we assume that mesh simplification is properly set up and we focus on texture and shading artifacts in complex scenes, while still considering the geometry impact on the visibility of such artifacts.

### 2.3. Texture compression masking

As observed in [GO15, LLPP18], texture compression is essential to keep the description of large scenes in the GPU memory. Multiple textures describing surface reflectance (diffuse, gloss, and specular components) and geometry (normal vectors for normal mapping effects) are required to achieve high-quality shading. Such textures interact not only with mesh geometry, as discussed in the previous section, but also mutually mask each other as a function of lighting and view directions.

Griffin et al. [GO15] investigated this effect and advocated for evaluating texture compression directly in the scene environment for stochastically sampled viewpoints. They observed significant differences in CIELAB $\Delta E_{94}$ and SSIM metric predictions with varying viewpoints, texture type and content, which precludes a unified compression setup for all textures. They concluded that current GPU compression algorithms are too conservative, and the bit rates can be significantly reduced while maintaining the rendering quality.

Lavoué et al. [LLPP18] performed a series of perceptual experiments to investigate the masking problem between diffuse and normal textures. They ran a 2AFC experiment to derive the compression visibility threshold with respect to non-compressed reference textures. They used planar texture patches with a fixed 128 × 128 resolution and 5 different compression levels as stimuli. They reported a limited performance of existing metrics such as SSIM, MS-SSIM, HDR-VDP2, and PSNR in predicting the perceptual data, in particular for the normal texture compression. They compared the SSIM prediction for illuminated complex geometry and planar patches (as in their experiment), and obtained a mean correlation of 0.59, which is significantly better than a naive comparison of 2D compressed textures where only 0.47 correlation was achieved. However, it can be expected that the actual correlation between the SSIM prediction for the planar patches and the human subject judgment for complex geometry and lighting can be even lower.

In this work, we draw inspiration from these previous works. We follow the approach of Griffin et al. and use rendered images to obtain a robust measure of artifact visibility. Our rigorous 4AFC threshold measurement procedure is similar to the one used by Lavoué et al. However, we extend beyond those works by proposing a task-specific visibility metric for texture resolution selection. We achieve this goal by retuning an existing CNN-based metric and propose an inexpensive way of collecting perceptual data that are required for this task.

## 3. Proposed approach

The goal of our work is to provide a technique for choosing texture resolution used for rendering complex game environments. We cannot directly use a state-of-the-art visibility metric, such as the CNN metric proposed in [WGY*18] (referred as CNN-GP) or HDR-VDP-2 [MKRH11], as they are designed under the assumption that the observer takes an arbitrary amount of time to attend each region of an image. We observed that this assumption is too conservative for our target application of a video game, in which a player immersed in a virtual environment is less likely to spot every small difference.

A possible solution to the above problem is to retrain the metric using a new application-specific dataset. However, collecting new experimental data with local marking (which is required for a visibility metric) for many objects, textures, and viewing position is impractical due to the amount of time needed for subjective studies. Moreover, it is unclear how to design a marking experiment that would reflect limited observation time expected in a game-like scenario.

In this work, we demonstrate that it is possible to retune existing CNN-GP metric using a much smaller, application-specific dataset, collected using a more efficient experimental procedure. To this end, we collect a new dataset[‡] in Experiment 1, in which the observers are asked to determine the smallest texture size for which quality degradation is not visible (Section 4). We also collect a smaller dataset of local visibility maps in Experiment 2, which we use only for validation (Section 5). Since the dataset from Experiment 1 provides only a single detection probability per image, we cannot use it directly to retune the CNN-GP metric, which requires local information. Our key idea to overcome this problem is to exploit existing visibility metrics, such as Butteraugli, HDR-VDP-2, or CNN-GP, and use them to produce dense visibility maps. While these maps are inaccurate, we demonstrate that it is possible to adjust them using our new dataset. To do that, we first find a pooling function (Section 6), which relates the predicted visibility maps to the collected subjective data. Then, we generate a large set of visibility maps using three existing visibility metrics and adjust them using the data collected in Experiment 1 (Section 7). Given the large number of locally labelled images, we can much improve the prediction performance for our dataset (Section 8) and demonstrate good performance in our application (Section 9). The schematic diagram of our pipeline is illustrated in Figure 1.

## 4. Experiment 1: Detection thresholds

The purpose of our first experiment was to collect data regarding the minimum texture resolution that does not introduce visible artifacts in rendered objects.

**Stimuli** We chose 33 complex 3D models from Unreal Market place. They include characters, creatures, buildings, and objects. Each of them has detailed geometry, high-resolution textures, and uses complex materials. The objects were placed in different environments that are shipped with Unreal Engine, which allowed us to

[‡] https://doi.org/10.17863/CAM.43380

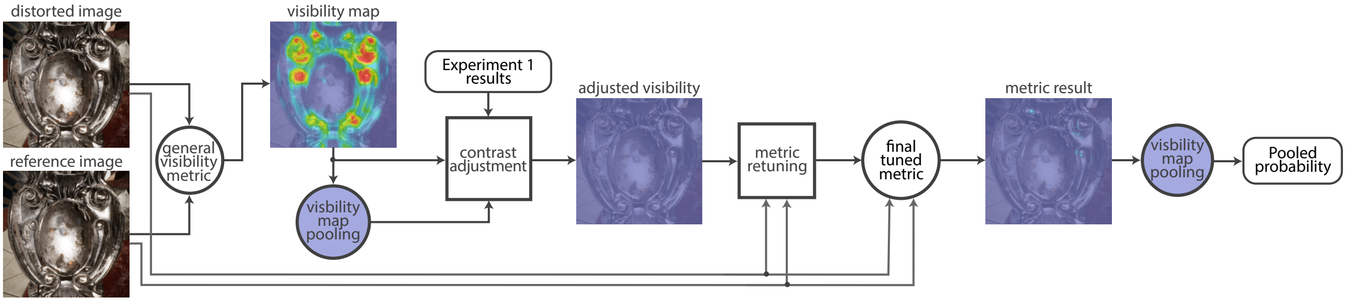


**Figure 1:** *The procedure for retuning a visibility metric. The reference and distorted images are used to generate the distortion visibility maps by one of existing VMs. Then, the thresholds from an application-specific perceptual experiment and pooled probabilities are used to adjust the visibility in the map. The resulted data is used for retuning a CNN-based metric that computes application-specific visibility maps. To asses the optimal texture resolution for target application the resulting visibility map is pooled to a single artifact detection probability value using same pooling strategy as one used for visibility adjustment.*

analyze the objects' appearance under different illumination conditions: directional light, point light, spotlight, skylight, and precomputed light probe. For each object, we considered one or two unique camera positions to capture texture details and shading variations. We rendered the scenes using the Unreal Engine 4 [Kar13]. The target objects were outlined with a red stroke to prevent them from blending with the background.

We created stimuli for the experiment by rendering the scenes with different resolutions of albedo and normal maps. We started with a resolution of 1024×1024 pixels for each texture, and we decreased it in 20-pixel steps down to 24×24 pixels. To limit the number of stimuli, we considered five different downscaling scenarios: reducing the resolution of (1) the albedo map only, (2) the normal map only, (3) both the normal and the albedo maps by the same amount, (4) the normal map twice as much as the albedo map, and (5) the albedo map twice as much as the normal map. The downscaling was performed using Lanczos filter. Using the above procedure, we generated 127 scenes with 51 combinations of resolutions for albedo and normal maps, which resulted in 6477 images. An example of stimuli is presented in Figure 2.

