Computational Aesthetics in Graphics, Visualization, and Imaging (2009)
O. Deussen and P. Hall (Editors)

Adding lighting and viewing effects to digital images

Cindy Grimm †

Abstract

Real paintings are not truly flat but change subtly with variations in viewing direction. The pigments and painting layers also interact with the lighting environment, producing changes that range from subtle to quite dramatic. These effects are lacking in digital images. This paper describes a system that allows an artist to introduce, and control, a variety of lighting and viewing effects, such as specular reflection and refraction, through the use of additional images.

Keywords: 3D Painting, image rendering

Real paintings are not truly flat, static images. Because of transparency in the binding media and pigments, the physical placement of the paint on the canvas, and even the canvas (or paper) itself, paintings change when viewed from different directions. The paint interactions with light are also more interesting than simply changing the luminance because of subtle variations in the surface normal and interactions between the pigments in the different paint layers.

One could, presumably, model all of these interactions in a physically correct manner. This approach, however, would be both prohibitively expensive computationally and difficult for the artist to control. Our approach is, instead, to begin with existing work in photo-realistic rendering, but modify it to make the effects more accessible and controllable. Our goal is not just to provide the artist with tools to add these effects to their images (see Figure 1), but to make tools that are interesting in and of themselves.

With the advent of programmable graphics hardware there has been an explosion of real-time, complex lighting effects, including modeling object inter-reflection using irradiance maps, shadow maps, refraction, and sub-surface scattering. We take a small slice of these to use as inspiration: These techniques simplify greatly because we are working with a single image. The challenge is not in implementing them, but in choosing how to *use* them.

Our criteria for the inclusion of a lighting effect are the following:

- Can the artist provide the input to the effect in a way that is reasonably intuitive, and does not require a deep understanding of the underlying calculations?
- Can we ensure that, given reasonable inputs, the result is not overly dark or washed out?
- Can the effect range from subtle to extreme?

The effects we support are 2.5D shading of the image, environment maps, shadow maps, and sub-surface scattering. Each of these has been modified to make it easier to use. Like other 2.5D approaches, we use a height field to add texture to the surface. This height field is not, however, generated automatically but is under the control of the artist.

It can be challenging to place lights so that a model is well-lit, with no area too dark or two bright. We avoid this problem by using the diffuse shading to select between a lit and unlit *light* color (see Figure 2(j)). Instead of using material properties, we let the artist specify the specular highlight colors as another image. To prevent wash-out, we use the specular lighting component to scale between the diffuse and the specular images.

We adopt environment and irradiance maps (see Figure 2(k,l)) to use as an alternative lighting source; the artist can directly paint both if desired, or the system will automatically generate an irradiance map by integrating the environment map.

Sub-surface scattering and refraction model how light bounces between layers. We greatly simplify these techniques, providing the artist with explicit control over where the lower layer shows through, how much the image can shift due to refraction, and how much scattering is allowed. These controls are not global but are defined spatially over the image plane.

