Supplementary Material for "MATch: Differentiable Material Graphs for Procedural Material Capture"

LIANG SHI, MIT CSAIL
BEICHEN LI, MIT CSAIL
MILOŠ HAŠAN, Adobe Research
KALYAN SUNKAVALLI, Adobe Research
TAMY BOUBEKEUR, Adobe
RADOMIR MECH, Adobe Research
WOJCIECH MATUSIK, MIT CSAIL

ACM Reference Format:
Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. 2020. Supplementary Material for "MATch: Differentiable Material Graphs for Procedural Material Capture" . *ACM Trans. Graph.* 39, 6, Article 196 (December 2020), 7 pages. https://doi.org/10.1145/3414685.3417781

1 DIFFMAT EVALUATION

With its large node arsenal and auto-translation tool, DiffMat can handle material graphs with production-size complexity and connectivity. To show its versatility, we collected a set of 88 procedural material graphs from the Substance Share and Substance Source exchange platform. This set consists of a wide variety of metallic and dielectric materials, including but not limited to oxidized/galvanized/rusted/polished/weaved/battered/sanded metal, skin, leather, paint, fabric, wood, ceramic, concrete, brick, travertine, stone, granite, and marble. These material graphs contain an average of 65 nodes and 335 continuous node parameters, with the largest graph containing more than 200 nodes and 2650 parameters (see Tab. 1).

DiffMat automatically translated all these graphs without any manual intervention. The graph parsing and program generation typically takes less than a second, while the time taken for precomputed bitmaps depends on the output resolution and the number of input generator nodes. Figure 4 in the main submission shows an image gallery of all our 88 materials rendered with a light source and a camera co-located on top of the material center. The rendering layer uses the Cook-Torrance microfacet BRDF model with GGX normal distribution [Walter et al. 2007].

To maximally leverage the Substance ecosystem, DiffMat is designed to accurately match its output to the Substance Designer's output. As a result, users can export the optimized materials and apply them to 3D objects using tools like Substance Painter or perform additional fine-tuning. In fact, our implementations have an exact per-pixel match for all atomic filter nodes except for minor differences in the *emboss* node. This high fidelity extends to non-atomic filter nodes as they are built from the atomic nodes. In Figure S1, we demonstrate this accuracy by comparing DiffMat's outputs against Substance Designer's output. Note that since the nodes (and graphs) are resolution-independent, this accuracy is retained across different resolutions.

Finally, we report DiffMat's performance by evaluating its runtime speed and memory consumption for the forward evaluation. Table 1 lists the results benchmarked on 8 materials at resolution of 1024 × 1024 pixel. We use an NVIDIA Tesla V100 GPU and Intel Xeon E5-2699v4 CPU (base frequency 2.2GHz, 55MB cache) for all reported numbers. The forward-backward evaluation costs ∼2× runtime and memory. We note that even for node graphs with thousands of parameters, a gaming-level GPU can easily hold the graph with sufficient space left.

Table 1. DiffMat performance on various translated procedural material graphs. The index refers to the location (row and column) of material in Fig. 4 in the main paper. Graphs marked by the * symbol include the computation of alpha channel while others do not. We report the memory reserved by PyTorch during forward evaluation and the total memory cost on GPU. The overhead is due to the storage of other system resources which is approximately constant across graphs. We note that the runtime is not directly proportional to the number of nodes or parameters as computational cost can vary significantly among nodes.

