Interactive Photo Editing on Smartphones via Intrinsic Decomposition

Sumit Shekhar¹, Max Reimann¹, Maximilian Mayer¹,², Amir Semmo¹,², Sebastian Pasewaldt¹,², Jürgen Döllner¹, and Matthias Trapp¹

¹Hasso Plattner Institute for Digital Engineering, University of Potsdam, Germany
²Digital Masterpieces GmbH, Germany

Figure 1: Different types of effects produced with our mobile app. It is the first that supports a large variation of image manipulation tasks within a unified framework, which is based on intrinsic image decomposition.

Abstract
Intrinsic decomposition refers to the problem of estimating scene characteristics, such as albedo and shading, when one view or multiple views of a scene are provided. The inverse problem setting, where multiple unknowns are solved given a single known pixel-value, is highly under-constrained. When provided with correlating image and depth data, intrinsic scene decomposition can be facilitated using depth-based priors, which nowadays is easy to acquire with high-end smartphones by utilizing their depth sensors. In this work, we present a system for intrinsic decomposition of RGB-D images on smartphones and the algorithmic as well as design choices therein. Unlike state-of-the-art methods that assume only diffuse reflectance, we consider both diffuse and specular pixels. For this purpose, we present a novel specularity extraction algorithm based on a multi-scale intensity decomposition and chroma inpainting. At this, the diffuse component is further decomposed into albedo and shading components. We use an inertial proximal algorithm for non-convex optimization (iPiano) to ensure albedo sparsity. Our GPU-based visual processing is implemented on iOS via the Metal API and enables interactive performance on an iPhone 11 Pro. Further, a qualitative evaluation shows that we are able to obtain high-quality outputs. Furthermore, our proposed approach for specularity removal outperforms state-of-the-art approaches for real-world images, while our albedo and shading layer decomposition is faster than the prior work at a comparable output quality. Manifold applications such as recoloring, retexturing, relighting, appearance editing, and stylization are shown, each using the intrinsic layers obtained with our method and/or the corresponding depth data.

CCS Concepts
• Computing methodologies .... Image-based rendering; Image processing; Computational photography;

1. Introduction
On a bright sunny day, it is quite easy for us to identify objects like a wall, a car, or a bike irrespective of their color, material or whether they are partially shaded. This remarkable capacity of human visual system (HVS) to disentangle visual ambiguities due to color, material, shape, and lighting is a result of many years of evolution [BBS14]. Replicating this ability for machine vision—to enable better scene understanding—has been a widely researched topic, but ever has been challenging because of its ill-posed and under-constrained nature.
The physical formation of an image involves various unknowns at macroscopic and microscopic levels, and decomposing them altogether makes it ill-posed. A more relaxed approximation is given by the Dichromatic Reflection Model where an image (I) is assumed to be composed of the sum of specular (I_S) and diffuse (I_d) components (at every pixel location x) [Sha85]:

\[ I(x) = I_d(x) + I_s(x). \] (1)

The diffuse component (I_d) can be further expressed as the product of albedo (A) and shading (S) [BT78]:

\[ I_d(x) = A(x) \cdot S(x). \] (2)

However, even this approximation is under-constrained, because three unknowns—A(x), S(x) and I_s(x)—need to be solved given only the image color I(x). In this work, we propose a novel smartphone-based system to extract intrinsic layers of albedo, shading, and specularities. In our system, the specularity removal is carried out as a pre-processing step followed by a depth-based energy minimization for computing the other two layers. The computed layers, apart from offering better scene understanding, facilitate a range of image-editing applications such as recoloring, retexturing, relighting, appearance editing etc. (Fig. 1).

Compared to many previous works, ours is not limited in assuming a complete diffuse reflection. In general, the decomposition of an image into diffuse reflectance (albedo) and shading is referred to as Intrinsic Image Decomposition (IID). The existing IID algorithms can be broadly classified into two categories:

**Learning-based methods:** the priors on albedo and shading are incorporated as loss functions, and the decomposition is learned by training. In the past few years—with the significant improvement in deep-learning technology—such methods have become quite popular [ZKE15, KPSL16, CZL18, LVv18]. However, capturing real-world training data for IID is challenging and existing datasets might not be sufficient [GJAF09, BKK15, BHK16, SBZ18]. Unsupervised learning does not require any training data, however, the results are generally of inferior quality [LVVG18, MCZ18, LS18]. Most learning-based models have high GPU memory consumption, making them potentially unsuitable for mobile devices—especially at those image resolutions that an image-editing application typically requires. Furthermore, these models are generally not controllable at run-time, i.e., the decomposition cannot be fine-tuned to the image at hand, which is a significant limitation for interactive editing applications.

**Optimization-based methods:** a cost function based on priors is minimized to find an approximate solution. Initial techniques use simplistic priors, which are not suitable for real-world scenes [TFA05]. More complex priors improve the accuracy at the cost of associated computational complexity [ZTD12, BBS14, BM15, WLYY17]. Readily available depth sensors fostered depth-based methods for IID [CK13, JCTL14]. Nowadays, with easily available mobile devices with depth sensors, a depth-based intrinsic image decomposition method can be a preferred choice for an intrinsic-image application in mobile environments.