Non-photorealistic effects range from completely auto-

† email:cmg@cse.wustl.edu

DOI: 10.2312/COMPAESTH/COMPAESTH09/099-105

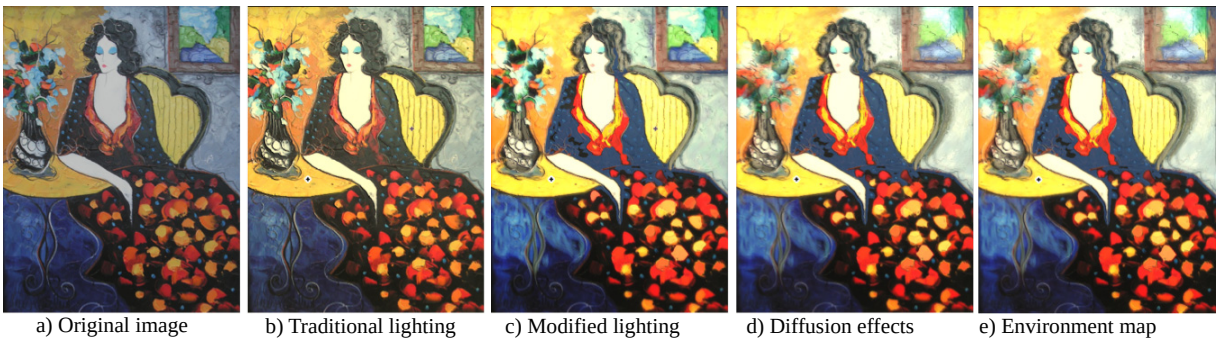

Figure 1: *Examples of adding increasingly more complex lighting effects to a digital photograph of a painting (original painting by Luis Cogley). (a) The original photograph of the painting. (b) Adding Phong shading based on a height field (shown in Fig. 2d). (c) Adding warm-cool lights and specular lighting effects (additional input images used are shown in Figs. 2b and 2c). (d) Adding diffusion, refraction, and layer effects (additional input images used are shown in Figs. 2e-h). (e) Lighting with an environment map (shown in Figs. 2k and 2l). Image size: 796 by 1024.*

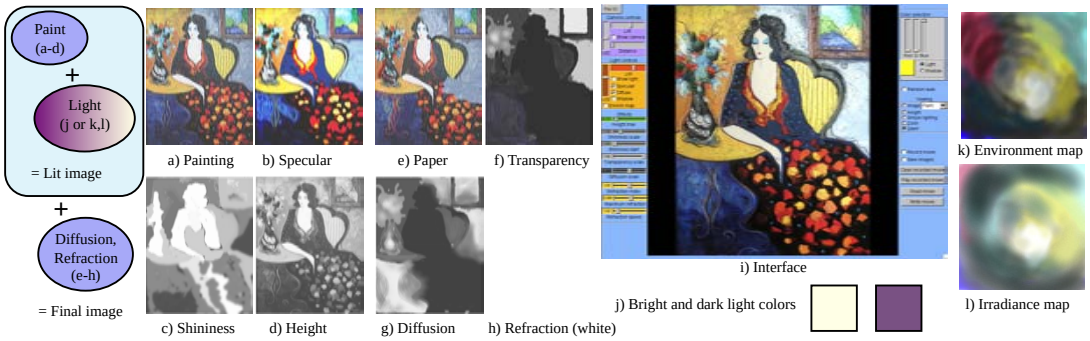

Figure 2: *Left: How the input images are combined to produce the final image. (a)-(d) Input images for shading. (e)-(h) Input images for sub-surface scattering. (i) The user interface. (j) Warm and cool light colors. (k,l) Environment map and irradiance map used for lighting.*

mated to artist-intensive. Our work falls more on the latter end, although reasonable results can be achieved by running simple filters over an existing image (see Figure 3).

Contributions: In summary, we have combined and modified several standard lighting effects to produce a non-photorealistic rendering system for digital images. The modifications were chosen to allow artist control and to produce reasonably lit images under a variety of lighting and viewing conditions.

1. Related work

Adding 2.5D to images is not new; this was originally implemented by Hertzmann [Her02] to add texture to automatically placed brush strokes. They generated a height field then used a simple bump-map approach with Phong shading to add brush stroke texture to the image. A similar lighting model, but with more accurate spectrum representation, was used in a fully interactive paint system which included modeling brush strokes, paint flow, and pigment interaction [BWL04]. Both of these systems were focused on re-creating the look and feel of a brush-stroked image. Our work is somewhat orthogonal to these systems in that we are not interested in re-creating a particular media. These systems could easily be used in conjunction with ours to automatically produce the primary image and height field.

A height field [TFFR07] or a depth map [SZKC06] can also be used to control stylization techniques based on shape information such as curvature. In this approach, the extra normal or depth information is used to guide placement of strokes, textures, and silhouette lines to better capture the 3D nature of the shape.

