Pushpins for Edit Propagation

Mylo, Marlon
Fraunhofer FKIE
KOM Department
Fraunhoferstraße 20
53343, Wachtberg,
Germany
and
University of Bonn
Institute of Computer
Science II
Regina-Pacis-Weg 3
53113, Bonn, Germany
mylo@cs.uni-bonn.de

Klein, Reinhard
University of Bonn
Institute of Computer
Science II
Regina-Pacis-Weg 3
53113, Bonn, Germany
rk@cs.uni-bonn.de

ABSTRACT

In this paper we present an approach for stroke-input based foreground estimation of measured materials with a near regular structure. To enable extraction of high-quality editing masks even from difficult materials, we combine a state of the art lattice-detection algorithm with a novel frequency convolution scheme, which we call *pushpins*. Despite being highly specialized, we consider this use-case as important for material design. A comparison with other state of the art editing and material recognition approaches will give proof of the robustness and ability of our algorithm.

Keywords

SVBRDF, Near Regular Texture, edit propagation.

1 INTRODUCTION

Measured materials are used to render 3D-scenes into images which evoke the impression of photorealism. While being able to edit those digital material representations is highly desirable for many applications, e.g. in film and advertising, manipulations are still a challenging tasks. Solving this problem may spare acquisition costs and admits to construct imaginary materials which appear as if they were real.

Editing measured materials is indivisibly tied to the process of isolating the geometrical or the radiometric regions which shall be manipulated.

While many brilliant algorithms have been published to master this classification problem, the productive use of those algorithms has to meet high demands. Small misclassifications lead to ugly artefacts in the resulting renderings and have to be corrected in tedious handcraft. The increasing quality in image segmentation and image matting is mostly based on a subtle exploitation of colour spaces and spatial continuity constraints. While those approaches do also apply for segmentation of materials, the results are often not good enough because different material components can very often not be distinguished by colour. But most digital material representations provide more than one diffuse colour channel. And many materials bear a *near regular structure* (*NRS*). In this paper we want to make use of those two facts to generate editing masks for measured materials in a quality which makes handcrafted optical debugging steps unnecessary. Our approach consists of a separated lattice detection step and and a classification by a *support vector machine* (*SVM*). The SVM-classification allows to use complex, high-dimensional descriptors whereas the lattice detection enables to construct only one model tile-mask and to propagate this mask via the detected lattice.

The technical contribution of this paper is 2-fold:

1. We provide a workchain to robustly solve the foreground estimation problem for measured materials with a NRS.
2. We introduce a convolutional technique to tag texels which have a similar environment like a given seed texel of the same material patch. This similarity recognition step alone is not reliable enough for stable lattice detection but it delivers a global similarity map which may be used to guide the indeterministic lattice detection step.

The structure of the paper is as follows: after a short section on the relevant related work (section 2), we will give an overview (section 3), which provides notations,

the problem statement and a walk-through. The algorithm is presented in section 4 and followed by the evaluation in section 5. The conclusion (section 6) closes with considerations about the possibilities to parallelize our system.

2 RELATED WORK

Editing measured opaque materials is an intensively studied field and there have been by far too many publications to give an exhaustive catalogue in this context. According to [7], interpolated reflectance data may directly be used for rendering materials. But those representations are expensive to store, lack explanatory power and are difficult to edit so there have been many approaches to fit measured reflectance data to analytical reflectance models, like [8,16,19,26]. Editing those analytical representations is still not easy. Some approaches operate directly on the radiometric data like the retargeting approach of An et al. [1] or the manifold based on aging simulation by Wang et al. [25]. Others try to estimate a propagation map, first, to isolate the texels to edit. Pellacini and Lawrence suggested, to use an k-nearest neighbour graph to construct a sparse adjacency matrix [21]. An and Pellacini made another step in this direction with AppProp [2], which has been extended to tabulated reflectance data by Xu et al. [27]. A recent state of the art report by Schmidt et al. [23] gives an extensive overview.

3 OVERVIEW

The overview provides the notations, the problem statement and a short walk through.

3.1 Notations and definitions

In this section we want to clarify our use of language.

Material By *material* we mean the digital representation of an existing or imaginary material-surface together with a description of the light exchange in every point of the surface.

BRDF A *BRDF* maps an incoming and an outgoing light direction onto a wavelength-dependent reflectance probability. Being reflectance distributions, BRDFs are limited to the upper directional hemisphere. In this paper we concentrate on *analytical, measured* BRDFs, meaning, that an analytical reflectance model has been optimized to fit a given set of reflectance measurements. We tested our algorithms also on tabulated reflectance representations. But to describe those reflectance tables in order to make them applicable for usage with a classifier it is necessary to bring them into a comparable format like for example Rusinkiewicz-parametrization [22] which makes a resampling-step necessary and to collect at least some elementary statistics. Investigations of this kind are beyond the scope of this paper.

SVBRDF A *spatially varying BRDF* (*SVBRDF*) is a material where the light exchange is described by a BRDF.