For rendering, we did not consider texture compression as it was done in [LLPP18]. Instead, we used a non-compressed RGB format due to Unreal Engine 4 design philosophy to refrain from texture compression until final packaging. This philosophy is motivated by the fact that each platform has different requirements regarding CPU, GPU, memory, and texture format. Another concern is that a mipmap level alone could determine the optimum texture resolution. For example, if the rendered image refers at most to the mipmap level 1 and never refers to mipmap level 0 (the highest resolution), we know that the resolution could be reduced by half. For that reason we selected the maximum texture resolution so that mipmap level 0 is used for the most of the rendered pixels. After ensuring this, we disabled mipmapping as UE4 does not support mipmapping for non-power-of-two textures (see the discussion and examples in the supplementary).

**Experimental procedure** Each session of our experiment was split into two phases: a method-of-adjustment (MoA) phase, giving an initial estimate of the detection threshold, and a four-alternative-forced-choice (4AFC) phase, which refined the initial estimates. In the MoA phase, the observer's task was to adjust the texture resolution with the up/down arrow keys, so that the difference with respect to the reference is barely visible. A screenshot from that phase is shown in Figure 3. When switching between two texture resolutions, a short (0.5 second) blank screen with the color of the background was shown so that the observers could not use temporal changes to locate the artifacts.

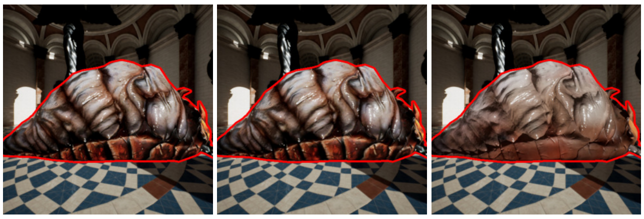
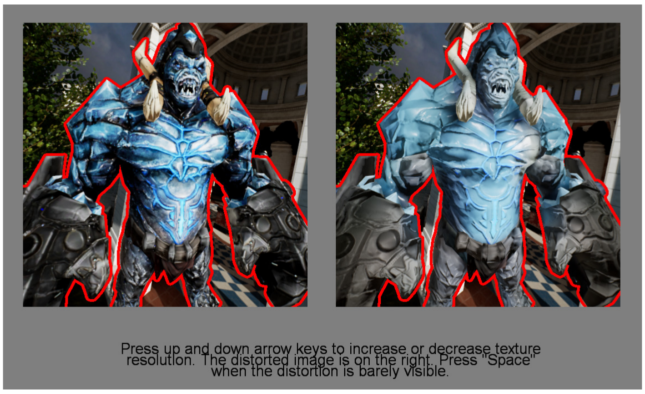


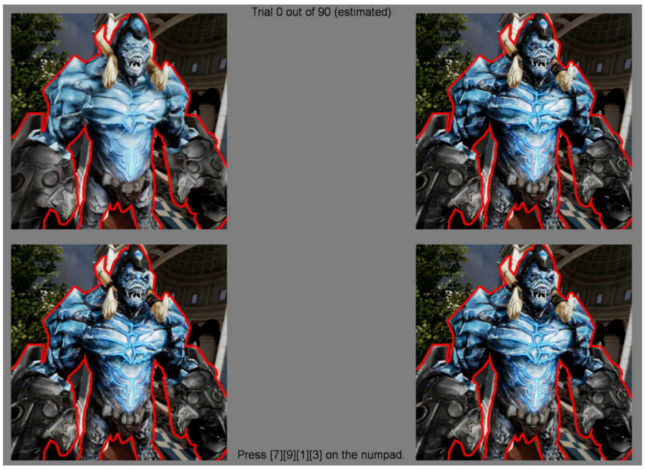
**Figure 2:** *Example of one scene from the dataset with 3 texture resolution variants of albedo map: highest resolution (left), medium resolution (middle), and lowest resolution (right). The target object in the scene has a red outline to prevent it from blending with the background.*

The second 4AFC phase of the experiment was a modified QUEST procedure [WP83]. Four images were presented to the observer, where only one image was distorted. The position of the distorted images was randomized. The observer was asked to choose the image that was different from the others. We used 4AFC as it needs fewer trials to converge than 2AFC because of the lower guess rate, while it gives similar results [VRH*18]. The QUEST adaptive method was initialized with the threshold estimate collected using the MoA method in the first phase. The QUEST trials for different scenes were interleaved so that each trail typically showed a different scene. This was done to prevent the observer from learning the location of distortions. Between 25 and 30 4AFC



Press up and down arrow keys to increase or decrease texture resolution. The distorted image is on the right. Press "Space" when the distortion is barely visible.

**Figure 3:** *Screenshot from the method-of-adjustment (MoA) phase of the experiment. On the left is the reference image rendered with the highest texture resolution, and on the right is the distorted image with reduced texture resolution.*



Trial 0 out of 90 (estimated)

Press [7][9][1][3] on the numpad.

**Figure 4:** *Screenshot from the 4AFC phase of the experiment. The observer's task is to choose the image that is different from the others.*

trials were collected from each observer for each scene. The presentation time of each trial was limited to 10 seconds to ensure that the data reflects a more realistic scenario of viewing rendered models in a game environment. The screenshot from the second phase is shown in Figure 4.

**Viewing conditions** The experiment room was darkened and the monitor was positioned to minimize screen reflections. The observers sat 96 cm from a 24″, 3840×2160 resolution Eizo CH246-4K display, resulting in an angular resolution of 120 pixels per visual degree. As such resolution is much higher than for a standard monitor seen from a typical viewing distance (45–60 pixels per visual degree), we enlarged the images to twice their original size using the nearest neighbor filter. The measured peak luminance of the display was 120 cd/m$^2$ and the black level was 0.17 cd/m$^2$. The display was calibrated to the sRGB color space using its built-in self-calibration functionality.

**Observers** To reduce the effect of fatigue, the experiment was split into 9 sessions, where each session lasted less than one hour. Each participant completed a subset of these sessions so that the threshold for each scene was measured by 10 participants. All observers (aged 21 to 27) were computer science students or researchers. The observers were paid for their participation. All observers had normal or corrected-to-normal vision and were naive to the purpose of the experiment.

### 4.1. Cross-population psychometric function

To use the data collected in the experiment for training, we need to estimate a cross-population psychometric function: the function that explains what portion of the population can see the difference at a particular texture resolution for a given scene, as shown in Figure 5. To find such a function, we first fitted a psychometric function per each observer to the QUEST trial responses and estimated a 75%-probability threshold. Then, the outliers were removed by eliminating per-observer thresholds that were more than two standard deviations away from the mean. Assuming that the threshold texture resolution is distributed normally in a population, we can model the cross-population probability of detection $P_{det}$ as a (negated) cumulative normal distribution:

$$P_{det}(r) = 1 - \Theta(r;\mu,\sigma) = 1 - \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{r-\mu}{\sigma\sqrt{2}}\right)\right] \qquad (1)$$

where $\Theta$ is the cumulative normal distribution, $r$ is the texture resolution in pixels, erf() is the Gauss error function, and $\mu$, $\sigma$ are the distribution parameters. The parameters can be found by computing the mean and standard deviation of per-observer thresholds.