Our work is perhaps most closely aligned in spirit with the original warm-cool shading introduced by Gooch et. al. [GGSC98]. Here, the calculated shading is used to select from a selected set of color blends or color palettes [SMGG01]. This idea was extended to rendering

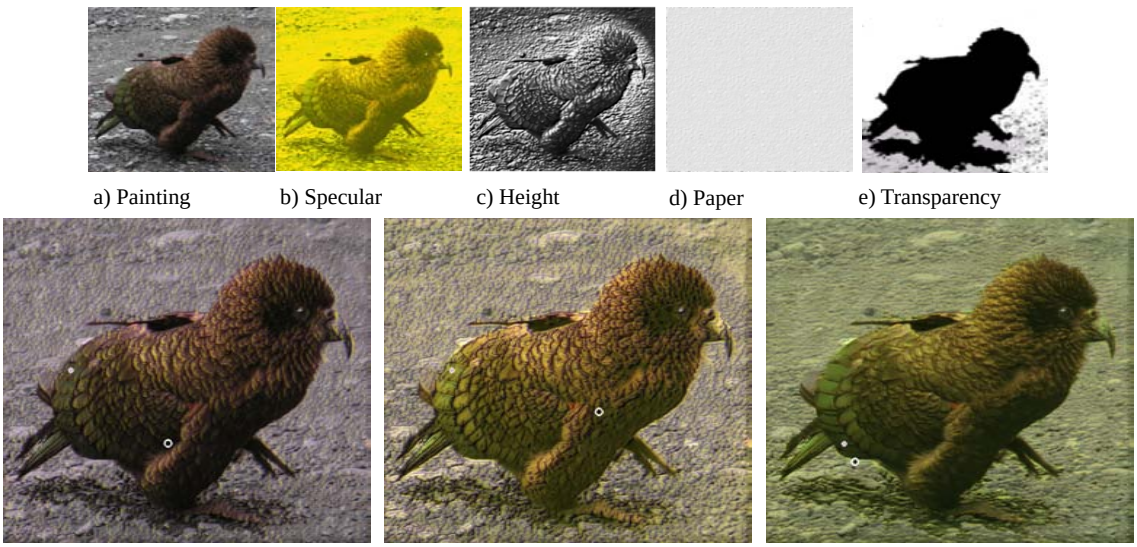

Figure 3: *(a)-(e) Input images. Specular image was made by color (un) balancing the original photograph. Height field was made by adding a light around the head and adding light-based texturing. Paper is a standard paper texture. The transparency image was made by flood filling the bird and the background using the "fill" tool. The circle in the image indicates the viewing direction. Image size: 906 by 798.*

textures based on shading as well [KTBG03]. A more artist-based approach was proposed in [GK07], where the artist painted additional texture maps that were blended in when the object was viewed from a similar viewing direction. Like these approaches, we use traditional shading as an input, only we use the shading to combine multiple artist-constructed images.

2. Specifying the effects

Figure 2 and table 3 summarize the artist's input to the rendering process. We chose images over a vector or widget approach because artists are use to working with images, and any higher-level set of interactions can always be reduced to an image.

In addition to the images the user interface has sliders to control the scale of the effects, the light and view directions, whether or not to include sub-surface scattering, the color of the "bright" and "dark" lights, and switching to environment maps (see Figure 2(i)) .

2.1. Shading

The artists begins by defining the primary image, the "painting". This image is, by default, copied to the two other full-color images, the "specular" image and the "paper" one. The specular image is used to define what color the specular highlights should be.

Next, the artist defines a grey-scale height field image. They can optionally specify a second grey-scale image which controls the size of the highlights (specular exponent). These, together with the "lit" and "unlit" light colors (or environment map) and lighting and viewing directions are enough to shade the image. Sudden changes in the specular exponent tend to emphasize small texture changes in the height image. Most of the height field images in this paper were made by converting the original image to grey-scale, then applying various image filters (gradient lighting, contrast enhancement, blurring).

Both the light and viewing directions relative to the painting are under the control of the user. In a traditional 3D rendering system changing the viewpoint would also change the angle the painting was viewed at. Here we just use the viewing direction in the lighting equation.

Motivation: If the height field is simply used to light the diffuse painting [Her02] then the resulting image has a very coherent texture, but it tends to have a plastic look (Figure 1(b)). The plastic look can be mitigated somewhat by adding material properties [BWL04] but this requires the artist to specify them for every color, which is difficult and not particularly intuitive. An alternative is to use the calculated shade value to blend between a "dark" and a "light" color, which works very well for a carefully selected set of color pairs on 3D objects [GGSC98]. Unfortunately, blending between a dark and light *image* loses that coherent texture look [GK07]. Our solution is a blend between the two — the "dark" and "light" light colors effectively produce a consistent warm-cool shading blend for every color in the dif-

fuse image, which helps to maintain the texture coherency. The artist is still free, however, to experiment with interesting material properties using the specular color image.