As an additional constraint, only a few previous methods perform both IID and specularity extraction together. Innamorati et al. [RWM17] and Shi et al. [SDSY17] employ a learning-based technique: both of them train and test for single objects but do not consider a realistic scene with many objects. The algorithm by Alperovich et al. [AG16] is designed for light-fields but cannot be used for a single image. The method of Beigpour et al. [BSM18] is applicable for a single image and, like ours, removes specularities in a pre-processing step. However, for specularity extraction, they do not consider chroma channels leading to artifacts in highly saturated image regions. Moreover, their method is an order of magnitude slower than ours. Unlike most of the previous standalone specularity removal techniques, we showcase our results based on a broad range of realistic images [ABC11]. Because we treat high- and low-frequency specularities differently, we obtain seamless outputs.

Finally, the processing schemes of many state-of-the-art techniques are comparably slow (optimization-based and learning-based), resource intensive and are limited to low image resolutions (learning-based). Thus, using an intrinsic decomposition for interactive image editing on mobile devices is considered challenging. We propose a system that provides a more practical approach to intrinsic decomposition. Specifically, we address the following design objectives:

**Accessibility:** a decomposition is provided on readily available mobile devices with depth sensors.

**Speed:** all post-capture processing takes at most a few seconds (on the mobile device) before the edited photo can be viewed, even when the device is offline. Thus, we cannot delegate processing to a desktop computer or the cloud.

**Interaction:** interacting with the decomposition and editing pipeline is possible in real-time, and the navigation affordances are fairly obvious.

**Quality:** the rendered application outputs look (i) plausible with respect to appearance editing and (ii) aesthetically pleasing for image-stylization tasks.

To this end, we split our processing pipeline into pre-processing and image-editing stages, of which the specular removal and image editing perform at interactive frame rates. Thereby, we provide the first mobile app that performs intrinsic decomposition in a unified framework and supports a large variation of image editing tasks (Fig. 1). This is technically achieved by utilizing the built-in depth sensor and dedicated GPU of modern smartphones for real-time capturing and interactive processing of RGB-D data.

Our contributions are summarized as follows, we propose:

1. A novel, interactive specularity removal method that treats high-frequency and low-frequency specularities differently, performs chroma-inpainting to address the problem of missing or little chromaticity information for saturated pixels, and that is well-suited for real-world images.
2. A fast and robust system for intrinsic decomposition of RGB-D images on smartphones that makes use of depth-data for local shading smoothness and enforce albedo (I_d)-sparsity by employing the efficient iPiano optimization solver [OCBP14].
3. A variety of mobile-based applications—to show the ubiquitous accessibility, speed, and quality of our method—using the given depth data and/or computed intrinsic layers of albedo, shading, and specularity.
2. Related Work

2.1. Specularity Removal

Some of the earliest methods for specularity removal were based on color segmentation, thus they were not robust against textures [KSK88, BLI96]. Mallik et al. [MZBK06] introduce a partial differential equation (PDE) in the SUV color space that iteratively erodes the specular component. A class of algorithms use the concept of specular-free image based on chromaticity values [TB05, SC09]. Yang et al. [YWA10] use a similar approach, and achieve real-time performance by employing parallel processing. Kim et al. [KHK13] use a dark channel prior to obtain specular-free images, followed by an optimization framework. Guo et al. [GZW18] propose a sparse low-rank reflection model and use a $L_1$ norm constraint in their optimization to filter specularities. A broad survey of specularity removal methods is provided by Ar- tusi et al. [ABC11]. Recently, Li et al. [LLZI17] utilize both image and depth data for removing specularity from human facial images. Most of these methods, however, employ specific object(s) or scene settings to evaluate their methods and do not consider generic real-world images. A recent method by Fu et al. [FZS*19] aims to address this issue; the authors assume that specularity is generally sparse and the diffuse component can be expressed as a linear combination of basis colors. They present a wide range of results, however, the optimization solving is comparably slow and is limited to low-resolution images. By contrast, our method is aimed for generic real-world high-resolution images with interactive performance on mobile devices.

2.2. Intrinsic Image Decomposition

The term intrinsic decomposition was introduced in the literature by Barrow and Tenenbaum [BT78]. The Retinex theory by Land and McCann proved to be a crucial finding, which became part of many following algorithms as a prior [LM71]. In the course of previous decades, intrinsic decomposition algorithms have been proposed for image [TFA05, BBS14, BM15, ZTD*17], video [YGL*14, BST*14, MZRT16], multiple-views [LBD13, DRC*15, MQD*17] and light-fields [GEZ*17, AG16, AJSIG18, BSM*18]. A survey covering many of these algorithms is provided by Bonneel et al. [BKPB17]. A particular class of algorithms use depth as additional information for IID. Lee et al. [LZT*12] use normals to impose constraints on shading and also use temporal constraints to obtain smooth results. Chen and Kolmam [CK13] further decompose shading into direct and indirect irradiance; the authors use depth to construct position-normal vectors for regularizing them. Hachama et al. [HGW15] use a single image or multiple RGB-D images to construct a point cloud. The normal vectors along with low dimensional global lighting model is used to jointly estimate lighting and albedo. Similarly, we use depth information to impose local shading smoothness constraints. However, unlike previous methods, a pre-processing step of specularity removal makes our method robust against specular image pixels. Moreover, we employ an efficient iPiano optimization solver [OCBP14] for our fast and robust mobile-based solution.