Ashikhmin Shirley reflectance model The measured reflectance distributions are modelled in the way suggested by Peter Ashikhmin and Michael Shirley in 2000 [3]. This is a Phong-like model which additionally controls the eccentricity of the specular lobe and is given by:

$$\begin{aligned}
&\rho(\omega_{\mathbf{in}}, \omega_{\mathbf{out}}) \\
&= \frac{\sqrt{(e_x+1)(e_y+1)}}{8\pi} \frac{\langle \mathbf{n}, \mathbf{h} \rangle^{e_x \cos^2\phi + e_y \sin^2\phi}}{\langle \omega_{in}, \mathbf{h} \rangle \max(\langle \omega_{in}, \mathbf{h} \rangle, \langle \omega_{out}, \mathbf{h} \rangle)} \\
&\cdot (R_s + (1-R_s)(1-\langle \omega_{in}, \mathbf{h} \rangle)^5) \\
&+ R_d (1-R_s) \frac{28}{23\pi} \\
&\cdot \left(1 - \left(1 - \frac{\langle \omega_{in}, \mathbf{n} \rangle}{2}\right)^5\right) \left(1 - \left(1 - \frac{\langle \omega_{out}, \mathbf{n} \rangle}{2}\right)^5\right)
\end{aligned} \tag{1}$$

for the incoming and outgoing directions $\omega_{\mathbf{in}}$ and $\omega_{\mathbf{out}}$. The vector $\mathbf{n}$ is the surface normal, $\mathbf{h} = (\omega_{\mathbf{in}} + \omega_{\mathbf{out}})/||\omega_{\mathbf{in}} + \omega_{\mathbf{out}}||$ and ϕ is the azimuth of $\mathbf{h}$.

This model has four reflectance parameters: the wavelength dependent diffuse and specular reflectance shares R_d and R_s and the surface roughness along the x-axis e_x and the surface roughness along the y-axis e_y. In the following, we will refer to R_d and R_s as the *diffuse colour* and the *specular colour*. We will assume that those colours are RGB colours and the term *lightness* will refer to the HSL description of the RGB-space. We assume that the parameters are stored in rectangular maps.

Being based on the Phong model, the Ashikhmin-Shirley model is neither normalized nor is the distribution function for the lobe physically founded. An up-to-date comparison between anisotropic analytical BRDF-models and a suggestion for a model without the mentioned flaws has been published by Murat et al. [14].

3.2 Problem statement

Given an SVBRDF, where at least some of the parameter channels bear a *roughly* periodic pattern in the following sense: there exists a periodic pattern which may be warped into those channel maps alongside of a small continuous flow field. Here we mean by periodic pattern an image which may be generated from a model tile and a concatenation of translations and rotations according to an appropriate wallpaper group. A specification of the term *small* is difficult and depends not only on the settings of the algorithm but also on the texturizing of the SVBRDF, itself.

Further we assume, that a user has marked a *foreground* component $\mathcal{F}$ of the SVBRDF and a *background* component $\mathcal{B}$ by the use of a stroke input $\mathcal{S}_\mathcal{F}$ for the foreground and a stroke input $\mathcal{S}_\mathcal{B}$ for the background stroke. Than we want to propagate this stroke input in a way that the periodic pattern is respected and a texel with the index i and the average reflectance distribution ρ_i obtains a value α_i which decomposes ρ_i into a convex combination of a foreground BRDF ϕ and a background BRDF ψ

$$\rho_i = \alpha_i \phi_i + (1 - \alpha_i)\psi_i$$

For the classical matting problem, the parameter α is described as opacity or transparency. For our application, this interpretation is not good, as transparency leads to complicated reflectance properties. α should be merely seen as area share of the foreground reflectance distribution. We will define the foreground $\mathcal{F} = \{t_i | \alpha_i = 1\}$, the background $\mathcal{B} = \{t_i | \alpha_i = 0\}$ and the boundary $\partial = \{t_i | 0 < \alpha_i < 1\}$.

3.3 Walk through

In figure 1 you can see an overview of our new algorithm. As input we take a SVBRDF together with a stroke input. Then we apply in parallel a segmentation via a support vector machine (paragraph 4.1.2) on the descriptors described in paragraph 4.1.1 and estimate a lattice on the diffuse colour (paragraph 4.2). Based on the detected lattice we extract a model tile (paragraph 4.3.1), calculate an optical flow between this model tile (paragraph 4.3.2) and all other tiles and warp the tiled SVM-classification results into the model tile. This set of warped masks is used to compose an average tile-mask which is then warped into the original tile positions (paragraph 4.3.3).