2.2. Refraction and sub-surface scattering

We currently implement one layer of paint (described above) and a paper layer. The "paper" is specified by an image — this image is refracted through the paint layer where the paint is transparent. The transparency is controlled through a standard alpha-matte image (given as a grey-scale image). The amount of diffusion and refraction can be controlled locally, if desired, through additional grey-scale images.

The paper color that is reflected back through the paint layer is calculated by blending together several samples from the paper image. If there is no diffusion, then these samples are all taken from where the refraction ray intersects the paper; otherwise, the samples are taken randomly around that point.

Motivation: We initially experimented with making the paper layer be lit like the paint layer, with its own height field and specular image. This introduced too much complexity into both the creation process and the resulting image. This simplified version corresponds better to the artist's notion of an underpainting, while still allowing for interesting interactions with the paint layer.

2.3. Display

The user interface allows the artist to explore their image by explicitly controlling the light, view, and scaling factors. Obviously, if this image is included as a painting in a 3D virtual scene then the lighting and view direction come from the 3D scene. We support two methods for displaying the finished painting without this information. First, the user can record a set of parameter changes and produce a movie. Second, the system can automatically generate random camera and lighting changes and continually change them, with the speed and parameter limits set by the user.

If the image is to be displayed in a public place then an alternative to the random changes is to use information from the environment, such as the location and number of people, or an approximation of the lighting [NBB04], to drive the changes.

3. Implementation

We discuss the three different components (surface generation and viewing, lighting, and diffusion) separately since they are largely independent.

3.1. Surface generation

The height gray scale image I_h (Figure 2(d)) is converted to a height field and the surface normal computed at every pixel. We produce one quad between each set of four pixels in the output image (i.e., the center of each pixel is a vertex) and add a box underneath the height field in order to support shadow mapping and diffusion. The height field is centered at $(0,0,0)$ and extends ± 1 in the x and y directions, with the box extending -1 in the z direction. The z value of the height field is the scaled image value, $S_h I_h(x,y)$. The point, texture coordinate, and surface normal at each pixel are therefore:

(δ_x, δ_y)	$=$	$(2/(1+W), 2/(1+H))$
$p(x,y)$	$=$	$\big(-1+\delta_x(x+1/2), -1+\delta_y(y+1/2),$
		$S_h I_h(x,y)\big)$
$t(x,y)$	$=$	$\big((\delta_x/2)(x+1/2), (\delta_y/2)(y+1/2)\big)$
$n(x,y)$	$=$	$\big((p(x+1,y)-p(x,y)) \times$
		$(p(x,y+1)-p(x,y))\big) +$
		$\big((p(x,y+1)-p(x+1,y+1)) \times$
		$(p(x+1,y)-p(x+1,y+1))\big)$

$$\hat{n}(x,y) = n(x,y)/\|n(x,y)\| \quad (1)$$

where $\times$ is the cross product. Note that the normal is the average of the cross product at the lower left and upper right of the pixel.

To view the painting so that it fills the screen but has minimal perspective distortion, place the camera at $(0,0,100)$, pointing at the origin with the up vector $(0,1,0)$ and the focal length $2\arctan(1/100)$.

I_d, I_s, I_p	Diffuse, specular, and paper RGB images
I_h, I_n	Height and shininess grey-scale images
I_f, I_r, I_t	Diffusion, refraction and transparency images
$S_h \in [0, 0.2]$	Height scale factor
$S_n \in [0, 5]$	Shininess scale factor
$M_n \in [1, 5]$	Shininess minimum value
$S_f, S_t \in [0, 1]$	Diffusion and transparency scale factors
$S_r \in [0, 0.2]$	Refraction scale
$R_v \in [0.2, 1.8]$	Snell coefficient

Table 1: *Images and parameters specified by the user.*

3.2. Lighting

We use the standard Blinn-Phong [Bli77] lighting model to calculate the diffuse and specular components of the BRDF, but combine them in a different way. The specular coefficient is defined in the shininess image I_n and is scaled and offset by the shininess values $M_n + I_n(x,y)S_n$. The diffuse component of the BRDF ($l_d = \hat{n} \cdot L$) is used to linearly blend between the bright light color c_b and the dark light color c_d. This blended light is used to light the diffuse image I_d. The