3. Method

A pre-processing step removes the specular highlights from the input image (Sec. 3.1), the diffuse component is further decomposed into albedo and shading layers using an efficient intrinsic decomposition optimization (Sec. 3.2). The resulting intrinsic layers are used to showcase various image editing applications (Sec. 5). A flowchart of our full pipeline is depicted in Fig. 2.

3.1. Specularity Removal Filtering

It has been shown that the perception of lightness and gloss is related to image statistics and can be altered by modifying the skewness of sub-bands of luminance histogram [SLM*08]. Our specularity removal step is motivated from the above observation. Further, in order to make our method robust against color artifacts we use image intensity $L$ instead of luminance for the above [BSM*18]. The chromaticity $C$ of the input image $I$ (with color channels $R$, $G$, and $B$) is processed separately to handle missing color information for saturated specular pixels.

$$L = \sqrt{R^2 + G^2 + B^2}, \quad C = \frac{I}{L} \quad (3)$$

A flowchart for our specularity removal algorithm is depicted in Fig. 3, the method broadly consists of three major steps as the following.
3.1.1. Identification of Specularity

In general, specular reflection increases the intensity of output spectrum and, furthermore, makes it more uniform. Both of these factors are efficiently captured by the unnormalized Wiener entropy \((H)\) introduced by Tian and Clark [TC13]. It can concisely be expressed as the product of input-image color channels \(R, G,\) and \(B\) (refer to Eqs. 1 - 6 in [TC13] for a detailed derivation):

\[
H(I) = R \cdot G \cdot B. \tag{4}
\]

The proposed unnormalized Wiener (UW) entropy encapsulates the color-direction-changing and intensity-increasing aspect of specularities. We can describe a specularity as a region where \(H\) of the total-reflection is significantly higher than the corresponding diffuse-reflection.

\[
\begin{align*}
H(Tot(\lambda)) - H(Dif(\lambda)) &> \tau' \\
H(Tot(\lambda)) &> \tau' + H(Dif(\lambda))
\end{align*} \tag{5}
\]

where \(Tot(\lambda)\) is the spectrum of the total reflection, \(Dif(\lambda)\) is the spectrum of the diffuse component and \(\tau'\) is a particular threshold.

The UW entropy for the diffuse component is assumed to have little variation within the scene and is considered a constant. Thus, a single universal threshold \(\tau = \tau' + H(Dif(\lambda))\) can be applied to the UW-entropy map for specular pixel identification. An image pixel is identified as specular (SM) if \(H(Tot(\lambda))\) is above a threshold (\(\tau\)). We assume that an image pixel is equal to the spectrum of total reflection (i.e., \(H(Tot(\lambda)) = H(I)\)), thus the specular mask is given as:

\[
SM(x) = \begin{cases} 
1, & \text{if } H(I) > \tau \\
0, & \text{otherwise.}
\end{cases} \tag{6}
\]

For our experiments, \(\tau \in (0, 0.5)\) has been empirically determined to give plausible results (Fig. 4). The above specularity identification approach is inspired by the work of Tian and Clark [TC13]. Please refer to this work for details.

3.1.2. Intensity Reduction of Specular Pixels

The highlights or specularity is efficiently captured by the positive coefficients in a luminance or intensity sub-band [BBPA15, BSM'18]. For this purpose, we perform multi-scale decomposition of the intensity image \((I)\) by repetitive edge-aware image filtering to obtain an intensity scale-space. In each repetition the spatial extent for the edge-aware filter is doubled producing a series of images of increasing smoothness. A fast way to achieve this on an iPhone is by downsampling the intensity image and then performing edge-preserving upsampling (CIEE) with original intensity image as guide, while the downsampling factor is doubled in each repetition. Subsequently a sub-band (or a frequency band) is obtained by taking the difference between the current and the next scale. A straightforward way to reduce the specular component is to scale the positive coefficients in a sub-band with a constant \(\kappa < 1\). In principle, the above operation will also erode image regions which are both, diffuse and bright. We omit such cases by checking for positive coefficients only within the specular mask (Sec. 3.1.1). A common observation regarding specularity is its occurrence as smooth patches of highlights along with some sparse irregularities due to rough object surfaces. To address these two aspects of specularity distribution, we reduce the positive coefficients of high-frequency \((\kappa_h)\) and low-frequency \((\kappa_l)\) sub-bands separately (Fig. 5). For all of our experiments, we use the values \(-0.5 \leq \kappa_h, \kappa_l \leq 0.2\). Even though we use this approach to reduce specularities, it can be easily extended (by using \(\kappa_h, \kappa_l > 1\)) to

![Flowchart of our specularity removal pipeline described in Sec. 3.1. Note the chroma inpainting depicted by the inset.](image-url)

Figure 3: Flowchart of our specularity removal pipeline described in Sec. 3.1. Note the chroma inpainting depicted by the inset.

![Input image and corresponding specularity mask with increasing value of threshold \(\tau\). Note that with a low threshold value, even diffuse pixels are marked as specular. On the other hand, with a higher threshold, some of specular pixels are missed.](image-url)

Figure 4: Input image and corresponding specularity mask with increasing value of threshold \(\tau\). Note that with a low threshold value, even diffuse pixels are marked as specular. On the other hand, with a higher threshold, some of specular pixels are missed.
seamlessly enhance it for appearance editing [BBPA15] (see supplementary material).