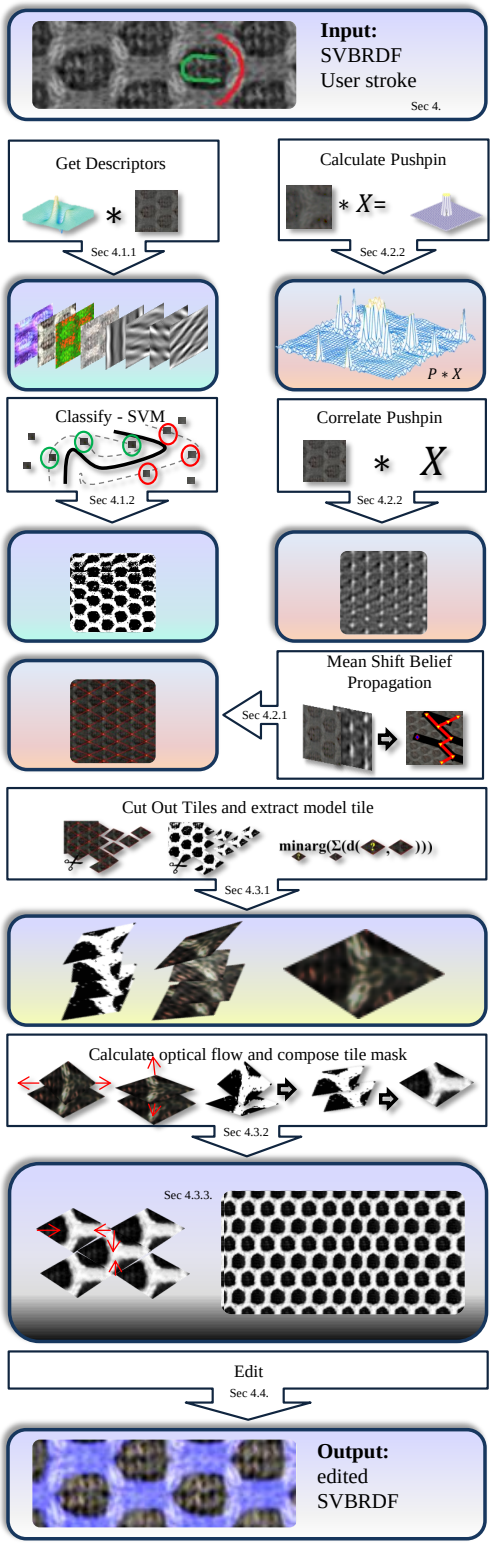


Figure 1: Overview: the arrows contain the processing steps and the boxes show the resulting data. In the top of the arrows we give the numbers of the paragraphs where the processing step is described in detail.

4 THE ALGORITHM IN DETAIL

In this section we want to describe the algorithm in detail.

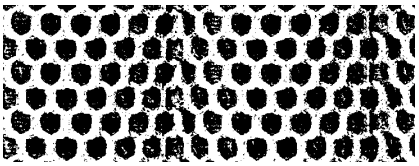


Figure 2: The mask resulting from the SVM classification step.

4.1 Classification

Based on the stroke input, we classify in this step all texels of the material probe, without reference to the NRS, into foreground and background texels. We tried several different descriptors and several different classifiers:

4.1.1 Descriptors

Additionally to the 8 reflectance parameters and the 2 parameters of the surface normal provided by the Ashikhmin-Shirley model (equation 1), we add the filter responses of Gabor filters. We use 8 different orientations and a wavelength of 3 texels. Gabor filters are applied to the volume-channel of the diffuse color. This strengthens the influence of line features on the classification result. We compare every texel on a patch with size 5x5 texel. So the dimension of our descriptor is altogether (8 + 2 + 8) x 5 x 5 = 450.

4.1.2 Classifier

We tried different state of the art classifiers: Support Vector Machines [6], Deep Belief Networks [10] and Convolutional Neural Networks [15]. The latter have been implemented in Theano for Python, for the SVM we used the implementation by Chang [5].

Though we made good experiences with neural networks in the past, they failed in the current scenario. According to a rule of thumb given in [18], the number of samples should be equal or more than the number of weights of the neural network. As stated in paragraph 4.1.1, the descriptor of a texel has the dimension of 450 which makes, dependent on the concrete topology, about 50,000 weights in a three layer neural network, whereas a stroke input provides between 100 and 500 samples. So the networks have simply not enough data for training. SVMs, in contrast, can be trained with a small amount of data and are easy to apply and quickly trained.

The trick of the SVM is that it estimates a decision boundary in an infinite dimensional space which makes it possible to have non-linear boundaries between clusters. By maximizing the margin between the decision boundary and the training-samples, the SVM reaches even in the linear setting better generalization than other linear classifiers. For the optimization, it is not necessary to map the data into the infinite dimensional feature space, but it is enough to calculate the inner product (so called *Kernel Trick*). We use radial basis functions as inner product kernels and parameter estimation is done by grid-search and 5-fold cross-validation.

In figure 2 you can see that the result of the svm classification step is already a good segmentation. Still there are some noticeable misclassifications.

4.2 Lattice detection

Our algorithm gains its strength from the combination of lattice detection and pattern-recognition. In our tests, the most successful approach to detecting lattices was the *mean shift belief propagation* (*MSBP*), published by Park et al. [20].