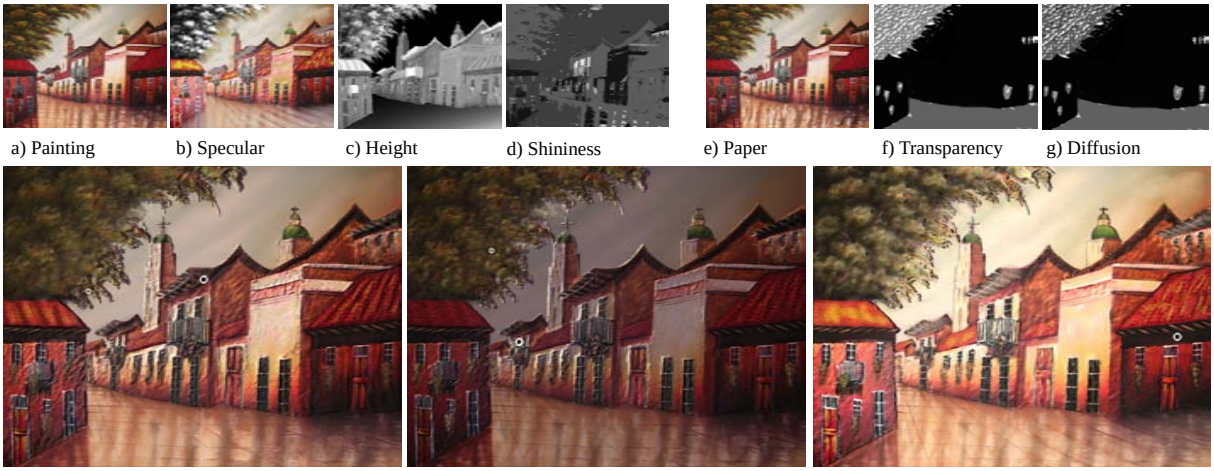

Figure 4: *(a) Photograph of acrylic painting. (b) Brightening the image, adding impasto strokes to the foreground, and turning the tree specular highlights black and white. (c) Painting the primary structures as a height field to create a relief effect. (e)-(g) Adding diffusion effects to the trees and hanging flowers. The big circle is the camera position, the little one the light position. Artist: J. Rafael. Image size: 1024 by 774.*

specular component l_s scales the bright light color and is used to light the specular image I_s. If the point is in shadow then both l_s and l_d are set to zero. The final color for the fragment is:

$$(l_d c_b + (1 - l_d) c_d) I_d(x,y) + l_s c_b I_s(x,y) \qquad (2)$$

For the images in this paper we used one point light source. The user controls the light source location by placing it in the image plane ($u, v \in [-1,1]$). The point in the plane is mapped to the upper hemisphere using a stereographic projection:

$$(x,y,z) \;=\; \frac{10}{1+u^2+v^2}(2u, 2v, 1-u^2-v^2) \qquad (3)$$

The point (u,v) is constrained to lie within a radius of 0.9 to keep the light from being nearly perpendicular to the paper.

The camera eye position is constructed in an identical way, except the distance from the paper is also under the user's control. Note that for calculating the Blinn-Phong model we only need the camera's eye position.

For building shadow maps we also need to create a projection matrix for the light. The light's camera position is as above, pointed at the origin with the up vector $(0,1,0)$ and a focal length of $2\arctan(1/5)$. This ensures that the height field stays within the light's field of view.

We can also use environment mapping to light the image. In this case, the specular light color is found by indexing the environment map with the reflected view vector. The diffuse light color is found by indexing the irradiance map with the surface normal. Since we only need one half of the hemisphere, we can again use a stereographic map to convert a normalized 3D vector (x,y,z) to the image plane (u,v):

$$(u,v) \;=\; \left(1/2 + (\frac{-x}{1-z})/2, 1/2 + (\frac{-y}{1-z})/2\right) \qquad (4)$$

3.3. Diffusion

The underlying paper image, I_p, is visible underneath the painting itself. The view vector is refracted through the painting's height field and intersected with the paper. The paper image I_p is then sampled with a random diffusion pattern around the intersection point. The transparency and amount of diffusion are controlled through two scaled images, I_t and I_f. The painting and paper color are linearly blended together using the scaled transparency image: $S_t I_t(x,y)$.