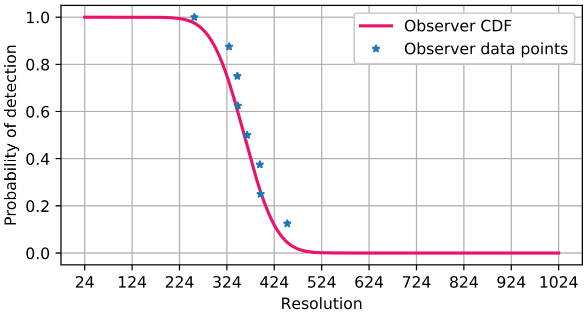


**Figure 5:** *Cross-population psychometric function explaining what portion of the population can see the difference between the reference image with full-resolution textures and a test image in which the texture resolution was reduced.*

## 5. Experiment 2: Local visibility maps

The goal of the second experiment was to collect a validation dataset with local visibility maps. We use this dataset to show the differences between both experimental protocols, but also to test

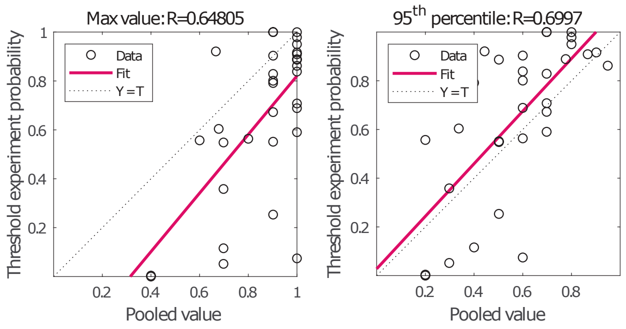


**Figure 6:** *The relation between the measured thresholds from Experiments 1 and 2. The scatter plots show the relationship between the maximum value (left) and 95$^{th}$ percentile (right) computed from the local visibility maps (Experiment 2) and the detection thresholds found in Experiment 1.*

our retuning procedure. We used the same experimental protocol as in [WGY*18], in which observers painted visible difference in images. Because the protocol is time-consuming, this experiment was feasible for only a subset of 32 from over 6477 images used Experiment 1. The monitor and viewing conditions were identical to those in Experiment 1.

**Experimental procedure** An observer was presented distorted and reference images side-by-side. She was then asked to carefully mark all the areas where artifacts are visible in the distorted image using a brush of adjustable size. The final visibility maps were obtained by averaging per-observer maps.

**Observers** 10 participants were recruited from the same demographic group as for Experiment 1. Most of the observers were new participants and only a few observers completed both experiments.

## 6. Pooling for task-specific visibility threshold

Our goal is to determine the texture resolution for which less than 20% of observers are going to notice any artifacts. One obvious approach is to run a visibility metric, such as CNN-GP or HDR-VDP-2, at every resolution level and use the predictions to determine the desired detection threshold. But, since each of these metrics predicts local visibility maps, where each pixel has associated probability of detection, we need to pool the values from the map into a single value that corresponds to the desired threshold.

For a conservative threshold, we could use the maximum value from the visibility map, as done in [AOS*17] and [WGY*18]. Such approach, however, suffers from two problems. Firstly, it assumes that every observer will always spot the most visible distortion. While this assumption is valid for a well-controlled marking experiment (Experiment 2), in which observers scrutinize every part of an image, it does not hold for the detection experiment (Experiment 1), or the target application of texture rendering. In less controlled conditions, different observers are going to spot different distortions, and only few will spot the one that is the most visible. This is illustrated in Figure 6 (left), where we plot the detection thresholds from Experiment 1, vs. the maximum value from the local visibility maps from Experiment 2. The plot on the right in the same figure shows that discarding the top 5% of the values (95th percentile) when pooling improves the relation between both datasets. Secondly, the maximum value is strongly affected by the prediction error. If we were to compute an average value across the visibility map, we could expect that a portion of the prediction error would cancel out. However, if we rely on a single maximum value, we cannot take advantage of averaging multiple predicted values.

To find a task-specific visibility threshold, we tested a number of pooling strategies on the predictions of three state-of-the-art visibility metrics: CNN-GP [WGY*18], HDR-VDP-2 [MKRH11] and Butteraugli [AOS*17]. The goal is to find a pooling that provides the lowest prediction error for the thresholds determined in Experiment 1. Firstly, we tested ten percentiles, from 90th to 99th. Secondly, we used a trainable pooling function realized by a neural network as shown in Figure 7. The network was composed of four convolutional layers interspersed with max-pooling layers and terminates with three fully connected layers, the last of which produces a scalar. To prevent over-fitting, we used 5-fold cross-validation with an 80:20 split between training and testing data.

An optimal pooling strategy is determined for each of the three metrics as follows. We run a given metric for each scene on all possible texture resolutions and consider all pooling strategies to find the probability of detection. To regularize the predictions across the resolution levels, we fit for each pooling strategy a psychometric function from Equation 1. Then, we compute a mean-absolute-error (MAE) between texture threshold resolution found in Experiment 1, and the texture resolution corresponding to the assumed 20% of probability of artifact detection derived from the psychometric function for this pooling option. We repeat this procedure for all scenes. The lowest prediction error was achieved for 93$^{rd}$ percentile for CNN-GP and Butteraugli, and 94$^{th}$ percentile for HDR-VDP-2. We were not able to train NN to provide higher correlation than these percentiles. Figure 8 gives an example of a single scene for which the psychometric functions were fitted for all the tested metrics. In Figure 8 we also present the maximum error value as a pooling option, which we denote as CNN-GP-Max. When comparing the match of CNN-GP-Max and CNN-GP-P93 to the experimental results, the benefits our pooling optimization are clearly visible.

Figure 9 compares optimal percentile pooling (continuous lines) to the results achieved with max-value pooling (dashed lines). The results are shown as an error histogram over all available scenes. The plots show that the trainable pooling also reduced the prediction error as compared to naïve max-value pooling, but provided worse results than the best percentile strategy.

## 7. Metric retuning

Although the percentile pooling much improves the prediction performance, it is too simple to account for complex changes in visibility thresholds in our application of texture rendering. We have also

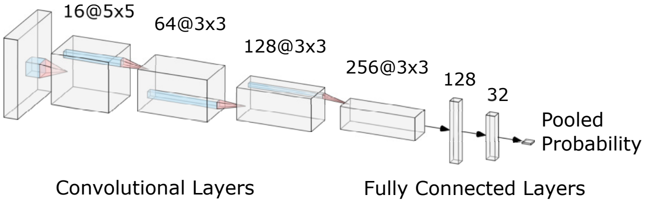


**Figure 7:** *The architecture of the CNN used for predicting the pooled threshold. Each convolution layer shows the number of kernels and the size of each one. The final fully connected layers show the number of neurons.*

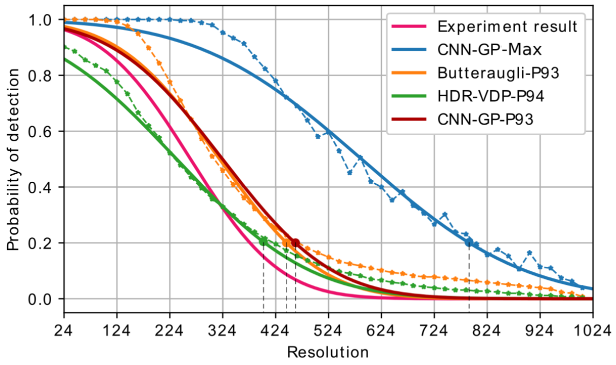


**Figure 8:** *Metric predictions (dashed lines, only for CNN-GP-Max, HDR-VDP-P94, and Butteraugli-P93) and psychometric functions fits (continuous lines) for a chosen scene. The desired resolution is selected for the probability of detection 0.2.*

shown that complex NN-based pooling generally does not provide better performance. To further improve the metric performance, we need to retune it to a new application. As the CNN-GP provides the most flexibility of retraining, we focus on this metric.

### 7.1. Retuning dataset

Before CNN-GP can be retuned for our texture optimization application, we need to generate a dataset consisting of pairs of reference and distorted images and the corresponding per-pixel visibility maps. Having such maps is vital for training a CNN-based metric, which operates on patches. Since our experiment does not provide per-pixel visibility information, we use instead existing visibility metrics, CNN-GP itself, HDR-VDP-2 and Butteraugli, to generate such maps. Each of these metrics was previously re-trained on the LocVis dataset as described in [WGY*18], which made them more accurate and robust to a broad set of artifact types. As the visibility maps predicted by existing metrics are not suitable for the target application because of the reasons outlined in Section 3, they need to be adjusted using the newly collected experimental data. The overview of the retuning procedure is illustrated in Figure 1.