Graph Name (index)	Nodes (Parameters)	Memory PyTorch/GPU	Runtime
ceramic_tiles (2,1)	42 (128)	450MiB / 1449MiB	0.055s
zombie_skin* (8,11)	124 (383)	1108MiB / 2099MiB	0.166s
silk_red (6,9)	51 (250)	506MiB / 1505MiB	0.212s
copper_oxydized (3,4)	114 (307)	938MiB / 1939MiB	0.246s
bricks_gray* (1,8)	201 (923)	1704MiB / 2701MiB	0.318s
wood_cherry* (8,4)	124 (925)	1038MiB / 2027MiB	0.389s
wooden_planks (8,10)	195 (2224)	1438MiB / 2511MiB	1.291s
travertine_persian (8,1)	96 (511)	822MiB / 1823MiB	1.295s

Authors' addresses: Liang Shi, MIT CSAIL, liangs@mit.edu; Beichen Li, MIT CSAIL, beichen@mit.edu; Miloš Hašan, Adobe Research, mihasan@adobe.com; Kalyan Sunkavalli, Adobe Research, sunkaval@adobe.com; Tamy Boubekeur, Adobe, boubek@adobe.com; Radomir Mech, Adobe Research, rmech@adobe.com; Wojciech Matusik, MIT CSAIL, wojciech@csail.mit.edu.

0730-0301/2020/12-ART196
https://doi.org/10.1145/3414685.3417781

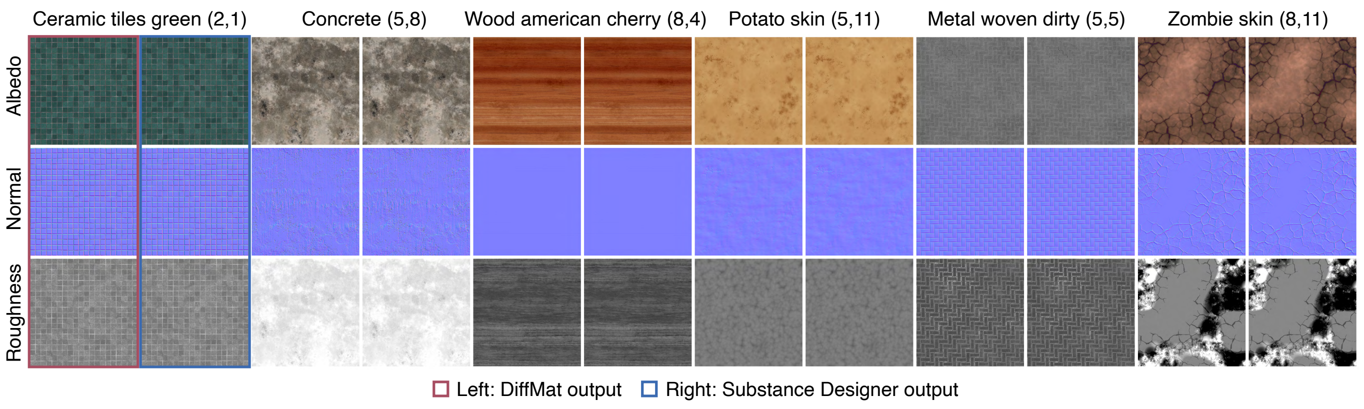

Fig. S1. Comparison of SVBRDF maps produced by DiffMat and Substance Designer. The indices after the graph name refer to the location of corresponding rendered images in Figure 4 in the main paper. The metallicity is omitted for the selected examples as it is either uniform black or white. All maps are either per-pixel matched or with negligible visual difference. Readers are encouraged to zoom in to exam and compare image details.

2 NODE DESCRIPTIONS

DiffMat implements 21 atomic nodes and 110 non-atomic nodes in total. This section provides a complete list of their functionalities.