The paper color is calculated by blending together a 3×3 set of samples centered around where the refraction vector intersects the paper. The refraction amount is globally controlled by the user ($R_v = n_1/n_2$, where n_1 and n_2 are the indices of refraction for the two media). The diffusion sample set is a randomly generated set of texture map offsets in the $(-0.5, 0.5)$ range; these offsets are scaled by $S_f I_f(x,y)$.

To create a smoothly-changing diffusion effect we generate two sets of random numbers and linearly interpolate between them, tying the interpolation rate to the wall clock. The user sets the speed of the interpolation.

The refraction amount is stored as a texture map offset

from the current pixel's texture map value. We pre-calculate these values at each pixel and find the maximum offset. *All* of the refraction offsets are then scaled uniformally so that the maximum offset is S_r. Moreover, we linearly scale down the refraction offset values in a strip along the image boundary ($u, v \in (0, S_r)$ or $(1 - S_r, 1)$) so that the refraction rays do not pass outside of the paper. If there is a local refraction image then the final refraction values are locally scaled by $I_r(x, y)$.

4. Results

Figure 3 shows an example of taking a digital photograph and applying simple filters to it to bring out the bird in the foreground and de-emphasize the background. The additional input images were made in under five minutes.

Figure 1 was created from a photograph of a real oil painting. The painting has a great deal of texture in it which is not apparent in the original picture. The goal was to bring out the texture of the brush strokes and the loose style of brush strokes in the flowers and background. The height image was made by converting the original image to grey scale and increasing the contrast. The specular image was made by brightening and shifting the original image into the yellow range. Additional brush strokes were added to unify the collar and dress. The diffusion effect was added in the area around the flowers and background to create a sense of movement. The additional input images took about an hour to make.

Figure 4 was created from a photograph of a real acrylic painting. Here the goal was to emphasize the overly bright, slightly cartoon imagery. The height field was used to create a 3D relief effect between the sky and the foreground and around the balconies. The specular image was created similar to the previous example. Again, diffusion was added to bring the trees and flowers some movement, and to add emphasis to the pavement highlights in the foreground. The additional input images took about two hours to make, with most of the time spent on the height image.

Figure 5 uses a maze generated automatically from a photograph [PS06] as the paper and to generate the height field. The specular image was created by turning the original image into a simplified woodcut. Construction time was about an hour and a half, with most of the time spent painting in the maze for the paper image.

The system runs in real-time on a 2GHz Pentium with a GeForce 6600 graphics card with a display area of 924 by 924. On an older system (GeForce FX Go 5200 32M/64M) the display time is approximately two second for just the lighting effects, and 10-15 seconds when the diffuse effects are used. The program is primarily GPU bound, with the running time determined by the size of the *output* image.

Figure 6 shows the effect of changing the light colors.

Please refer also to the video and additional, full resolution images.

4.1. Remarks

The Kubelka-Munk model produces more realistic and interesting color blends than the simple alpha blending we use here [BWL04]. Unfortunately, it requires defining paints, each of which carries with it specific blending properties — the artist then defines colors by mixing these paints. If the source images for our rendering process were made using paints then the linear blending currently in the system could easily be replaced by this better color blending approach.

There are other lighting effects (sub-surface scattering, glossy surfaces, anisotropy) that could also be included in this approach.

5. Conclusion

We have presented a rendering system for digital images that supports view and lighting-based effects that can range from subtle to more extreme. These effects are relatively simple to define and suitable for implementation on the GPU.

Acknowledgments: This work was funded in part by NSF grant 0238062. Source code, executables, and the example images in this paper can be found at http://www.cs.wustl.edu/čmg/3DPainting/.

References

[Bli77] BLINN J. F.: Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph. 11*, 2 (1977), 192–198.

[BWL04] BAXTER W., WENDT J., LIN M. C.: Impasto: a realistic, interactive model for paint. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2004), ACM, pp. 45–148.