3.1.3. Chroma Inpainting of Specular Pixels

For saturated specular pixels, the chromaticity image might have little or no information. We fill in this missing detail from neighboring pixels using iterative bilateral filtering [TM98]. The initial chromaticity image with the missing information in specular pixels is considered as $C_0$, and after $k + 1$ iteration the modified image is given as

$$C^{k+1}(p) = \frac{1}{W_p} \sum_{q \in M(p)} G_a(||p - q||) \cdot G_s(||C^k(p) - C^k(q)||) \cdot C^k(q),$$

(7)

where the normalization factor $W_p$ is computed as:

$$W_p = \sum_{q \in M(p)} G_a(||p - q||) \cdot G_s(||C^k(p) - C^k(q)||).$$

(8)

The amount of filtering in each iteration is controlled by parameters $\sigma_a$ and $\sigma_r$ for image $C$. As seen in Eqn. 7, the next iteration of chromaticity image is a normalized weighted average of the current one: where $G_a$ is a spatial Gaussian that decreases the contribution of distant pixels, $G_s$ is a range Gaussian that decreases the contribution of pixels that vary in intensity from $C^k(p)$. We search for neighboring pixels in a square pixel window, $M(p)$, of length $(5,15)$ pixels. In principal, any sophisticated inpainting algorithm can be used for this purpose. However, we chose the above procedure because of its locality enabling parallel processing. The range of the inpainting parameters is: $\sigma_a \in (2,8)$ and $\sigma_r \in (0,2,4,0)$.

3.2. Intrinsic Decomposition of RGB-D Images

In this section, we describe our optimization framework for decomposition of the resulting diffuse image (Fig. 7). We assume monochromatic, white illumination similar to previous IID methods, thus shading is scalar-valued and image intensity $I$ (Eqn. 3) is used as shading initialization for the optimization framework. Initial albedo is defined accordingly using Eqn. 2. We logarithmically linearize the constraints to enable simpler optimization strategies, a common practice in previous methods [BKPB17].

$$i(x) = a(x) + s(x)$$

(9)

In the above formulation, the lower case letters of $i, a, \text{ and } s$ denotes log values of $I, A, \text{ and } S$ respectively at pixel location $x$. In order to avoid log indeterminacy at close to zero values we add an offset for logarithm computation i.e., $i_d = \log(I_d + \epsilon)$, for all our experiments we set $\epsilon = 1.4$. We enforce the constraints per color channel in the log-domain, i.e., $i_d[c] \approx a[c] + s$ for $c \in \{R, G, B\}$. For our decomposition, we solve for both $a$ and $s$ simultaneously by minimizing the energy function,

$$E(x) = \frac{1}{2} \left( \lambda_a E_a(x) + \lambda_{ia} E_{ia}(x) + \lambda_{s} E_s(x) + \lambda_{sp} ||a(x)||_1 \right)$$

(10)

where $\lambda_{a}, \lambda_{ia}, \lambda_{s}, \text{ and } \lambda_{sp}$ are data, retinex-albedo smoothness, and retinex-shading smoothness terms respectively with their corresponding weights. We use a $L_1$ regularizer to enforce sparsity in the resulting albedo controlled by the weight $\lambda_{sp}$.

![Figure 5: Effect of high frequency (HF) and low frequency (LF) specularity removal on an input image.](image)

3.2.1. Data Term

The data term ensures that the image is equal to the sum of resulting albedo and shading in the log-domain. To make the solution robust, this term is weighted by pixel intensity to avoid contributions from noisy low-intensity pixels:

$$E_d(x) = L(x) \left( ||i(x) - s(x) - a(x)||^2 \right).$$

(11)

We minimize the energy function (Eqn. 10) with respect to albedo and shading separately using an iterative solver. The data term exclusively contributes in the gradient-of-energy w.r.t. both albedo as well as shading, thus coupling both the minimization. The weighting of the energy term is controlled by $\lambda_d \in \{0.005, 0.05\}$.

3.2.2. Retinex Terms

The Retinex Theory [LM71] forms the basis of many intrinsic decomposition techniques [BKPB17]. It imposes priors on how edges vary differently for albedo and shading. Most of the existing methods assume that an image edge is either an albedo or a shading edge. However, this is not always true and an edge can be present due to both albedo and shading. Moreover, we can identify the shading edges efficiently using the given depth data. Thus, we utilize the Retinex theory and impose constraints on albedo and shading smoothness separately.

Albedo Smoothness. Ideally, an albedo image should be piece-wise smooth. A straightforward way to achieve this is to perform edge-preserving smoothing. We employ a weighting function to identify and prevent smoothing at prominent albedo edges,

$$E_{ia}(x) = \sum_{y \in N(x)} w_a(x,y)||a(x) - a(y)||^2$$

(12)

The edge weight is controlled by a parameter $\alpha_{ia}$, where a relatively higher value ensures texture preservation,

$$w_a(x,y) = \exp \left( -\alpha_{ia}||a(x) - a(y)||^2 \right)$$

(13)
For all our experiments, we use $\alpha_{tr} \in (5.0, 20.0)$ and consider a $3 \times 3$ pixel neighborhood $N(x)$ around pixel $x$. The weighting of the energy term is regulated by $\lambda_{tr} \in (2.0, 40.0)$.