4.2.1 Mean Shift Belief Propagation

MSBP makes the assumption that a repeating structure in an image is a slightly deformed periodic pattern. As such it is possible to find an ideal pattern element and two linearly independent lattice base vectors to reconstruct this periodic pattern by operating via the corresponding wallpaper group [9, 17]. By clustering points of interest, MSBP estimates the base vectors for the periodic pattern and a seed point, and the algorithm extracts a characteristic tile around this seed point. The lattice base vectors define symmetry-mappings, so the seed point and all symmetry-images of this seed point may be mapped to further symmetry-images by translation along the base vectors. Those images are the vertices of the constructed lattice. As the lattice is deformed by assumption, the exact symmetry mapping has to be found by searching for a good fit for the characteristic tile in the area of the estimated new vertex position. This search is done for all new lattice-vertex candidates simultaneously, meaning that the search for two neighbouring vertices is constrained by an energy term which punishes deviation from the according base translation. Mean shift belief propagation has proven to be an extremely powerful algorithm. Still we had to struggle with two problems:

1. The results are not deterministic.
2. Regions of big distortions like the fold in the grey mesh material often stop the expansion of the lattice.

Both difficulties are illustrated in figure 3. The result of MSBP, reflected by the red lattice in the left image was successful: the algorithm found the smallest possible tile and the lattice covers the whole material patch. On the right image, we have an example for an abortive run of MSBP: you can see that the algorithm was not able to cross the fold in the material and the base vectors are the sum and the difference of the base vectors found in the right image.

We clear this problem by the use of a cross-correlation based approach we call *pushpins*.

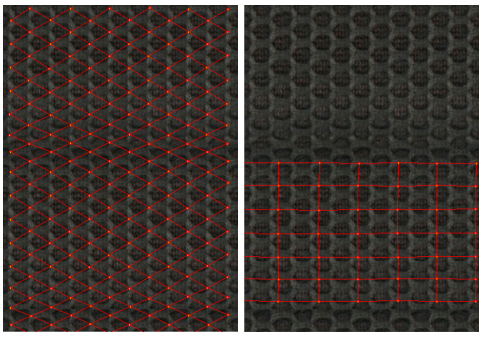


Figure 3: Two different runs of mean shift belief propagation on the grey mesh material.

4.2.2 *Pushpins*

To make the results of MSBP more stable and more predictable, we guide the lattice detection step by a weaker but therefore global repetition detector. The main idea is to mask the frequency spectrum of a given material $\mathcal{T}$ in such a way that a specific quadratic region $\mathcal{P} \subset \mathcal{T}$ in the spatial domain and therefore all similar regions in the spatial domain show a peak. This may be done by cross correlating $\mathcal{T}$ with $\mathcal{P}$, but simple cross correlation does not bring the desired results. Instead, we construct a patch which generates a peak when convolved with $\mathcal{P}$.

Masking the frequency domain in order to isolate particular features is a common technique in signal processing but we did not find our approach in the computer graphics literature so we will briefly introduce it.

Lets first assume that we are looking for a texture $\mathcal{X}$ with the same size as $\mathcal{P}$ so that:

$$\mathcal{P} * \mathcal{X} = \delta$$

where δ models a spike in form of the dirac distribution. By convolution and application of the convolution theorem, we get:

$$\mathbf{F}\mathcal{P}\mathbf{F}\mathcal{X} = \mathcal{C}$$

for a constant texture $\mathcal{C}$. $\mathbf{F}$ is the fourier-transform and $\mathbf{F}^{-1}$ its inverse. Thus, a candidate for $\mathcal{X}$ is:

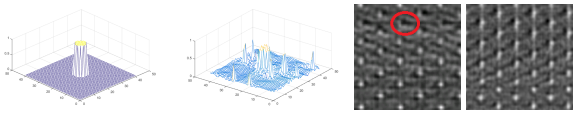
$$\mathbf{F}^{-1}\left(\frac{\mathcal{C}}{\mathbf{F}\mathcal{P}}\right)$$

Here we presumed correct scaling and frequency sampling and point wise multiplication.

For numerical reasons it is advisable to suppress high frequencies. Thus we substitute C by a gaussian filter $\mathcal{G}$ and get:

$$\mathcal{P} * \mathbf{F}^{-1}\left(\frac{\mathcal{G}_{-1/\sigma\pi^2}}{\mathbf{F}\mathcal{P}}\right) = \mathcal{G}_{-\sigma}$$

for the variance σ. Note that equation 4.2.2 becomes wrong, when $\mathbf{F}P$ is not continued by zeros, but by the surrounding pixels in the material. This can be circumvented by calculating $\mathcal{X}$ not by convolution but by deconvolution as the solution of



(a) Mesh($\mathcal{P} * \mathcal{X}$) (b) Mesh($\mathcal{P} * \mathcal{X}_\Delta$) (c) $\mathcal{T} * \mathcal{X}$ (d) $\mathcal{T} * \mathcal{X}_\Delta$

Figure 4: The effect of regularization to push pins. From left: the response of the original tile ($\mathcal{P}$) to an unregularized pushpin ($\mathcal{X}$), the reponse to a regularized push pin ($\mathcal{X}_\Delta$), the response of a distorted material-patch ($\mathcal{T}$) to the unregularized pushpin and the reponse of the same material-patch to the regularized filter. In image (c) you can see that the unregularized pushpin fails to produce some spikes (see red circle).

$$\sum_{i=0,j=0}^{n-1} \mathcal{X}(i,j)\mathcal{T}(i_0-i, j_0-j) = \mathcal{G}_\sigma(i_0, j_0) \qquad (2)$$

$\forall i_0, j_0 \in \text{supp}(\mathcal{P})$. n is the edge length of $\mathcal{P}$. As n is also the edge length of $\mathcal{X}$, we have the same number of variables and equations.