2.1 Atomic Nodes

1. **Blend**: blends or mixes together two separate inputs with a user-defined blend mode and an optional opacity mask.
2. **Blur**: performs a "box-blur" operation by averaging the values of pixels over a set distance.
3. **Channel Shuffle**: takes two inputs and returns an output where any of the red, green, blue, and alpha channels are swapped or set to any of the channels from the input.
4. **Curve**: remaps the tonality of an input using Bézier curves.
5. **Directional Blur**: performs a simple motion-blur like operation on an input along a user-defined angle.
6. **Directional Warp**: warps an input in a user-set direction, multiplied by a user-set intensity map.
7. **Distance**: creates an outward linear fade from any pixels in the input over 0.5 grayscale value.
8. **Emboss**: performs a simple 2D shading based on two inputs, simulating light falling on a surface with height and depth variation.
9. **FX-Map**: replicates and subdivides an image or pattern input over and over again, and control the distribution of each pattern via parameters and value functions.
10. **Gradient (Dynamic)**: remaps a grayscale input to a new grayscale or color ramp where color keys are provided through an additional input.
11. **Gradient Map**: remaps a grayscale input to a custom-set grayscale or color ramp.
12. **Grayscale Conversion**: converts a color input to a grayscale output as a weighted average over RGBA channels.
13. **HSL**: adjusts the hue, saturation and lightness of a color input.
14. **Levels**: Remap the tones of an input according to input and output remap factors, including gamma correction.
15. **Normal**: converts an input grayscale map to a tangent-space normal map output.
16. **Pixel Processor**: executes a custom function for every pixel that is returned as output, on one or more optional inputs.
17. **Sharpen**: performs a sharpening operation on an input, similar to unsharp masking.
18. **Transformation 2D**: performs an affine transformation on an input, optionally including mipmapping.
19. **Uniform Color**: returns a solid, user-defined color or value.
20. **Value Processor**: executes a custom function on optional inputs for one or more values as output.
21. **Warp**: transforms a base input by warping or pushing pixels as specified by a gradient input.

2.2 Non-atomic Nodes

1. **Alpha Merge**: adds an alpha channel to an input.
2. **Alpha Split**: strips the alpha channel out of an input.
3. **Ambient Occlusion**: takes a height map as input and generates an ambient occlusion map.
4. **Anisotropic Blur**: applies a high quality motion blur effect to the input.
5. **Anisotropic Blur (Grayscale)**: see 'Anisotropic Blur'.
6. **Auto Levels**: automatically adjusts the levels of the input to use the full range from black to white.
7. **Bevel**: applies an edge-beveling effect to an input grayscale height map and returns beveled height and normal maps.
8. **BaseColor / Metallic / Roughness Converter**: converts basecolor, metallic and roughness maps to different PBR model outputs, such as Specular/Glossiness model.
9. **Blur HQ**: performs a high-quality Gaussian blur on the input.
10. **Blur HQ (Grayscale)**: see 'Blur HQ'.
11. **Cartesian to Polar**: converts an input in Cartesian coordinates to polar coordinates.
12. **Cartesian to Polar (Grayscale)**: see 'Cartesian to Polar'.
13. **Channel Mixer**: mixes, swaps and blends RGB channels.
14. **Chrominance Extract**: extracts the chrominance value from the input.