Shading Smoothness. Ideally, a shading image should be smooth except for discontinuities due to irregular scene geometry or indirect illumination (such as inter-reflections and shadows). We assume only direct-illumination and ignore discontinuities due to the latter. By only taking scene geometry into consideration, we expect two scene points to have similar shading if they have similar position and normal vectors [RH01]. The position vectors are constructed as $[x, y, z]^T$ where $x, y$ are pixel coordinates and $z$ is the corresponding depth. The normal vector $[n_x, n_y, n_z]^T$ is constructed using the depth $D(x)$ as,

$$n = [\nabla_x D, \nabla_y D, 1.0]^T$$ (14)

$\nabla_x D$ and $\nabla_y D$ represent depth gradients in horizontal and vertical directions. The normalized position vector and normal vector is combined to construct a feature vector $f$ (for a given pixel $x$): $[x, y, z, n_x, n_y, n_z]^T$. Thus, all pixels are embedded in a six-dimensional feature space. The distance between two pixels in this feature space is used to construct a weight map,

$$w(x, y) = \exp(-\alpha_{tr}|f(x) - f(y)|^2)$$ (15)

The above weight preserves shading variations, captured as distance in feature space and the overall constraint is formulated as,

$$E_{fs}(x) = \sum_{y \in N(x)} w(x, y)||s(x) - s(y)||^2$$ (16)

Similar to the previous term, $N(x)$ represents the $3 \times 3$ pixel neighborhood around pixel $x$. The weight is controlled by a parameter $\alpha_{fs}$; for all our experiments we use $\alpha_{fs} \in (20.0, 200.0)$. The weightage of the energy term is regulated by $\lambda_{fs} \in (15.0, 100.0)$. The feature space introduced above is based on the work of Chen and Koltun [CK13]. However, we consider this distance only in a local neighborhood to increase runtime performance.

3.2.3. Optimization Solver

All the energy terms discussed above are smooth and convex except for the $L_1$ regularizer, which is specific for albedo. This allows for a straightforward energy minimization w.r.t. shading. For both albedo and shading we minimize the energy iteratively. By using an iterative solver, we overcome the limitation of storing a large matrix in memory and calculating its inverse. Moreover, an iterative scheme allows us to stop the solver once we achieve plausible results. A shading update $\hat{s}^{k+1}$ is obtained by employing Stochastic Gradient Descent (SGD) with momentum [Qu99],

$$\hat{s}^{k+1} = \hat{s} - \alpha\nabla E(\hat{s}) + \beta(\hat{s} - \hat{s}^{k-1})$$ (17)

where $\alpha$ and $\beta$ are the step size parameters, $\nabla E$ is the energy gradient w.r.t. shading and $k$ is the iteration count.

In order to enforce albedo sparsity, we utilize an $L_1$ regularizer for albedo. The regularizer is convex but not smooth and thus makes the minimization of energy w.r.t. albedo challenging. The solution for a class of problems that aim to solve for,

$$\arg\min_{a \in \mathbb{R}^n} g(a) + h(a)$$ (18)

where $g(a)$ is smooth and $h(a)$ is non-smooth while both are convex, is generally given by proximal gradient descent (PGD) [LM79]. A more efficient way to solve the above is proposed by Ochs et al. [OCPB14] in their iPiano algorithm with the following update scheme,

$$a^{k+1} = (I + \alpha \delta h)^{-1} \left( a^k - \alpha \nabla g(a^k) + \beta (a^k - a^{k-1}) \right)$$ (19)

where the step size parameters $\alpha$ and $\beta$ are same as in 17. The inertial term makes iPiano more effective than PGD, where the update scheme comprises of only forward descent step and backward proximal mapping. For the special case where $h(a) = \lambda ||a||_1$ the proximal operator is given by soft thresholding,

$$(I + \alpha \delta h)^{-1} (u) = \max\{|u| - \alpha \delta, 0\} \cdot \text{sgn}(u)$$ (20)

For our problem, the data (3.2.1) and retinaex terms (3.2.2) are smooth and their sum can replace $g$ in Eqn. 18. The $L_1$ regularization is achieved with $h = \lambda_0 ||a||_1$. The regularized albedo is solved for iteratively using Eqns. 19 and 20. For most of our experiments, $\alpha = 0.003, \beta = 0.015$, and $\lambda_0 = 0.15$ yield plausible results.

Our stopping criteria is a trade-off between performance and accuracy, we do not compute energy residue for this purpose. We aim to achieve a close to interactive performance with visually convincing application results. To this end, we empirically determined 100 iterations to be a sufficient approximation (Fig. 6).

4. Evaluation

We evaluated our approach for a variety of real-world images and ground truth data. We perform qualitative comparisons with recent methods and quantitative evaluations with existing datasets for both specularity removal and intrinsic decomposition.

Specularity Removal. We compare our method against recent specularity removal techniques by Fu et al. [FZS19], Akashi et al. [AO16], Yang et al. [YW A10], and Shen et al. [SC09]. For the method of Fu et al., the results were generously provided by the authors, and for others we use the implementation by Vítor Ramos [Ram20] to generate the results. We observe that most of the existing specularity removal techniques are not well suited for real-world images. The method by Fu et al., which is especially tailored for real-world scenario, also struggles to handle high-resolution images. The proposed algorithm performs better than state-of-the-art works for natural images (Fig. 7). It is comparable to results in a controlled lab setting (see supplementary material). Moreover, our method works at interactive rates on a mobile device for high-resolution images. Please refer to the supplemental material for how the intermediate steps improve the output quality.