We will call the solution $\mathcal{X}$ of equation 2 a *pushpin* and $\mathcal{P}(\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor)$ the *puncture* of the pushpin. By *nailhead* we mean the support of $\mathcal{P}$.

A pushpin, constructed in this way, does respond a bit stiff: tiles have to be very similar to the original tile to generate a detectable spike. This may be relaxed massively by using a regularization: instead of solving the equation system 2, we constrain this equation system by a spatial smoothing term namely by the minimization of the discrete laplace operator (Δ). This leads to a minimization problem:

$$\mathcal{X} = \text{argmin}_{\mathcal{W}}\Bigg(\sum_{i_0=0,j_0=0}^{n-1} || \sum_{i=0,j=0}^{n-1} \mathcal{W}(i,j)\mathcal{T}(i_0-i, j_0-j) - \mathcal{G}_\sigma(i_0, j_0)|| + || \sum_{i=0,j=0}^{n-1} \Delta_{(i,j)(k,l)}\mathcal{W}|| \Bigg) \qquad (3)$$

The effectiveness of this regularization step is illustrated in figure 4. Pushpins can be made tolerant against noise or small distortions by adding energyterms to equationsystem 3. And the other way round it is possible to concentrate on certain regions of the pushpin by adding weights to the corresponding equations.

In figure 5 we visualize the influence of push pins to the lattice detection process. We have made several test runs some of which had one or two nodes missing, but we obtained always the same lattices covering the whole material patch.

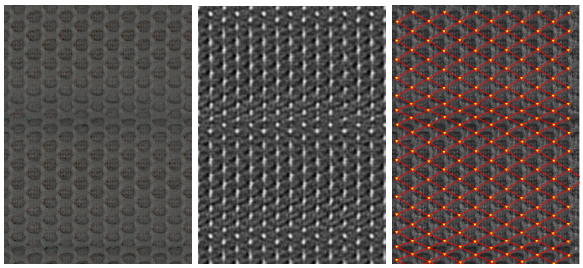


Figure 5: The left image shows the grey mesh material, the image in the middle depicts the filter response of a push pin applied to the volume channel of the diffuse color of the grey mesh material and the third image shows the result of MSBP on a combined map of the filter response and the diffuse channel. Now the lattice detection is extremely stable.

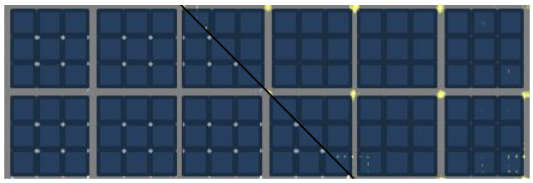


Figure 6: On the left side you can see in light blue the filter response of a pushpin with nailhead radius approximately equal to the size of half a small square on the right side we used a pushpin with a nailhead radius approximately equal to half a big square.

To conctruct a pushpin, it is necessary to determine a centerpoint and a radius. We have chosen the mean of the positions of the stroke inputs as center point and the size of the nailhead was chosen so as to cover the whole stroke.

Nested symmetry groups

Pushpins generate automatically a region of dominance. This shall be demonstrated on a simple example. In figure 6 you may see a simple texture consisting of small squares arranged in groups to bigger squares. On the left side, you can see the response of a pushpin with a nailhead diameter in the size of a small square, on the right side we used a pushpin with a nailhead diameter in the size of a big square. The clipped filter response of the smaller pushpins are held blue the filter response of the bigger pushpins is shown in yellow. You can see that the pushpin on the left side detected the crossings between the small squares whereas the pushpins on the right side detected *exclusively* the crossings between the big squares. This means that pushpins can distinguish between nested symmetry groups. That is an improvement against plain mean shift belief propagation because MSBP simply uses the symmetry group it gets first.

Though pushpins are not limited to a certain number of channels, particularly not to 1, we confine their use to the lightness channel of R_s or R_d. Note that the use of more channels does also lead to more noise in the filter response.

4.3 Generation of a mask

In the next step we combine the results of the classification and of the lattice detection to obtain a model mask tile and a warping field to plaster the whole material patch with this model mask patch. After cutting the mask and the material into a set of tiles which we interpret as distorted version of the same model tile, we extract a model tile, we calculate an optical flow between the model tile and all other tiles and we compose a mask for the whole material probe.

4.3.1 Finding a model tile

To generate a reliable segmentation of a single tile we first choose one tile which is every bodies friend. We assume that changes in the size of tiles are due to perspective distortion. Thus the best fit for an average tile should be a tile with maximum edge-length. So in the first step we resize all tiles to the maximum edge-length. The comparison is made on base of the $L2$-norm applied to the difference of the diffuse channel of two tiles. As the number of tiles is small, we simply apply a brute force approach and compare all tiles pairwise. This procedure is quadratic in the number of tiles, so for big numbers of tiles, the time requirement may be optimized by using a dynamic programming approach. Note that generating a mean tile instead of searching the tile with the most friends is not advised as we want to calculate the optical flow between this model tile and all other tiles. This is more difficult with a mean tile because the algorithm has to find features.