15. **Clamp**: clamps input values to user-defined limits.
16. **Clamp (Grayscale)**: see 'Clamp'.
17. **Color Blend**: performs blending by preserving the luminance of the background while adopting the hue and chrominance of the foreground.
18. **Color Burn**: performs a color burn blend between foreground and background.
19. **Color Dodge**: performs a color dodge blend between foreground and background.
20. **Color to Mask**: turns a selected color range into a black-and-white mask.
21. **Contrast/Luminosity**: adjusts the contrast and brightness of the input.
22. **Contrast/Luminosity (Grayscale)**: see 'Contrast/Luminosity'.
23. **Convert to Linear**: converts an sRGB color space input to linear color space.
24. **Convert to Linear (Grayscale)**: see 'Convert to Linear'.
25. **Convert to sRGB**: converts a linear input to sRGB color space.
26. **Convert to sRGB (Grayscale)**: see 'Convert to sRGB'.
27. **Curvature**: performs a simple, harsh single-pass curvature conversion to the input normal map.
28. **Curvature Smooth**: performs a smooth multi-pass curvature conversion to the input normal map.
29. **Curvature Sobel**: performs a single-pass curvature conversion to input normal map with Sobel sampling.
30. **Difference**: performs a difference blend between foreground and background.
31. **Dissolve**: blends two inputs together with white noise as the mask for transition.
32. **Edge Detect**: detects contrast in a grayscale input, then creates a black and white mask highlighting the contrast.
33. **Emboss with Gloss**: performs an embossing effect with added gloss on a color and height input.
34. **Glow**: performs a glowing effect on an input.
35. **Glow (Grayscale)**: see 'Glow'.
36. **Facing Normal**: takes a normal map as an entry image and produces a grayscale image in which the value corresponds to how much the normal vectors are facing the viewer.
37. **Grayscale Conversion Advanced**: converts a color input to a grayscale output in several preset modes.
38. **Height Map Frequencies Mapper**: separates a height map's frequencies into two separate maps: one with large-scale differences and one with small-scale differences.
39. **Height Normal Blender**: converts and blends a grayscale height map onto a normal map.
40. **Height to Normal World Units**: converts a height map input to a normal map while making use of real-world units during the conversion.
41. **Highpass**: applies a highpass filter to the input.
42. **Highpass (Grayscale)**: see 'Highpass'.
43. **Histogram Range**: reduces or moves the histogram range of a grayscale input.
44. **Histogram Scan**: remaps the contrast and brightness of an input grayscale image based on its histogram.
45. **Histogram Scan Non-Uniform**: performs a histogram scan filter with additional controls and input to drive the effect on a per-pixel level.
46. **Histogram Select**: remaps a grayscale input by setting a grayscale value position with a fade range around it, whose contrast can be adjusted to make the range sharper.
47. **Histogram Shift**: shifts the whole range of the image, wrapping around when reaching range limits.
48. **Invert**: inverts input colors.
49. **Invert (Grayscale)**: see 'Invert'.
50. **Linear Burn**: performs a linear burn blend between foreground and background.
51. **Luminance Highpass**: cancels out lighting information by applying a highpass filter to the input's luminance value.
52. **Luminosity Blend**: performs a luminosity blend between foreground and background.
53. **Make It Tile Patch**: repeats and stamps an input image patch around with controlled randomness, generating a tiling output.
54. **Make It Tile Patch (Grayscale)**: see 'Make It Tile Patch'.
55. **Make It Tile Photo**: fixes the edges of an input which might not tile due to non-continuous edges, producing a tiling version.
56. **Make It Tile Photo (Grayscale)**: see 'Make It Tile Photo'.
57. **Mirror**: mirrors the input image over a user-defined axis, and from a chosen side.
58. **Mirror (Grayscale)**: see 'Mirror'.
59. **Mosaic**: applies a facetization effect to an input gradient map through multi-pass warping.
60. **Mosaic (Grayscale)**: see 'Mosaic'.
61. **Multi Directional Warp**: applies directional warp multiple times in opposite directions while the displaced texture stays in place.
62. **Multi Directional Warp (Grayscale)**: see 'Multi Directional Warp'.
63. **Multi Switch**: passes through one of the inputs defined by the selection parameter.
64. **Multi Switch (Grayscale)**: see 'Multi Switch'.
65. **Non-Square Transform**: similar to 'Transform 2D', but detects non-square ratios and can transform square input images onto a non-square canvas.
66. **Non-Square Transform (Grayscale)**: see 'Non-Square Transform'.
67. **Non-Uniform Blur**: performs a high-quality blur on the input where the intensity is driven by an input mask, optionally supporting anisotropy and assymetry.
68. **Non-Uniform Blur (Grayscale)**: see 'Non-Uniform Blur'.
69. **Normal Blend**: blends two normal maps together with an optional mask, while making sure all values stay normalized.
70. **Normal Color**: returns a uniform color in the normal map color space, according to normal direction and slope angle.
71. **Normal Combine**: blends two normal maps while combining their details in a mathematically correct way.
72. **Normal Invert**: inverts any or all channels of a normal map.
73. **Normal Normalize**: performs a mathematical vector normalization operation on every single pixel in the input normal map.
74. **Normal Sobel**: converts a heightmap input to a normal map output, using Sobel sampling.