Note that the comparisons for specularity removal are performed using the desktop-based implementation of our algorithm, which makes use of guided image filtering for multi-scale decomposition of image intensity. For our mobile version, we replace guided filtering by inbuilt edge-aware filters on iOS (iPhone) to achieve interactive performance while compromising on quality.
Table 1: Quantitative evaluation for intrinsic decomposition (pixel value is scaled between 0 to 1), the lower the error value, the better.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ours</th>
<th>Bell</th>
<th>Lettry</th>
<th>Jeon</th>
<th>Ours</th>
<th>Bell</th>
<th>Lettry</th>
<th>Jeon</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFID</td>
<td>0.074</td>
<td>0.066</td>
<td>0.072</td>
<td>0.085</td>
<td>0.094</td>
<td>0.159</td>
<td>0.144</td>
<td>0.035</td>
</tr>
<tr>
<td>MPI-Sintel</td>
<td>0.245</td>
<td>0.041</td>
<td>0.044</td>
<td>0.034</td>
<td>0.325</td>
<td>0.244</td>
<td>0.253</td>
<td>0.258</td>
</tr>
</tbody>
</table>

**Intrinsic Decomposition.** We compare our intrinsic decomposition results with a RGB (Bell et al. [BBS14]), a RGB-D (Jeon et al. [JCTL14]) and a learning (Lettry et al. [LVVG18]) based technique to cover a broad range of methods. We use the implementations provided by the authors. Our results are comparable to the above methods (Fig. 12). Note that the methods of Bell et al. and Jeon et al. perform at an order of magnitude slower than ours on a GPU-enabled desktop system. Moreover, unlike ours the quality of their result for indoor and outdoor scene is not consistent. They perform quite well for indoor scenes however, their output quality degrade significantly for outdoor scenes (see supplementary material). Even though the time taken by Lettry et al. is comparable to our mobile-phone based technique, we perform comparatively better in terms of output quality.

**Quantitative Evaluation.** For a quantitative evaluation, we require a dataset that includes ground truth depth, albedo, shading, and specularly. To this end, we use the Light-Field Intrinsic Dataset (LFID) [SBZ’18]. We also test only the intrinsic decomposition component of our approach on the MPI-Sintel dataset [BWSB12]. We use MSE and DSSIM as error metric while comparing the computed albedo (for intrinsic decomposition evaluation) and diffuse image (for specular removal evaluation) with the respective ground truth. We compare our intrinsic decomposition results with other methods (specified in Fig. 12) in Tab. 1. For the MPI-Sintel case, we consider one frame from all the scenes, and for LFID we use three views from Street Guitar and Wood Metal light-fields. Our method performs comparatively better on LFID than MPI-Sintel dataset because the modeling assumptions for LFID is similar to ours which is physically more accurate. For specular removal we employ the desktop implementation of our approach and achieve MSE and DSSIM values of 0.001 and 0.018 respectively.

**Run-time Performance.** Our whole processing pipeline has been implemented on an iPhone 11 Pro smartphone running on the iOS 13 operating system with an Apple A13 Bionic processor and 4GB of RAM. We make use of Apple’s Metal API for GPU-based processing. The captured image is downsampled by a factor of 0.3 for interactive performance while maintaining sufficient quality. The resulting image resolution is of 1120 × 1504 pixels and the corresponding depth map is either of resolution 480 × 640 pixels for the front facing true-depth sensor or 240 × 320 pixels for the back camera passive stereo setup. We scale the depth map using built-in filters to match the image resolution, for consistent processing. On average, the pre-processing step of specular removal takes 0.1 seconds. For solving the optimization described in Sec. 3.2, we employ an iterative solver and analyze its performance with an increase in number of iterations for two kernel resolutions of 3 × 3 and 5 × 5 pixels. Our goal is to achieve visibly plausible results with interactive processing. We empirically determine 100 iterations as a good trade-off for the above requirement with an execution time of ≈ 1.5 seconds for a 3 × 3 pixels kernel resolution (Fig. 6). Our material editing pass requires to compute sub-bands in a pre-processing stage for each intrinsic layer, which takes ≈ 3.5 seconds. Subsequent thereto, the editing is interactive. The other application components run interactively allowing for seamless editing.

**5. Applications**

A perfect, physically accurate editing of a photo would require full inverse rendering with high precision. However, one can achieve convincing material [BSM’18, KFBB06] and volumetric media [NN03] editing even without the above. The intrinsic decomposition output can also be effectively used for enhancing image stylization results [MZRT16]. The following applications in our work are based on the above observations.