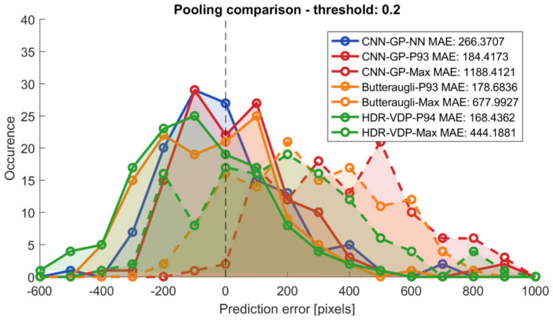


**Figure 9:** *Histograms of the error distribution for pooled probability value for scenes considered in Experiment 1. The error is expressed as signed texture resolution (both horizontal and vertical) differences in pixel units with respect to the experimental observer data. The percentile pooling (continuous lines) results in much lower maximum absolute error (MAE) values than naïve max value pooling (dashed lines). Trainable pooling strategy (NN Pooling) also performs reasonably well.*

First, we need to understand how visibility maps are produced by each visibility metric. Chosen visibility metrics involve several stages of spatial processing and point-wise nonlinearities, which culminate in transforming their semi-final output, a perceptually-normalized difference of contrast $C$, into probabilities using a sigmoidal psychometric function $\psi$:

$$\tilde{P}_{det}(x,y) = \psi(C(x,y)) \ , \tag{2}$$

where $x$ and $y$ are pixel coordinates. The per-image probability of detection is computed as the the optimal percentile value (Section 6) of $\tilde{P}_{det}$ over the image. CNN-GP uses a standard, non-parametrized sigmoid function:

$$\psi_{CNN}(c) = \frac{1}{1+e^{-c}} \tag{3}$$

In HDR-VDP-2 and Butteraugli the psychometric function has the form:

$$\psi(c) = 1 - e^{\ln(0.5)\, c^{\beta}} \ , \tag{4}$$

where $c$ is the image difference (in perceptual contrast space) and $\beta$ (typically 3.5) controls the slope of the psychometric function. Because the psychometric function $\psi$ saturates at large positive and negative values and introduces the loss of information after discretization, we need to adjust the values of difference contrast $C$ rather than probabilities $\tilde{P}_{det}$ to generate visibility maps consistent with our experiment. This means that for each series of images corresponding to the texture resolution of particular scene, we find a linear transformation of $C$ that, when transformed by Equation 2 and pooled using the $93^{rd}$ percentile, results in the detection threshold closest to the Experiment 1 results (in terms of MSE). The best linear transform is found using exhaustive search over the range of bias values. The adjustment is illustrated in Figure 10 in which the original metric predictions (dashed blue line) over-predicts the probability of detection. However, after adjusting $C$, the predictions (solid blue line) are close to the experimental data (pink line).

We separately generate three retuning datasets, each using a different visibility metric: CNN-GP, HDR-VDP-2, and Butteraugli.

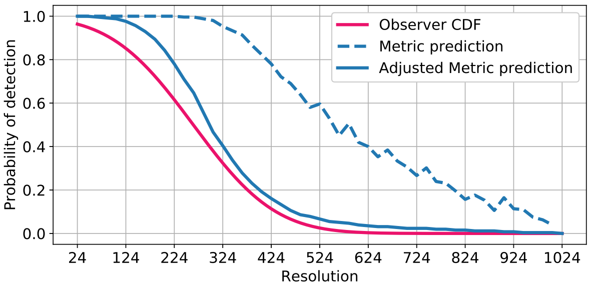


**Figure 10:** *Adjusted CNN-GP prediction (solid blue line) better matches observer cross-population psychometric function (pink line) than original CNN-GP prediction (dashed blue line).*

Each dataset consists of all images used in the experiment: 51 down-sampling factors, 33 3D models and all 127 variants. If the pooled value of the adjusted visibility maps differs by more than 0.05 from our experimental data, we remove that image from the dataset. This results in an extensive set of approximately 3,000 image pairs and per-pixel visibility maps, suitable for training a CNN-based metric.

### 7.2. Loss function

In [WGY*18], the authors proposed a probabilistic loss function, which was designed to account for inaccuracies of manual marking but also to conservatively predicts all visible distortions, even those that are more difficult to find. However, since our dataset is automatically generated by the metrics and accounting for the difficulty of finding artifacts is essential for our application, such a loss function is unsuitable.

To match probability predictions to the generated dataset, we have experimented with Kullback-Leibler KL-divergence, but that resulted in poor predictions. Instead, we use the loss that penalizes probability differences close to the range bounds:

$$L = \sqrt{\sum_{(x,y)\in\Theta} \left[g\left(p_{label}(x,y)\right) - g\left(p_{pred}(x,y)\right)\right]^2}, \tag{5}$$

where:

$$g(v) = \ln \frac{v+\varepsilon}{1-v+\varepsilon}, \tag{6}$$

$\Theta$ is the set of all pixels with coordinates $(x, y)$, $p_{label}$ is the target probability generated in the adjustment process, $p_{pred}$ is the metric prediction and $\varepsilon$ is a small constant.

### 7.3. Transfer learning and testing performance

The dataset is partitioned into 5 equal parts for 5-fold cross-validation. Each 3D object/texture is present in only one fold. The network is initialized with the weights used in the original CNN-GP metric and then retrained over 50000 iterations using the new dataset. We experimented with training only the final convolutional layers or the entire network and achieved very similar results. We report the results for the latter.

The CNN is trained on 48×48 pixel non-overlapping patches with a batch size of 128, in Tensorflow v.1.7.0 [§] with Python v3.6. We use Adam optimizer [**?**] with the learning rate of 0.00001 and set the dropout to 0.5.

The metric is intended for offline use. Processing of one 800×600 image takes about 3,5 seconds on Nvidia GeForce GTX 1070 and Intel Xeon E5-1650.

## 8. Results

First, we want to determine which of the existing visibility metrics produces the retuning dataset that gives the best predictions. We denote new metrics trained on the three retuning datasets as CNN-T-CNN-GP-P93, CNN-T-Butteraugli-P93, CNN-T-HDR-VDP-P94, where the label contains CNN-T (CNN reTunned), followed by the name of the existing metric used to generate the retuning dataset and the percentile pooling function. To evaluate the accuracy of retuned metrics we used the same approach as described in Section 6 and summarized in Figure 8. We run each metric on all possible texture resolutions and use the optimal percentile of the visibility map to find the probability of detection for each resolution. Then, we fit a psychometric function and compute a prediction error for threshold 0.2. Please refer to the supplemental material for the results computed using different threshold values.

The results of 5-fold cross-validation are shown as the histogram of errors in Figure 11. The legend includes the maximum-absolute-error (MAE), which we use instead of RMSE as it is more robust to outliers. The top plot shows that CNN-T-CNN-GP-P93 results in a lower prediction error than CNN-T-Butteraugli-P93 and CNN-T-HDR-VDP-P94. The distribution for the latter ones is also skewed towards predicting lower resolution.

Second, we show the improvement obtained by retuning the metrics as compared to using existing metrics with the percentile pooling. The middle plot of Figure 11 shows that the prediction error of CNN-GP was reduced from MAE=184 to MAE=115 after retuning, demonstrating that retuning is an effective mean of improving metric predictions. The plot also shows that CNN-GP retuned using a dataset generated with HDR-VDP-2 or Butteraugli performs better than the original metrics.