75. **Normal to Height**: converts a tangent-space normal map back into a height map.
76. **Normal Vector Rotation**: rotates all vectors of an input normal map in tangent space.
77. **Polar to Cartesian**: converts an input in polar coordinates to Cartesian coordinates.
78. **Polar to Cartesian (Grayscale)**: see 'Polar to Cartesian'.
79. **Pow**: powers the input by a user-specified exponent.
80. **Pow (Grayscale)**: see 'Pow'.
81. **Pre-Multiplied to Straight**: removes the pre-multiplied alpha from RGB on the input.
82. **Quad Transform**: performs transformation of a quad shape by specifying its corner points.
83. **Quad Transform (Grayscale)**: see 'Quad Transform'.
84. **Quantize**: approximates the color range of an input to a predefined number of values.
85. **Quantize (Grayscale)**: see 'Quantize'.
86. **Replace Color**: adjusts an color input by hue shifting a source color towards a target color.
87. **Replace Color Range**: replaces the source Color by the target Color, with additional controls.
88. **RGBA Merge**: packs four separate grayscale inputs into a single color output.
89. **RGBA Split**: splits an input into its respective red, green, blue, and alpha channels.
90. **Safe Transform**: scales, rotates, or offsets the input without breaking tiling or losing details.
91. **Safe Transform (Grayscale)**: see 'Safe Transform'.
92. **Shape Drop Shadow**: Performs the "drop shadow" effect on an input black-and-white mask or an input with transparency.
93. **Shape Drop Shadow (Grayscale)**: see 'Shape Drop Shadow'.
94. **Shape Glow**: creates a soft glow around an input mask or an input with transparency.
95. **Shape Glow (Grayscale)**: see 'Shape Glow'.
96. **Skew**: skews an input image.
97. **Skew (Grayscale)**: see 'Skew'.
98. **Slope Blur**: performs an advanced, high-quality blur where the direction is driven by a grayscale slope map.
99. **Slope Blur (Grayscale)**: see 'Slope Blur'.
100. **Straight to Pre-Multiplied**: converts a straight alpha to pre-multiplied by multiplying a color into alpha-blended pixels.
101. **Swirl**: transforms an input image by warping it in a swirling direction.
102. **Swirl (Grayscale)**: see 'Swirl'.
103. **Switch**: returns one or the other input based on choice setting.
104. **Switch (Grayscale)**: see 'Switch'.
105. **Trapezoid Transform**: modifies the input in a trapezoid warping manner.
106. **Trapezoid Transform (Grayscale)**: see 'Trapezoid Transform'.
107. **Vector Morph**: distorts an input image smoothly and progressively by a color input as a vector map.
108. **Vector Morph (Grayscale)**: see 'Vector Morph'.
109. **Vector Warp**: applies an advanced warping effect to the input, driven by a vector map.
110. **Vector Warp (Grayscale)**: see 'Vector Warp'.

3 OPTIMIZED MAPS FOR SYNTHETIC MATERIALS

Figure S2 visualize the optimized material maps for 4 additional synthetic materials. Each example is presented as a septuple (from left to right): rendering of the default material, rendering of the target material, rendering of the optimized material, and the optimized albedo, normal, roughness, and metallic maps. The optimization was performed directly from the default parameters under the exposed parameters preserved mode if the exposed parameters are defined or full parameters mode otherwise.

4 OPTIMIZED MAPS FOR REAL-WORLD MATERIALS

Figure S3 and Figure S4 visualize all 45 optimized real-world materials. Each example is presented as a sextuplet (from left to right): the user-input photograph, rendering of the optimized procedural material, and the generated albedo, normal, roughness, and metallic maps.