**5.1. Material Appearance Editing**

Our material editing framework is based on the work of Beigpour et al. [BSM’18], where the authors modify the intensity of albedo, shading, and specularly using band-sifting filters [BBPA15]. The modified intrinsic layers are merged to form the output image \( (I_{out}) \) with edited appearance,

\[
I_{out} = A(r_{1m_{1}g_{1}}, \eta_{1}) \cdot S(r_{2m_{2}g_{2}}, \eta_{2}) + I_{r}(r_{3m_{3}g_{3}}, \eta_{3})
\]

(21)

where \( r_{im_{ig}}, \) with \( i \in \{1, 2, 3\} \) represents a component of respective intrinsic layer—\(A, S, \) and \( I_{r}\) (described in Eqsns. 1 and 2)—intensity, that is band-sifted. The component categorization is based on the following signal attributes: spatial frequency \((r), \) magnitude \((m), \) and sign \((g)\). Only a predefined set of sub-categories is defined: \( r_{i} \in \{H, L, A\}, m_{i} \in \{H, L, A\}, g_{i} \in \{P, N, A\}, \) where \( H \) and \( L \) denote high and low frequency/magnitude range, \( P \) and \( N \) represent positive and negative values, and \( A \) denote “all”, i.e., the complete category. The amount of sifting is controlled by the scaling factor \( \eta_{i}. \) We can boost \((\eta_{i} > 1), \) reduce \((0 < \eta_{i} < 1), \) or invert \((\eta_{i} < 0)\) the selected component respectively.

In our framework, we replace the original manual object-segmentation with a mask generation step based on machine learn-
Figure 7: Comparison of specularity removal for real-world images. The figure contains input image and the corresponding diffuse image obtained using ours, Fu et al. [FZS⁺19], Akashi et al. [AO16], Yang et al. [YWA10], and Shen et al. [SC09] specularity removal methods.

Figure 8: Comparing our translucency effect with [BSM⁺18].

Figure 9: Input image and atmospheric edit with virtual fog.

Input Ours Fu et al. [FZS⁺19] Akashi et al. [AO16] Yang et al. [YWA10] Shen et al. [SC09]

We perform atmospheric editing as de-weathering and relighting in the form of God rays. Our de-weathering approach is based on the work of Narasimhan et al. [NN03], which enables to synthesize an image-based fog-like appearance. According to their de-weathering model, the output image \( I_{\text{out}} \) can be expressed as a linear combination of the input image \( I_{\text{in}} \) and the brightness of the sky \( F \) using the given depth data \( D \):

\[
I_{\text{out}} = I_{\text{in}} \cdot \exp(-\theta D) + F \cdot \left(1 - \exp(-\theta D)\right)
\]  

(22)

The scattering parameter \( \theta \in (0, 2.7) \) controls the above linear combination. We further improved the result by using an ad-
5.3. Image Stylization using Intrinsic Layers

We implement a cartoon stylization pipeline based on the extended difference-of-Gaussians (XDoG) filter by Winnemöller et al. [WOG06, WKO12]. The filtering pipeline is enhanced using the computed intrinsic layers as follows.

5.3.1. Depth-based Edge Detection

Color-based edge detection methods generally fail to accurately identify edges in the case of smooth or non-apparent lighting transitions between objects, and might over-emphasize noisy patterns in the image. To improve these issues and enhance geometric edges in the image, we make use of the given depth data.

We intensify depth variations by computing the angle-sharpness ($\phi \in [0, 1]$), defined as the magnitude of normal vectors pointing away from the camera, $\phi = \frac{||N||}{D}$, where the image normal $N$ (produced by Eqn. 14) and depth $D$ is used to decrease the edge magnitude for distant objects of usually noisy depth information. The angle-sharpness is used to boost gradients—derived from the structure tensor—in areas of high angle-sharpness,

$$ST_\phi = \begin{cases} (\phi \omega + 1)ST_D, & \text{if } \phi < \frac{(\omega-1)}{\omega} \\ ST_D, & \text{otherwise} \end{cases}$$

where $ST_D$ is the structure tensor calculated on the depth image in log space, and $\omega \in [0, 1000]$ is a boost factor for low-luminosity edges (we use $\omega = 100$ in our experiments). Smoothing $ST_\phi$ with a Gaussian yields the smoothed structure tensor from which the edge tangent flow is derived via an eigenanalysis [BWBM06].

The flow-based difference-of-Gaussians, as defined in [KD08, WKO12], is then applied on the angle-sharpness $\phi$ along the flow field induced by the smoothed $ST_\phi$ to obtain coherent depth edges (Fig. 10(d) and supplementary material).

5.3.2. Albedo and Shading Combination

In the color-based cartoon stylization pipeline, the luminance values of the input are smoothed and quantized to create a flat material effect. Through the use of our image decomposition framework, shading and albedo can be combined in multiple ways to enhance this stylization. Using albedo only, a flat cartoon-like style can be created (Figs. 10(c) and 13(g)), due to the removal of shading, the output is brighter than the original image and geometric features are mainly indicated by XDoG edges.

There are several ways of abstracting the shading information before recombining it with albedo for enhanced cartoon stylization. Edge-preserving smoothing of the shading layer with a large filter kernel yields an airbrush look (Fig. 10(d)), while quantizing the shading yields a look similar to a classical cartoon stylization [WOG06]. Another method for flattening shading information is to use a segmentation-based approach. We implemented a GPU-based quick-shift filter [VS08] to segment shading according to albedo clusters (Fig. 13(h)). Shading alone, combined with halftoning and edges, can create a vintage effect (Fig. 13(i)). Shading abstraction is a single-channel operation and is recombined uniformly with albedo in RGB space.