Finally, we compare our best performing metric (CNN-T-CNN-GP-P93) with the predictions of the existing visibility metrics: CNN-GP [WGY*18], and SSIM, which was found to be a good indicator of texture distortions [LLPP18]. We use T-SSIM version, which was augmented with the psychometric function in [WGY*18] to predict visibility rather than image quality. For a fair comparison, we also retrained T-SSIM on the same dataset as CNN-T-CNN-GP-P93. We denote such a retuned SSIM as SSIM-T-CNN-GP-P93. The bottom plot of Figure 11 shows very substantial improvement obtained by our retuning procedure, as compared to the existing visibility metrics. The predictions of retuned SSIM-T-CNN-GP-P93 are much improved as compared to T-SSIM, but still much worse than those of the retuned CNN-T-CNN-GP-P93.

[§] https://www.tensorflow.org/

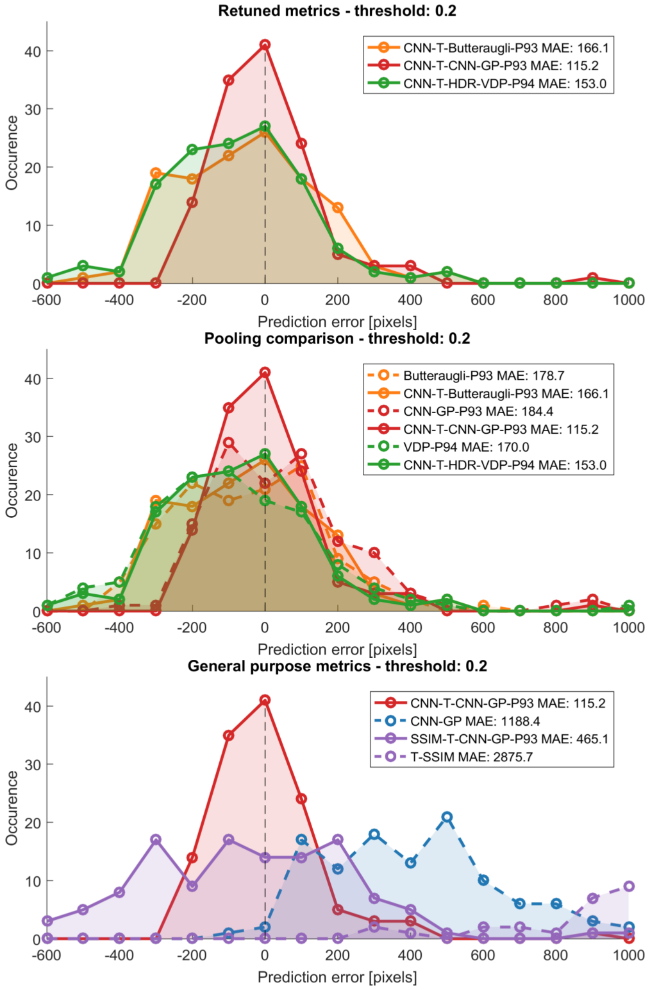


**Figure 11:** *The error distribution for the retuned and existing metrics. Top: comparison of three retuning datasets. Middle: the performance gain of retuning as compared to pooling. Bottom: comparison with the existing metrics. The existing visibility metrics, CNN-GP and T-SSIM, predict the threshold for higher than required texture resolution (bottom). The prediction error in histograms is expressed as signed texture resolution differences (both horizontal and vertical) in pixel units with respect to the experimental data. The value in the legend denotes mean-absolute-error (MAE).*

### 8.1. Visibility map improvement

Despite the fact that our target application requires only a single pooled value, we can also observe improvement in the visibility map predictions. To evaluate visibility maps predictions, we compare them to 32 manually marked visibility maps from Experiment 2. Table 1 shows that the retrained metrics improve predictions over the vanilla CNN-GP metric for the probabilistic loss from [WGY*18]. We compared maps using the following indicators: Pearson correlation coefficient (r), Spearman correlation coefficient (ρ), Root Mean Squared Error (RMSE), and the loss function described in Section 7.2 (Loss). The best performing CNN-T-CNN-GP-P93 improves the predictions for all performance measures. For visual examples of improvement please refer to the supplemental material.

| Image Metric | r | ρ | RMSE | Loss |
|---|---|---|---|---|
| CNN-GP | 0.706 | 0.789 | 0.214 | 0.311 |
| CNN-T-CNN-GP-P93 | 0.792 | 0.832 | 0.179 | 0.272 |
| CNN-T-Butteraugli-P93 | 0.663 | 0.769 | 0.2 | 0.285 |
| CNN-T-HDR-VDP-P94 | 0.715 | 0.791 | 0.167 | 0.248 |

**Table 1:** *All indicators show that retuned metrics provide visibility with higher accuracy than general-purpose metric CNN-GP.*

## 9. Texture size optimization

To test performance in reducing the texture resolution without introducing visible artifacts, we generated a new set of 3D models. Four of them: *TreasureBox*, *Plant*, *DemonDoor*, and *AngelStatue* are presented in Figure 12. Initially, all of them are textured with albedo and normal maps at the original texture resolution of 1024×1024. We placed the models in different environments, which provide realistic lighting.

To determine the minimum texture resolutions, we first sample viewing and illumination directions from those that can be encountered in a game scenario. In Figure 13, we show 5 out of 8 such views (all 8 used views are shown in the supplemental material). Next, each view is rendered using all possible combinations of normal and albedo map resolutions. Here, we restrict our choice to resolutions equal to the powers-of-two, which are more relevant for the game-engine application. All scene renderings are then compared to the reference rendering with full-resolution textures using the proposed visibility metric (CNN-T-CNN-GP-P93). The outcome of this prediction is summarized in the tables shown in Figure 13, which contain the probability of spotting artifacts for all combinations of normal and albedo map resolutions. Given these matrices, we can choose the combination which results in the smallest memory footprint and at the same time meets our visibility criterion. One possible example of such a criterion could be that at most 10% of the sampled views have the probability of detection greater than 0.2. Table 2 summarizes the optimal texture size selection and the resulting reduction in memory footprint. It shows that the optimal texture resolution varies substantially with the model, making it hard to manually choose the right texture configuration. The proposed metric can greatly facilitate this task.

Figure 12 shows the final rendering of two scenes using optimized texture size in comparison to reference textures and suboptimal texture size choice. In *Plant* scene leaf veins are the most important fine details. The image rendered with optimal texture size preserves these details, while for the rendering with the sub-optimal texture resolution, they tend to get blurred. In the second example, *TreasureBox*, the emblem on the chest's lid gets darker due to downsampling of albedo texture. The artifact is not present in the optimal resolution image. In *AngelStatue* scene, high-frequency stone is present on the surface. For optimal texture resolution, the

| Scene | Albedo [px] | | Normal [px] | | Memory [MB] | |
|---|---|---|---|---|---|---|
| | before | after | before | after | before | after |
| TreasureBox | 1024 | 512 | 1024 | 512 | 8 | 2 |
| Plant | 1024 | 512 | 1024 | 128 | 8 | 1.1 |
| DeamonDoor | 1024 | 256 | 1024 | 128 | 8 | 0.3 |
| AngelStatue | 1024 | 512 | 1024 | 512 | 8 | 2 |
| King | 1024 | 1024 | 1024 | 1024 | 8 | 8 |

**Table 2:** *The texture size reductions guided by the proposed metric and the corresponding memory savings. The resolution reported for the scenes shown in Figure 13.*

pattern remains preserved. It is not true for the sub-optimal texture resolution. In the last scene *DeamonDoor*, we can observe cracks and scratches on the horns. Once again, the details are correctly preserved for optimized textures while they disappear when the sub-optimal textures are used.

The results in Figure 13 demonstrate a general trend where normal maps can be reduced to lower resolutions than albedo maps. We can also notice an interaction between the textures. For example for the *Plant* model the lower resolution of the albedo map results in strong artifact visibility with a relatively weak impact of the normal texture resolution. In *TreasureBox* for most of the camera positions both the albedo and normal texture resolutions are important and only moderate resolution reduction is possible (refer also to Table 2). In the *King* scene the mesh contains spikes and horns sticking out of the body, which causes the object surface to be bigger compared to other scenes. Because of that, the texel density is lower and texture resolution can not be reduced without introducing visible artifacts.