REFERENCES

Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. 2007. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*. Eurographics Association, 195–206.

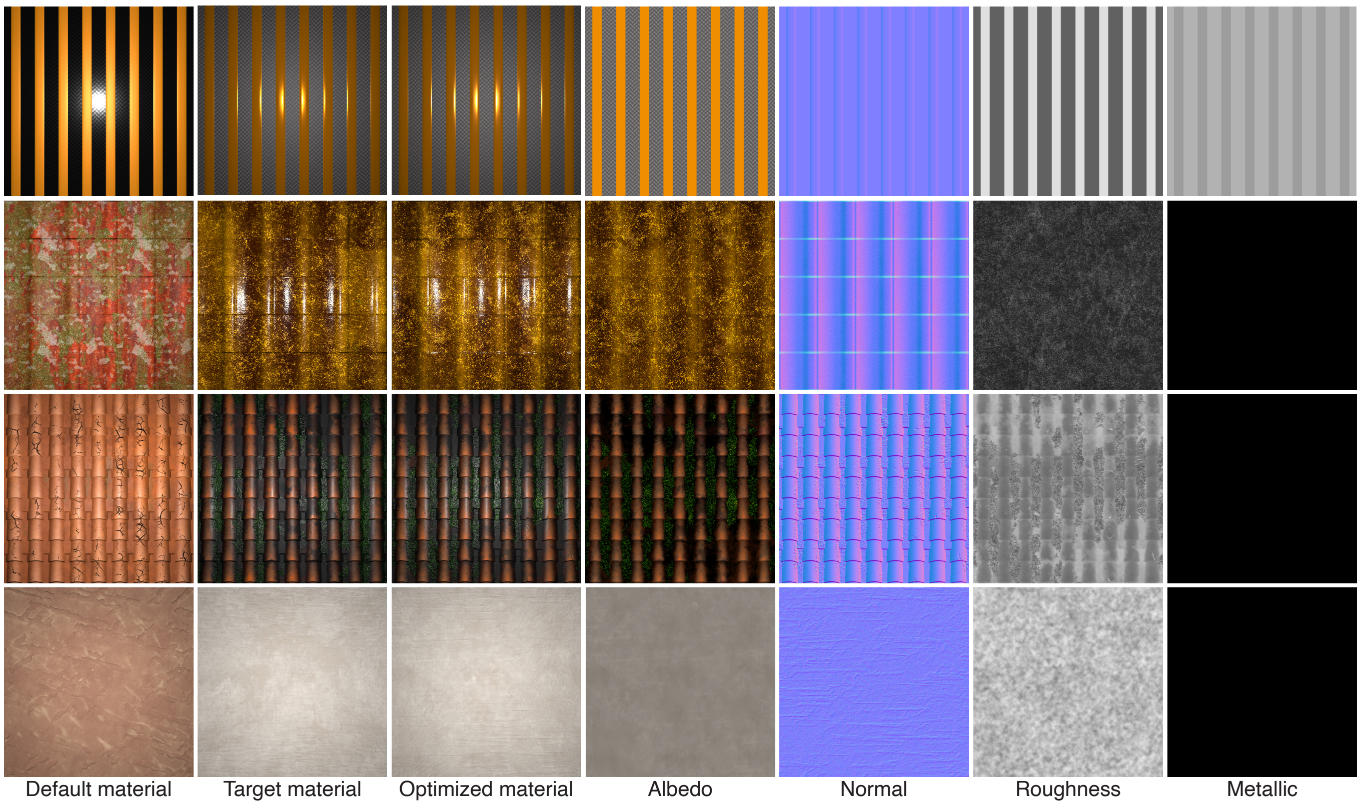

Fig. S2. Material map visualization of the optimized synthetic material images.