4.3.2 Optical flow

For the estimation of the optical flow between the principal tile and the test tile, we use the algorithm suggested by Sun et al. [24]. For warping we use thin-plate splines [4].

4.3.3 Reconstruction

An arithmetical mean mask is calculated from the warped masks. This mean mask is warped back into the position of the original tiles.

4.4 Applying the edits

Our algorithm assigns an alpha value to every texel. This value will scarcely be exactly one or zero. So we will do a segmentation by thresholding. Aside from distortions the alpha-values may be seen as voting for the background or the foreground, so 0.5 is a good threshold. The segmentation mask is of course not suitable for editing as it will obviously lead to strong artefacts. So we will substitute all texels, which have at least one

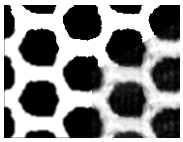


Figure 7: On the top the binary mask, on the right the original superposed mask and on the left the mixed mask.

corner-neighbour from the opposite component, by its alpha-value, so that the intuitive use of the word *boundary* and the definition given in section 3.2 coincide.

On the foreground component, editing can of course be done as e.g. described in the literature cited in section 2, but on the boundary it has to be taken under consideration, that for many edits it is necessary to know the exact decomposition $\rho_i = \alpha_i\phi_i + (1-\alpha_i)\psi_i$, which to find is an ill-posed problem.

5 EVALUATION

In the evaluation section we will show that our algorithm is capable of dealing with materials, which do not show the strong colour-contrasts, which are mostly necessary for matting and foreground-segmentation purposes.

5.1 Test set-up

To describe our test set-up we will start with a short description of the input data. Next, we will give a detailed overview over the competing algorithms to convey an idea where those algorithms run into problems. Of course the test set-up is strongly biased into the direction of our algorithm as both algorithms, AppProp and RepSnapping are by far more general. But we did not find a more fitting approach in literature.

5.1.1 Input data

The materials we use in this paper have been acquired with an enhanced version of the *linear light source reflectometer* (*LLSR*), introduced by Gardner et al. [8]. This new system has been developed by Meseth et al. [19] and is capable of measuring anisotropic reflectance distributions.

Additionally to the reflectance properties (equation 1), LLSR has to estimate a surface normal **n**. All values have been stored as 16 bit integer values. One texel represents a surface of roughly $1/4$ mm^2.

We use two different materials for the comparison: the grey mesh material which we have used to demonstrate the single steps of the algorithm and a structured steel material (see figure 8).

The grey mesh material is nearly uni coloured. It is particularly difficult to derive a near regular structure because it contains a strong bulge and the material normals do not convey much information.

While it is really simple, to derive the regular structure from the structured steel material, the only visible difference between foreground and background is a slightly less isotropic distribution of the noise. The metal material does not have a diffuse colour channel so we have to use R_s, instead.

5.1.2 Comparison with other algorihtms

Our algorithm combines techniques from the field of material manipulation with techniques from the field of repetition finding in images. Thus for comparison we have chosen one outstanding algorithm from each of those field. For the task of segmenting repetitions in images we decided for the RepSnapping algorithm [12], published in 2011 by Huang et al. And to cover the field of SVBRDF-editing we will compare against AppProp [2], published by An and Pellacini in 2008. Moreover, we compare those results with the segmentation of the SVM from step 4.1.

AppProp

The authors use a low rank approximation of the full appearance adjacency matrix and minimize the following functional:

$$\sum_{i,k} w_k z_{ik}(e_i - g_k)^2 + \lambda \sum_{i,j} z_{ij}(e_i - e_j)^2$$

with

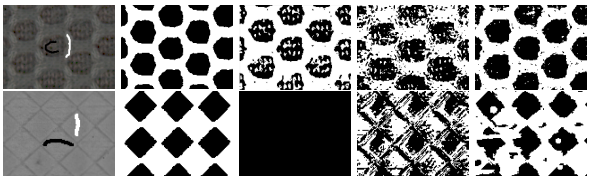
$$z_{ij} := \exp(-||f_i - f_j||^2/\sigma_a)\exp(-||x_i - x_j||^2/\sigma_s).$$

Where i and j go over all texel in the texture, k goes over all texels in the stroke input, w are weights, e is the edit and therefore the solution of the optimization problem, g is the stroke-input and therefore the right hand of the optimization problem, x is the position of the texel, λ the weight of the smoothing term and f is a texel-dependent appearance term. The resulting equation system is roughly solved by a low-rank approximation. The appearance comparison of AppProp is not limited to three dimensions or a single texel, so we can apply it to our descriptor (section 4.1.1).