## 10. Limitations

As any learning-based method, the proposed visibility metric is restricted to the examples that are close to the training dataset. For example, our dataset considers only albedo and normal maps, while 3D assets may also contain specularity, transparency, anisotropy and other types of texture maps, for which our metric can be less accurate. Unlike white-box visibility metrics, such as HDR-VDP, our metric does not account for the display parameters (peak brightness, contrast) and viewing distance. We would like to address these limitations in future work.

## 11. Conclusions

Predicting the visibility of artifacts in rendering is a challenging task. Existing state-of-the-art metrics are often trained for a specific task, such as detecting side-by-side difference, which may not be applicable in many scenarios, and therefore, lead to over-conservative predictions. We demonstrate how a CNN-based metric can be easily retuned for a specific task using a dataset collected with limited effort. Such retuning leads to a greatly improved prediction performance, which let us use our new metric in a practical application of optimizing the texture resolution of 3D assets.

## 12. Acknowlegments

The project was supported by the Fraunhofer and Max Planck cooperation program within the German pact for research and innovation (PFI). This project has also received funding from the European Union's Horizon 2020 research and innovation programme, under the Marie Skłodowska-Curie grant agreements N° 642841 (DISTRO), N° 765911 (RealVision), and from the European Research Council (ERC) grant agreements N° 725253 (EyeCode), and N° 804226 (PERDY).

## References

[AOS*17] ALAKUIJALA J., OBRYK R., STOLIARCHUK O., SZABADKA Z., VANDEVENNE L., WASSENBERG J.: Guetzli: Perceptually guided JPEG encoder. *arXiv:1703.04421* (2017). 2, 6

[AVS*17] ADHIKARLA V. K., VINKLER M., SUMIN D., MANTIUK R. K., MYSZKOWSKI K., SEIDEL H.-P., DIDYK P.: Towards a quality metric for dense light fields. In *Computer Vision and Pattern Recognition* (2017). 2

[BMM*18] BOSSE S., MANIRY D., MÜLLER K. R., WIEGAND T., SAMEK W.: Deep neural networks for no-reference and full-reference image quality assessment. *IEEE Transactions on Image Processing 27*, 1 (2018), 206–219. 2

[Car18] CARRIER E.: Procedural world generation of 'far cry 5', 2018. Game Devlopers Conference. 1

[Cha13] CHANDLER D. M.: Seven challenges in image quality assessment: Past, present, and future research. *ISRN Signal Processing* (2013), Article ID 905685. doi:10.1155/2013/905685. 2

[ČHM*12] ČADÍK M., HERZOG R., MANTIUK R., MYSZKOWSKI K., SEIDEL H.-P.: New Measurements Reveal Weaknesses of Image Quality Metrics in Evaluating Graphics Artifacts. *ACM SIGGRAPH Asia 2012, Transactions on Graphics (TOG) 31*, 6 (2012), pp. 147–157. 2

[COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-preserving simplification. In *SIGGRAPH '98* (1998), pp. 115–122. 2

[Dal93] DALY S.: The Visible Differences Predictor: An algorithm for the assessment of image fidelity. In *Digital Images and Human Vision* (1993), MIT Press, pp. 179–206. 2

[DOS16] DEGUY S., OLGUIN R., SMITH B.: Texturing uncharted 4: a matter of substance, 2016. Game Devlopers Conference. 1

[Epi18] EPIC GAMES: Optimizing ue4 for fortnite: Battle royale, 2018. Game Devlopers Conference. 1, 2

[FSPG97] FERWERDA J. A., SHIRLEY P., PATTANAIK S. N., GREENBERG D. P.: A model of visual masking for computer graphics. In *SIGGRAPH '97* (1997), pp. 143–152. 2

[GO15] GRIFFIN W., OLANO M.: Evaluating texture compression masking effects using objective image quality assessment metrics. *IEEE Transactions on Visualization and Computer Graphics 21*, 8 (2015), 970–979. 1, 2, 3

[GVC*16] GUO J., VIDAL V., CHENG I., BASU A., BASKURT A., LAVOUÉ G.: Subjective and objective visual quality assessment of textured 3d meshes. *ACM Trans. Appl. Percept. 14*, 2 (2016), 11:1–11:20. 2

[HER00] HOLLY E. RUSHMEIER BERNICE E. ROGOWITZ C. P.: Perceptual issues in substituting texture for geometry. In *Proc.SPIE* (2000), vol. 3959, pp. 3959 – 3959 – 12. 2

[Kar13] KARIS B.: *Real Shading in Unreal Engine 4*. Tech. rep., Epic Games, 2013. URL: http://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf. 4

[KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in*

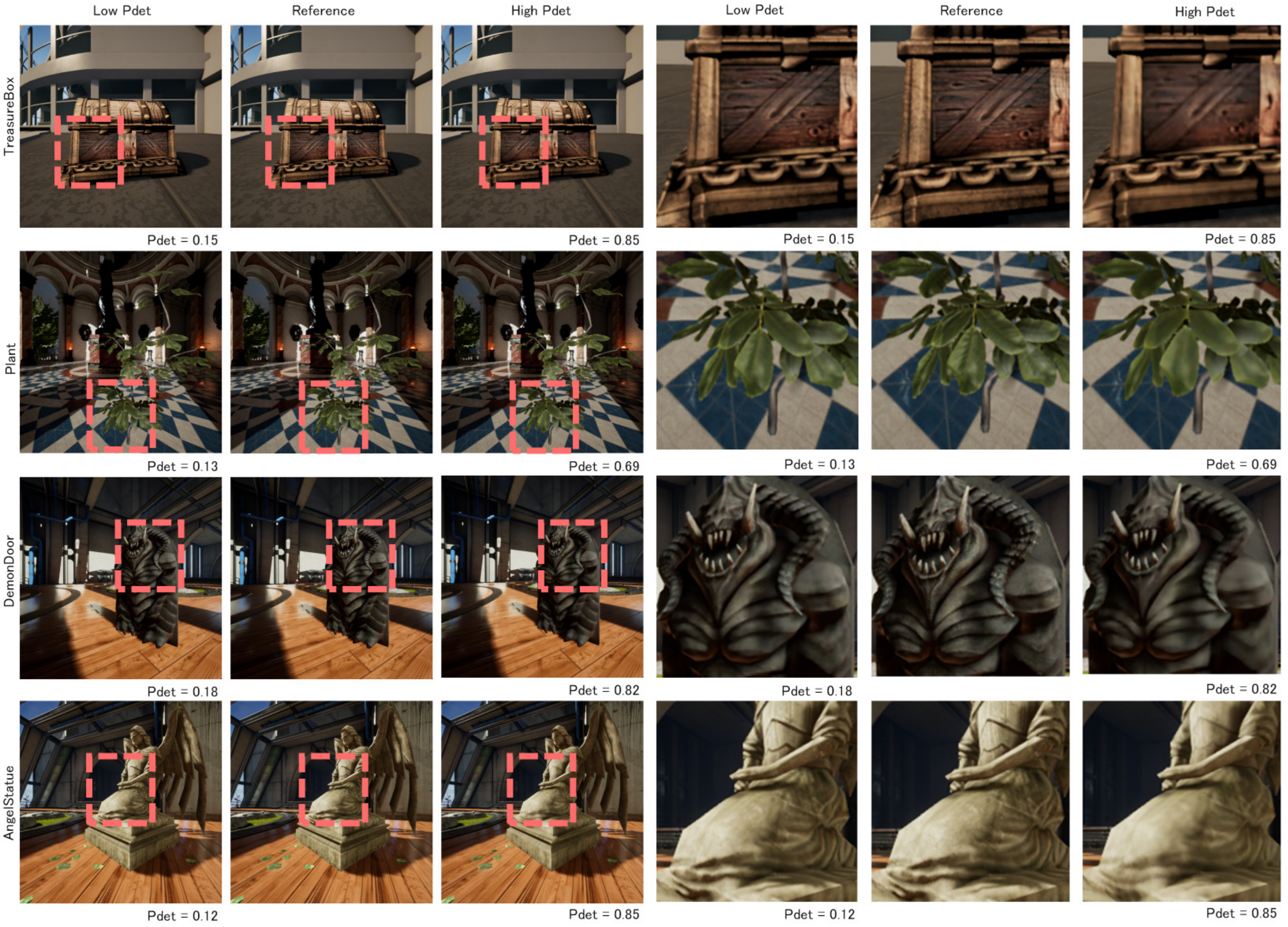


**Figure 12:** *Examples of texture size optimization. From left in each row: rendering with a low probability of detecting the artifacts, rendering with full texture resolutions, and with a high probability of detection. The following three images in the row detail respectively the regions in the boxes. For proper viewing conditions at the 60 cm viewing distance, the images and the insets should span 9 cm and 3 cm on the screen, respectively.*