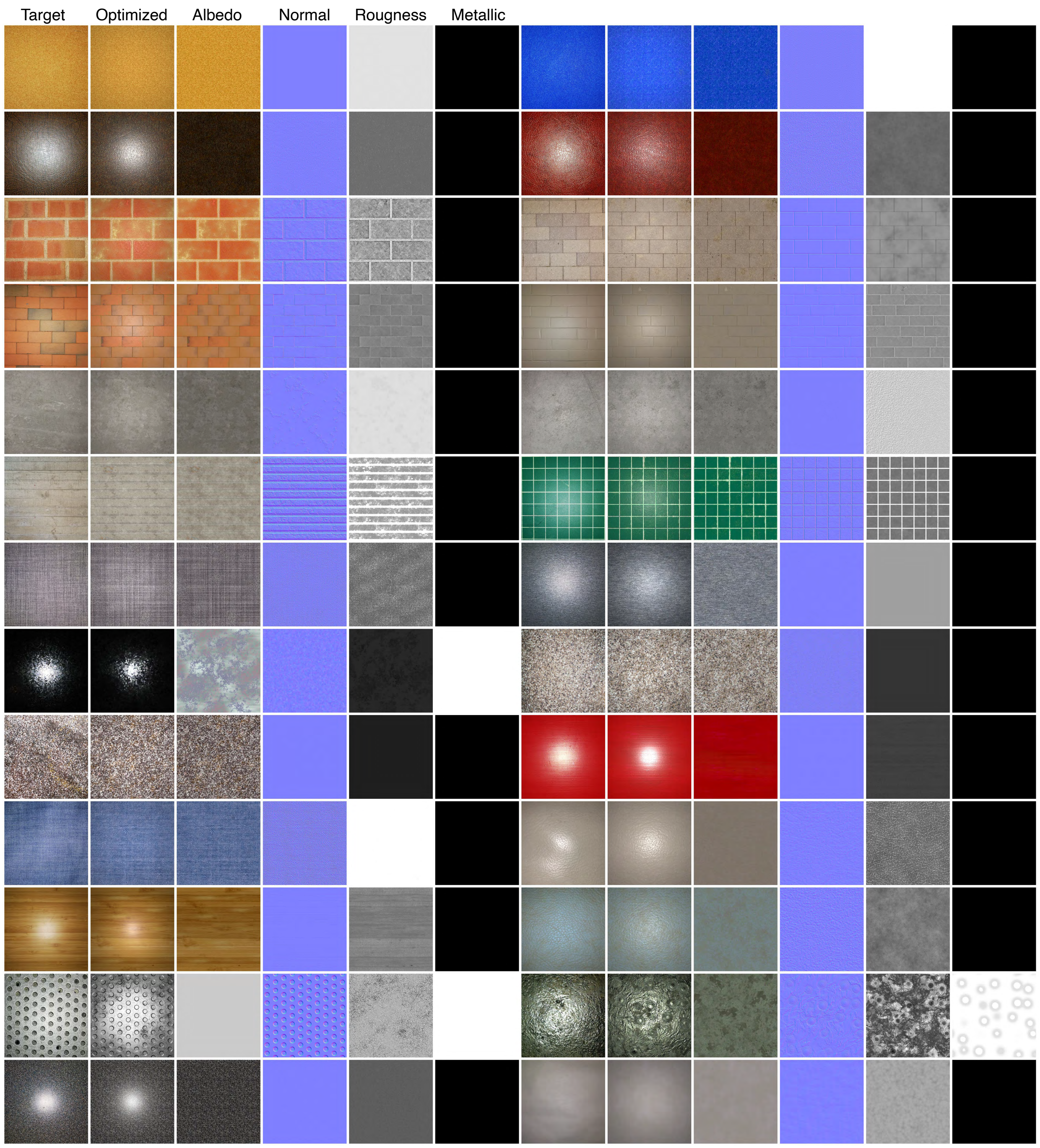

Fig. S3. Material map visualization of the optimized real-world material images, Part 1.