[GGSC98] GOOCH A., GOOCH B., SHIRLEY P., COHEN E.: A non-photorealistic lighting model for automatic technical illustration. *Computer Graphics 32*, Annual Conference Series (1998), 447–452.

[GK07] GRIMM C., KOWALSKI M.: Painting lighting and viewing effects. In *GRAPP* (March 2007), INSTICC, pp. 204–212.

[Her02] HERTZMANN A.: Fast paint texture. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2002), ACM, pp. 91–ff.

[KTBG03] KULLA C. D., TUCEK J. D., BAILEY R. J., GRIMM C. M.: Using texture synthesis for non-photorealistic shading from paint samples. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2003), IEEE Computer Society, p. 477.

[NBB04] NAYAR S. K., BELHUMEUR P. N., BOULT T. E.: Lighting sensitive display. *ACM Trans. Graph. 23*, 4 (2004), 963–979.

[PS06] PEDERSEN H., SINGH K.: Organic labyrinths and mazes. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2006), ACM, pp. 79–86.

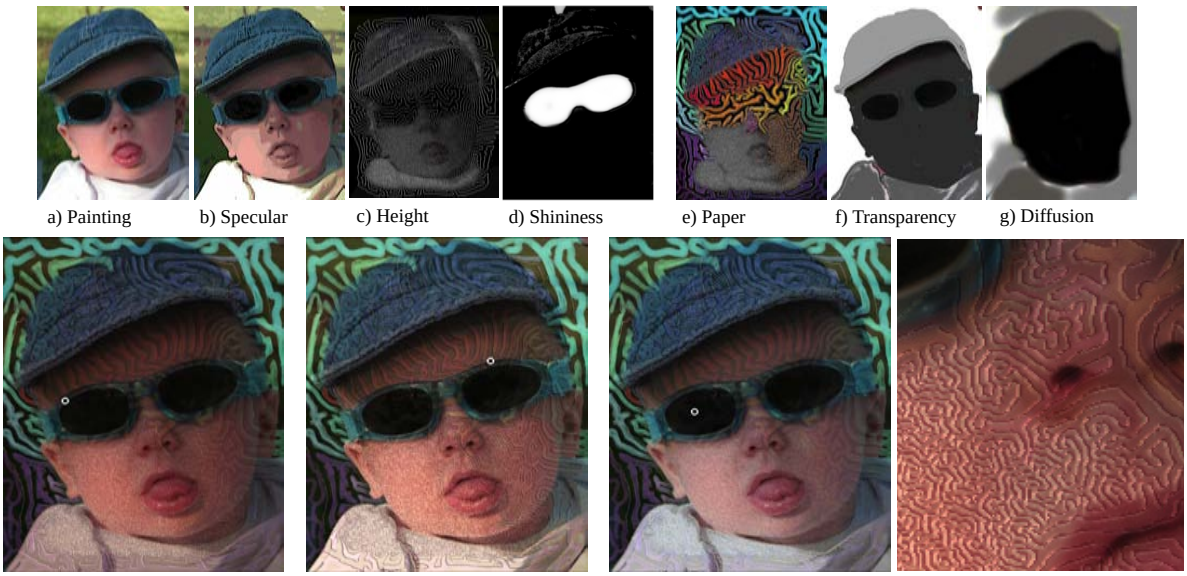

Figure 5: *(a) Original photograph. (b) Using an automated wood-cut filter to create the specular image. (c) The height field was created by using an automated program that generates a maze from the original photograph. (e)-(g) Coloring the maze and adding transparency primarily to the background and hat. Image size: 796 by 1024. The little circle indicates the light position.*

Figure 6: *Changing the light colors. "Lit" and "unlit" light colors are shown upper left. All other parameters are the same for all three images.*

[SMGG01] SLOAN P.-P., MARTIN W., GOOCH A., GOOCH B.: The lit sphere: A model for capturing npr shading from art. In *GI 2001* (June 2001), pp. 143–150.

[SZKC06] SNAVELY N., ZITNICK C. L., KANG S. B., COHEN M.: Stylizing 2.5-d video. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2006), ACM, pp. 63–69.

[TFFR07] TOLER-FRANKLIN C., FINKELSTEIN A., RUSINKIEWICZ S.: Illustration of complex real-world objects using images with normals. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)* (Aug. 2007).