*Neural Information Processing Systems 25*, Pereira F., Burges C. J. C., Bottou L., Weinberger K. Q., (Eds.). Curran Associates, Inc., 2012, pp. 1097–1105. 2

[LLPP18] Lavoué G., Langer M., Peytavie A., Poulin P.: A psychophysical evaluation of texture compression masking effects. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1. 1, 2, 3, 4, 8

[LM15] Lavoué G., Mantiuk R.: Quality assessment in computer graphics. In *Visual Signal Quality Assessment: Quality of Experience (QoE)* (2015), Deng C., Ma L., Lin W., Ngan K. N., (Eds.), Springer International Publishing, pp. 243–286. 2

[LO11] Larkin M., O'Sullivan C.: Perception of simplification artifacts for animated characters. In *ACM SIGGRAPH Symposium on Applied Perception in Graphics and Visualization* (2011), pp. 93–100. 3

[LT00] Lindstrom P., Turk G.: Image-driven simplification. *ACM Trans. Graph. 19*, 3 (2000), 204–241. 2

[Lub95] Lubin J.: A visual discrimination model for imaging system design and evaluation. *Vision models for target detection and recognition 2* (1995), pp. 245–357. 2

[MG10] Menzel N., Guthe M.: Towards perceptual simplification of models with arbitrary materials. *Computer Graphics Forum 29*, 7 (2010), 2261–2270. 2

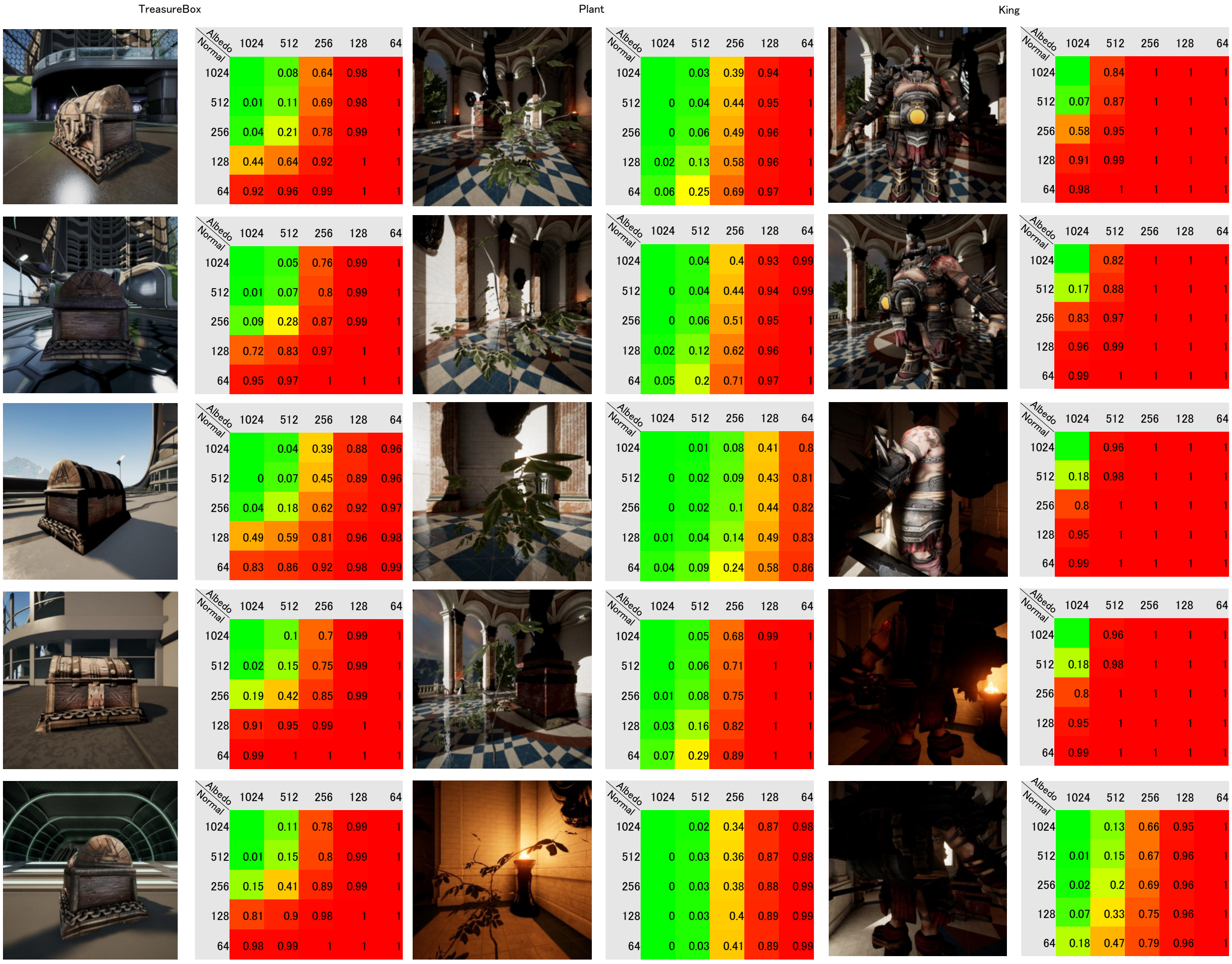
[MKRH11] Mantiuk R., Kim K. J., Rempel A. G., Heidrich W.: HDR-VDP-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 40. 2, 3, 6

[OHM*04] O'Sullivan C., Howlett S., Morvan Y., McDonnell R., O'Conor K.: Perceptually Adaptive Graphics. In *STAR-Proceedings of Eurographics 2004* (2004), State of the Art Reports, Eurographics Association, pp. 141–164. 2

[PCB05] Pan Y., Cheng I., Basu A.: Quality metric for approximating subjective evaluation of 3-d objects. *IEEE Transactions on Multimedia 7*, 2 (2005), 269–279. 2

[QM08] Qu L., Meyer G. W.: Perceptually guided polygon reduction. *IEEE Transactions on Visualization and Computer Graphics 14*, 5 (2008), 1015–1029. 2