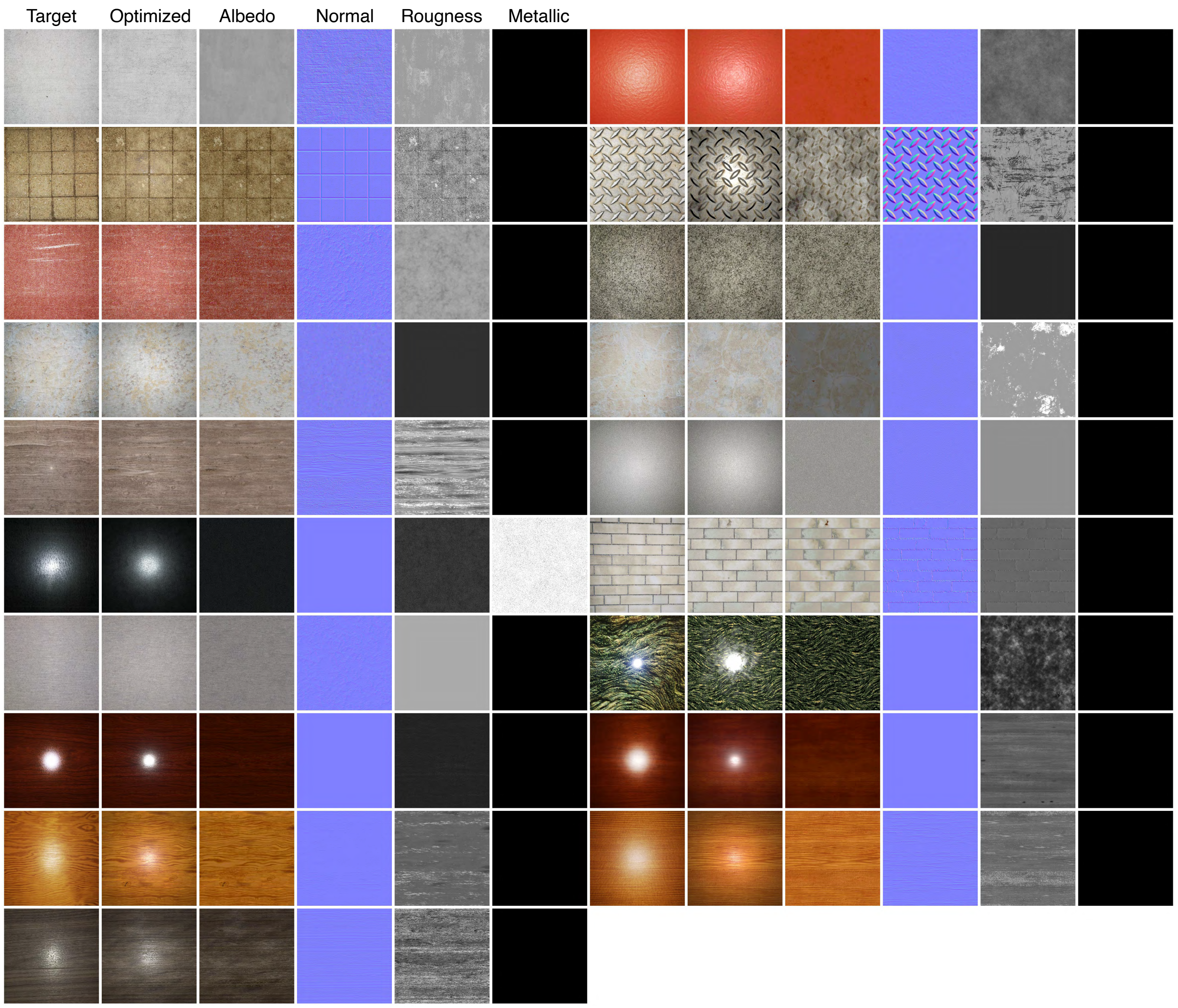

Fig. S4. Material map visualization of the optimized real-world material images, Part 2.