The spatial parameter σ_s is not interesting in our setting, but to find a reasonable value for σ_a is difficult for our high dimensional descriptor and has to be done in a preprocessing step for every material separately. This is not surprising because the term

$$\exp(-||x_i - x_j||^2/\sigma_s) = \prod_k \frac{1}{e^{(x_i^k - x_j^k)^2/\sigma_s}}$$

consists of 450 factors in our case and has therefore the inclination to explode or to collapse beyond numerical accuracy. λ controls the consistency of the edit and had not much influence. We set λ and w_k to one. Thresholding has been done manually, in order, to get the best possible segmentation.



(a) Input (b) PushPin (c) RepSnap (d) AppProp (e) SVM

Figure 8: On the left a patch from the original image with the stroke input the second image shows the mask generated by RepSnapping. The third mask is the result of AppProp and the last mask is our result. The first row shows the grey mesh material the second row shows a metal.

RepSnapping

RepSnapping has been published by Huang et al. in 2011 [12], and is based on the idea of co-segmentation [11]. It is specialized to cutting out repeated elements in natural images. The algorithm solves the energy functional:

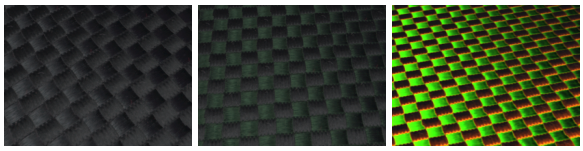
$$E(e) := \sum_i D_i(e_i) + \sum_{i<j} V_{i,j}(e_i,e_j) + \sum_{i,j\in H} U_{i,j\in Nbh}(e_i,e_j)$$

by the use of graph cuts [13]. Here D_i describes the probability that $e_i \in \mathcal{F}$ and is given as a normalized set distance to a clustering (H) of the foreground. $D_i(e_i=1) = \frac{\min_{k\in\mathcal{H}(F)}||f_i-f_k||}{\min_{k\in\mathcal{H}(F)}||f_i-f_k||+\min_{k\in\mathcal{H}(B)}||f_i-f_k||}$ with the appearance function f and $D_i(e_i=0) = 1-D_i(e_i=1)$. $V_{i,j} = \lambda|e_i-e_j|\exp(-\beta||f_i-f_j||^2)$ is a smoothing term and goes over all adjacent pixel pairs. $U_{j,j} = \mu|e_i-e_j|\exp(-\beta\gamma(i,j)^2)$ assures that pixels with similar appearance are treated similar. The main idea is that the neighbourhood graph is extended by the neighbourhood-system Nbh which contains edges between the pixels i and j iff $\gamma(i,j) < \varepsilon$, where γ is a correlation based similarity measure, described in [11].

We applied RepSnapping with the parameters given in [12], namely: $\mu = 10$, $\beta = 0.1$, $\lambda = 2$ and $\varepsilon = 4$. RepSnapping might easily be extended to the high-dimensional descriptor used in our algorithm but it would suffer from the same stability issues as AppProp.

5.2 The results

In figure 8 we present the comparison of the image segmentation step. You can see that our algorithm delivers artefact-free masks for both materials (8.b). The other three algorithms are more successful on the grey mesh material than on the metal material. An interesting result is, that the raw SVM delivers the second best results. We see the main reason in the descriptors: AppProp is numerically overcharged with the big number of descriptors, which results in this big amount of noise, and RepSnapping uses a correlation based approach to describe texel neighbourhoods. Autocorrelating the R_s-channel of the metal material reveals that the surface does not have enough structure to provide significant correlation results. Together with the fact that R_s is unicoloured, this explains, why RepSnapping fails completely.



(a) Original (b) Changed R_d (c) Changed R_s

Figure 9: On the left the original material in the middle a rather subtle edit of R_d, on the right a more noticeable manipulation of R_s.

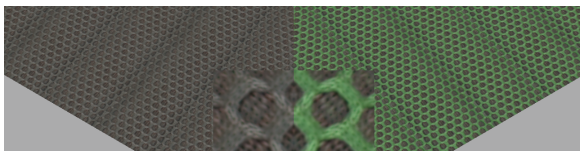


Figure 10: In the close-up of the edit of the grey mesh material one may see that the editing boundary coincides exactly with the perceived boundary of the foreground material.

Figure 11: A shiny material. The left image shows the unedited material. In the right image the background has been changed: R_s has been changed from yellow to green.

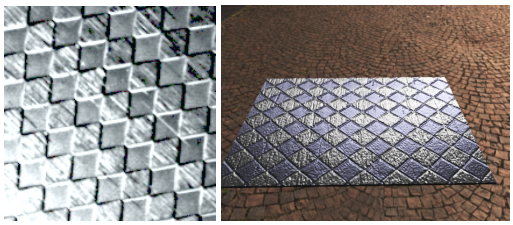


Figure 12: On the left the metal material, on the right a rendering of the edited material.

5.3 Editing examples

In this section we want to present the resulting edits on four different materials (figure 9 - 12).