5.3.3. Shadows

Shadows in hand-drawn cartoons are an important tool to convey geometric and lighting cues about the scene and are also often strategically placed to emphasize character expressions. A method based on occlusion maps can be used to generate soft-shadows with semi-realistic lighting (Fig. 11(b)). To create less realistic but more cartoon-like hard shadows, we assume that shadows are only set on a foreground object and approximate the lighting based on an angular thresholding of the depth map. For a given pixel, the re-lighted shading $\hat{s}$ is defined as:

$$\hat{s} = \begin{cases} s, & \text{if } ||\arctan(n_z, n_x) - p|| < \theta \text{ and } \arccos\left(\frac{n_z}{|n_x|}\right) < \gamma \\ |s|, & \text{otherwise} \end{cases}$$

where $s$ is the original shading, $n_z$ and $n_x$ are the depth gradient and $p$ is the pixel position.
Figure 12: Comparison of intrinsic decomposition with other methods. The figure contains input image and the corresponding albedo obtained using ours, Bell et al. [BBS14], Jeon et al. [JCTL14] and Lettry et al. [LVVG18] intrinsic decomposition methods. Please see supplementary material for shading results.

where $l \in [0, 2]$ is a luminance multiplier that either emulates shadow ($l < 1$) or lighting ($l > 1$), $\rho$ is an angle that controls the shadow direction around the foreground object, and $\theta$ is the shadow circumference that is calculated by thresholding the angle deviation from $\rho$. To emulate the depth of the light source, normal z-angle thresholding includes only surface-normals that point at least $\gamma$ degrees away from the camera (Fig. 11(c), with $\rho = \pi, \theta = \pi, \gamma = 0.01$).

6. Discussion and Limitations

Our goal is to provide photorealistic, interactive image editing using readily available RGB-D data on high-end smartphones. To this end, we implement an intrinsic decomposition technique capable of running on smartphones. The trade-offs between performance and accuracy (Sec. 4) is biased towards performance for the sake of interactivity, but nonetheless we are able to obtain high quality results. Unlike most of the previous methods, we perform a preprocessing step of specularity removal and do not assume “only diffuse reflection” in the scene. We observe that the above ambiguity, apart from state-of-the-art methods, is also present in the popular intrinsic dataset – MPI-Sintel [BWSB12]. For MPI-Sintel, specularity is encoded as part of the shading information, which is physically inaccurate. Our observations suggest that specularities are formed as a complex interplay between reflectance and shading, and thus should be handled separately.

The extracted intrinsic layers—along with available depth data—allows for a variety of image manipulations. However, we make some simplifying assumptions to achieve interactive processing and cope with the limited computing capabilities of mobile phones—note that most of these assumptions are also common for many state-of-the-art desktop-based methods. First of all, we only consider direct illumination and ignore the multi-bounce effects of light, such as color bleeding and soft shadows. The assumption of white colored illumination is also not valid for many
real-world scenes. A multi-color illuminant can cause color variations that can be mistakenly classified as albedo instead of shading. We initialize albedo with a chromaticity image for improved performance [MZRT16], and do not perform clustering in the chromaticity domain, which leads to color shifts especially in regions with low pixel-intensity. Despite the above limitations, our technique gives plausible application results at interactive frame rates.

7. Conclusions and Future Work

We present a system approach that performs intrinsic image decomposition on smartphones. To the best of our knowledge, it is the first such approach for smartphones. Using the depth data captured by built-in depth sensors on smartphones, together with a novel specularity removal pre-processing step, we are able to obtain high-quality results. A GPU-based implementation using the Metal API allows for close to interactive optimization solving and interactive image editing. A qualitative evaluation shows that our specularity removal method performs better than state-of-the-art approaches for real-world images. The albedo and shading layer results are on par with state-of-the-art desktop-based methods. Finally, we showcase how the intrinsic layers can be used for a variety of image-editing applications.

A mobile-based intrinsic decomposition, as provided in this work, could be used for photo-realistic image editing in Augmented Reality (AR) applications. As part of future work, we aim to relax some of the existing assumptions and address image scenes with multi-color illuminant [BT17] and indirect illumination ef-
fects [MSZ’19]. We also assume that the super-resolution of depth maps can further enhance our results [VAE’19]. Moreover, we believe that our specular pixel detection can be made more robust with a non-binary thresholding and better handling of bright image regions.

Acknowledgements

We thank the anonymous reviewers for their valuable feedback. We thank Mohammad Shafiei and Mahesh Chandra for valuable discussion w.r.l optimization solver. We thank Florence Böttger for her help with the development of atmospheric editing pipeline. We thank Ariane Morassi Sasso, Harry Freitas da Cruz, Orhan Konak and Jessica Jall for patiently posing for the pictures. This work was funded by the German Federal Ministry of Education and Research (BMBF) (through grants 01IS15041 – ”mdViProject” and 01IS19006 – “KI-Labor ITSE”) and the Research School on “Service-Oriented Systems Engineering” of the Hasso Plattner Institute.

References


[AO16] AKASHI Y., OKATANI T.: Separation of reflection components by sparse non-negative matrix factorization. Computer Vision and Image Understanding 146, C (May 2016), 77–85. 6, 7


[BBS14] BELL S., BALA K., SNAVELY N.: Intrinsic images in the wild. ACM Transactions on Graphics 33, 4 (July 2014), 1, 2, 3, 7, 10


© 2021 The Author(s)

Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.


[LMT99] Lions P. L., Mercier B.: Splitting algorithms for the sum of two nonlinear operators. SIAM Journal on Numerical Analysis 16, 6 (1979), 964–979. 6