[SSB06] Sheikh H., Sabir M., Bovik A.: A statistical evaluation of



TreasureBox

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.08 | 0.64 | 0.98 | 1 |
| 512 | 0.01 | 0.11 | 0.69 | 0.98 | 1 |
| 256 | 0.04 | 0.21 | 0.78 | 0.99 | 1 |
| 128 | 0.44 | 0.64 | 0.92 | 1 | 1 |
| 64 | 0.92 | 0.96 | 0.99 | 1 | 1 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.05 | 0.76 | 0.99 | 1 |
| 512 | 0.01 | 0.07 | 0.8 | 0.99 | 1 |
| 256 | 0.09 | 0.28 | 0.87 | 0.99 | 1 |
| 128 | 0.72 | 0.83 | 0.97 | 1 | 1 |
| 64 | 0.95 | 0.97 | 1 | 1 | 1 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.04 | 0.39 | 0.88 | 0.96 |
| 512 | 0 | 0.07 | 0.45 | 0.89 | 0.96 |
| 256 | 0.04 | 0.18 | 0.62 | 0.92 | 0.97 |
| 128 | 0.49 | 0.59 | 0.81 | 0.96 | 0.98 |
| 64 | 0.83 | 0.86 | 0.92 | 0.98 | 0.99 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.1 | 0.7 | 0.99 | 1 |
| 512 | 0.02 | 0.15 | 0.75 | 0.99 | 1 |
| 256 | 0.19 | 0.42 | 0.85 | 0.99 | 1 |
| 128 | 0.91 | 0.95 | 0.99 | 1 | 1 |
| 64 | 0.99 | 1 | 1 | 1 | 1 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.11 | 0.78 | 0.99 | 1 |
| 512 | 0.01 | 0.15 | 0.8 | 0.99 | 1 |
| 256 | 0.15 | 0.41 | 0.89 | 0.99 | 1 |
| 128 | 0.81 | 0.9 | 0.98 | 1 | 1 |
| 64 | 0.98 | 0.99 | 1 | 1 | 1 |

Plant

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.03 | 0.39 | 0.94 | 1 |
| 512 | 0 | 0.04 | 0.44 | 0.95 | 1 |
| 256 | 0 | 0.06 | 0.49 | 0.96 | 1 |
| 128 | 0.02 | 0.13 | 0.58 | 0.96 | 1 |
| 64 | 0.06 | 0.25 | 0.69 | 0.97 | 1 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.04 | 0.4 | 0.93 | 0.99 |
| 512 | 0 | 0.04 | 0.44 | 0.94 | 0.99 |
| 256 | 0 | 0.06 | 0.51 | 0.95 | 1 |
| 128 | 0.02 | 0.12 | 0.62 | 0.96 | 1 |
| 64 | 0.05 | 0.2 | 0.71 | 0.97 | 1 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.01 | 0.08 | 0.41 | 0.8 |
| 512 | 0 | 0.02 | 0.09 | 0.43 | 0.81 |
| 256 | 0 | 0.02 | 0.1 | 0.44 | 0.82 |
| 128 | 0.01 | 0.04 | 0.14 | 0.49 | 0.83 |
| 64 | 0.04 | 0.09 | 0.24 | 0.58 | 0.86 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.05 | 0.68 | 0.99 | 1 |
| 512 | 0 | 0.06 | 0.71 | 1 | 1 |
| 256 | 0.01 | 0.08 | 0.75 | 1 | 1 |
| 128 | 0.03 | 0.16 | 0.82 | 1 | 1 |
| 64 | 0.07 | 0.29 | 0.89 | 1 | 1 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.02 | 0.34 | 0.87 | 0.98 |
| 512 | 0 | 0.03 | 0.36 | 0.87 | 0.98 |
| 256 | 0 | 0.03 | 0.38 | 0.88 | 0.99 |
| 128 | 0 | 0.03 | 0.4 | 0.89 | 0.99 |
| 64 | 0 | 0.03 | 0.41 | 0.89 | 0.99 |

King

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.84 | 1 | 1 | 1 |
| 512 | 0.07 | 0.87 | 1 | 1 | 1 |
| 256 | 0.58 | 0.95 | 1 | 1 | 1 |
| 128 | 0.91 | 0.99 | 1 | 1 | 1 |
| 64 | 0.98 | 1 | 1 | 1 | 1 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.82 | 1 | 1 | 1 |
| 512 | 0.17 | 0.88 | 1 | 1 | 1 |
| 256 | 0.83 | 0.97 | 1 | 1 | 1 |
| 128 | 0.96 | 0.99 | 1 | 1 | 1 |
| 64 | 0.99 | 1 | 1 | 1 | 1 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.96 | 1 | 1 | 1 |
| 512 | 0.18 | 0.98 | 1 | 1 | 1 |
| 256 | 0.8 | 1 | 1 | 1 | 1 |
| 128 | 0.95 | 1 | 1 | 1 | 1 |
| 64 | 0.99 | 1 | 1 | 1 | 1 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.96 | 1 | 1 | 1 |
| 512 | 0.18 | 0.98 | 1 | 1 | 1 |
| 256 | 0.8 | 1 | 1 | 1 | 1 |
| 128 | 0.95 | 1 | 1 | 1 | 1 |
| 64 | 0.99 | 1 | 1 | 1 | 1 |

| Albedo<br>Normal | 1024 | 512 | 256 | 128 | 64 |
|---|---|---|---|---|---|
| 1024 | | 0.13 | 0.66 | 0.95 | 1 |
| 512 | 0.01 | 0.15 | 0.67 | 0.96 | 1 |
| 256 | 0.02 | 0.2 | 0.69 | 0.96 | 1 |
| 128 | 0.07 | 0.33 | 0.75 | 0.96 | 1 |
| 64 | 0.18 | 0.47 | 0.79 | 0.96 | 1 |

**Figure 13:** *Examples of 5 views rendered using reference texture (rows) and the corresponding metric predictions for all combinations of normal and albedo texture resolutions. The columns in the tables represent variation in the albedo map resolution, and the rows the variation in the normal map resolution. The color coding represents the probability of detection, from 0 (green, invisible) to 1 (red, definitely visible).*

recent full reference image quality assessment algorithms. *IEEE Trans. on Image Processing 15*, 11 (2006), 3440–3451. 2

[TA04] TIAN D., ALREGIB G.: Fqm: A fast quality measure for efficient transmission of textured 3d models. In *ACM MULTIMEDIA '04* (2004), pp. 684–691. 2

[VRH*18] VANCLEEF K., READ J. C., HERBERT W., GOODSHIP N., WOODHOUSE M., SERRANO-PEDRAZA I.: Two choices good, four choices better: For measuring stereoacuity in children, a four-alternative forced-choice paradigm is more efficient than two. *PloS one 13*, 7 (2018), e0201366. 4

[WB06] WANG Z., BOVIK A. C.: *Modern Image Quality Assessment.* Morgan & Claypool Publishers, 2006. 2

[WGY*18] WOLSKI K., GIUNCHI D., YE N., DIDYK P., MYSZKOWSKI K., MANTIUK R., SEIDEL H.-P., ANTHONY S., MANTIUK R. K.: Dataset and metrics for predicting local visible differences. *ACM Transactions on Graphics (ToG)* (2018). 2, 3, 6, 7, 8, 9

[WLC*03] WILLIAMS N., LUEBKE D., COHEN J. D., KELLEY M., SCHUBERT B.: Perceptually guided simplification of lit textured meshes. In *In Proceedings of the 2003 symposium on Interactive 3D graphics* (2003), ACM Press, pp. 113–121. 2

[WP83] WATSON A. B., PELLI D. G.: Quest: A bayesian adaptive psychometric method. *Perception & Psychophysics 33*, 2 (1983), 113–120. 4

[WPG02] WALTER B., PATTANAIK S. N., GREENBERG D. P.: Using perceptual texture masking for efficient image synthesis. *Computer Graphics Forum 21*, 3 (2002). 2

[WSR*17] WEIER M., STENGEL M., ROTH T., DIDYK P., EISEMANN E., EISEMANN M., GROGORICK S., HINKENJANN A., KRUJIFF E., MAGNOR M. A., MYSZKOWSKI K., SLUSALLEK P.: Perception-driven accelerated rendering. *Computer Graphics Forum 36*, 2 (2017), 611–643. 2

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep networks as a perceptual metric. In *Computer Vision and Pattern Recognition* (2018). 2