5.4 Time Requirement

The bottleneck of the algorithm was to calculate the optical flow on all tiles (paragraph 4.3.2). For the grey-mesh material we had about 180 tiles. Calculating the optical flow on one tile (~80x40 texel) took about 2.2 s, which sums up to about 7 min. Depending on the

number of training samples, the SVM classification step took between 10 s and 3 min (paragraph 4.1.2). MSBP (paragraph 4.2.1) ran for about 45 s. Warping a tile with tps took about 0.03 s. Finding a principal tile took less than a second. So the overall processing time lay between 8 and 12 minutes.

For comparison: RepSnapping took 3 s, SVM took 10 s and AppProp took 40 s.

5.5 System

Computations have been done on an i5-2500 with a clock rate of 3.3 G/s and 8 GB RAM.

6 CONCLUSIONS AND FUTURE WORK

In this paper we demonstrated an algorithm to solve the task of extracting a repeating foreground pattern from a high dimensional reflectance representation map in a way, which is robust and reliable enough, to make additional optical debugging steps unnecessary. While the task is relatively simple on suitable materials, we could show, that the competing state of the art algorithms failed for difficult material probes. Our algorithm permits high quality segmentation and editing on complex materials.

Yet our algorithm is too slow for productive and industrial use. But many steps of the algorithm may be parallelized, particularly with respect to computations on the tiling, so that efficiency and responsiveness may be improved drastically.

7 REFERENCES

[1] Xiaobo An, Xin Tong, Jonathan D. Denning, and Fabio Pellacini. Appwarp: retargeting measured materials by appearance-space warping. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, pages 147:1–147:10, New York, NY, USA, 2011. ACM.

[2] Pellacini F. An X. Appprop: all-pairs appearance-space edit propagation. *ACM Transactions on Graphics*, 27(3):1–9, 2008.

[3] Michael Ashikhmin and Peter Shirley. An anisotropic phong brdf model. *Journal of graphics tools*, 5(2):25–32, 2000.

[4] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585, 1989.

[5] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[7] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 151–157, 1997.

[8] Andrew Gardner, Chris Tchou, Tim Hawkins, and Paul Debevec. Linear light source reflectometry. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 749–758. ACM, 2003. No. 3.

[9] Branko Grünbaum and Geoffrey Colin Shephard. *Tilings and patterns*. Freeman, 1987.

[10] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[11] Dorit S Hochbaum and Vikas Singh. An efficient algorithm for co-segmentation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 269–276. IEEE, 2009.

[12] Hua Huang, Lei Zhang, and Hong-Chao Zhang. Repsnapping: efficient image cutout for repeated scene elements. In *Computer Graphics Forum*, volume 30, pages 2059–2066. Wiley Online Library, 2011. No. 7.

[13] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *IEEE transactions on pattern analysis and machine intelligence*, 26(2):147–159, 2004.

[14] Murat Kurt, László Szirmay-Kalos, and Jaroslav Křivánek. An anisotropic brdf model for fitting and monte carlo rendering. *ACM SIGGRAPH Computer Graphics*, 44(1):3, 2010.

[15] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[16] Hendrik PA Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatially varying materials. In *Rendering Techniques 2001*, pages 103–114. Springer, 2001.

[17] Yanxi Liu, Robert Collins, and Yanghai Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):354 – 371, March 2004.

[18] Timothy Masters. *Practical neural network recipes in C++*. Morgan Kaufmann, 1993.

[19] Jan Meseth, Shawn Hempel, Andrea Weidlich,

Lynn Fyffe, Graham Fyffe, Craig Miller, Paul Carroll, and Paul Debevec. Improved linear light source material reflectance scanning. In *ACM SIGGRAPH 2012 Posters*, page 42. ACM, 2012.

[20] Minwoo Park, Kyle Brocklehurst, Robert T Collins, and Yanxi Liu. Deformed lattice detection in real-world images using mean-shift belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1804–1816, 2009.

[21] Fabio Pellacini and Jason Lawrence. Appwand: editing measured materials using appearance-driven optimization. In *ACM Transactions on Graphics (TOG)*, volume 26, page 54. ACM, 2007. No. 3.

[22] Szymon M Rusinkiewicz. A new change of variables for efficient brdf representation. In *Rendering techniques 98*, pages 11–22. Springer, 1998.

[23] Thorsten-Walther Schmidt, Fabio Pellacini, Derek Nowrouzezahrai, Wojciech Jarosz, and Carsten Dachsbacher. State of the art in artistic editing of appearance, lighting and material. In *Computer Graphics Forum*. Wiley Online Library, 2015.

[24] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE, 2010.

[25] Jiaping Wang, Xin Tong, Stephen Lin, Minghao Pan, Chao Wang, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Appearance manifolds for modeling time-variant appearance of materials. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 754–761, New York, NY, USA, 2006. ACM.

[26] Hongzhi Wu, Julie Dorsey, and Holly Rushmeier. A sparse parametric mixture model for btf compression, editing and rendering. In *Computer Graphics Forum*, volume 30, pages 465–473. Wiley Online Library, 2011. No. 2.

[27] Kun Xu, Jiaping Wang, Xin Tong, Shi-Min Hu, and Baining Guo. Edit propagation on bidirectional texture functions. In *Computer Graphics Forum*, volume 28, pages 1871–1877. Wiley Online Library, 2009. No. 7.