Chrome, Gold and Silver on the Screen

Julia Wucharz
Hochschule Bremen
University of Applied Sciences
Flughafenallee 10
28199 Bremen, Germany
wucharz@gmx.de

Jörn Loviscach
Fachhochschule Bielefeld
University of Applied Sciences
Wilhelm-Bertelsmann-Str. 10
33602 Bielefeld, Germany
joern.loviscach@fh-bielefeld.de

ABSTRACT

A plausible rendering of metallic effects on a computer display is of high importance for 3D representations—as used in advertising and sales—and for pre-visualizing print designs that include special inks such as gold and/or silver. A human viewer recognizes these materials by their specific reflection properties. Hence, simulating them requires taking the illumination from the environment and the position of the viewer's head into account. We demonstrate that this can be achieved in a Web-based application that leverages the webcam installed on the user's computer. Thus, metallic color effects can be made available almost ubiquitously, in particular in Web shops.

Keywords
Lighting-sensitive displays, head-tracking, virtual reality, Web-based applications

1. INTRODUCTION

A car manufacturer's Web site may show the newest model of that brand as an almost photorealistically rendered 3D object. Typically, a canned environment map is employed to simulate the look of parts made of chrome. The rendered image does not depend, however, on the viewer's position so that the illusion breaks down when the user moves his or her head. The reproduction of metallic effects has been addressed even less in prepress applications, that is: applications that deal with simulating the look of a printed sheet of paper. Color management systems have been employed for more than a decade to ensure the optimal simulation of matte color prints on computer displays. Current color management systems do not, however, simulate metallic printing inks.

With 3D catalogs and 2D prepress as two fields of application in mind we have developed a Web-based system (see Figure 1) to address these issues in the reproduction of metallic colors. The system reads data from the user's webcam, leveraging the fact that webcams have become household items and mostly are already integrated in the screen bezels of notebook computers. Thus, the method cannot solely be used in software locally installed on the computer. Rather, it is also available to electronic product catalogs as used by Web shops and to online print services that want to show the effect of non-standard printing inks in advance. The contributions of this work to the state of the art comprise

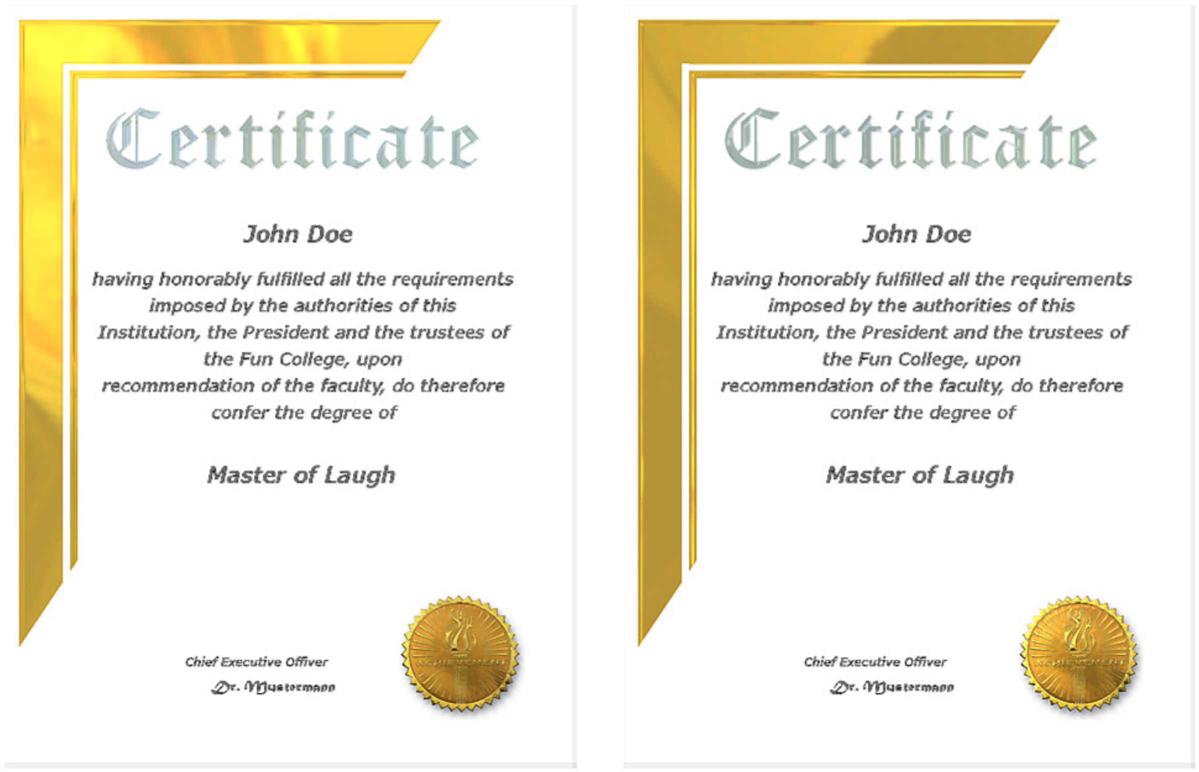
- the use of the webcam to track the position of the viewer's head—in addition to capturing the illumination—and
- the integration of all components into a Web-based application.

This paper is structured as follows: Section 2 outlines relevant related work on displays for virtual and augmented reality and on color reproduction. Section 3 describes the architecture of the prototype system, the implementation of which is covered by Section 4. Section 5 reports the results achieved; and Section 6 concludes the paper, indicating directions for future research.

2. RELATED WORK

Displays that react to their environment have been proposed at highly different levels of complexity:

Ropinski et al. [Rop04] create an environment map from the camera image to improve the look of 3D objects inserted into augmented reality displays, a



Certificate

John Doe

having honorably fulfilled all the requirements imposed by the authorities of this Institution, the President and the trustees of the Fun College, upon recommendation of the faculty, do therefore confer the degree of

Master of Laugh

Chief Executive Officer

Dr. Mustermann

Certificate

John Doe

having honorably fulfilled all the requirements imposed by the authorities of this Institution, the President and the trustees of the Fun College, upon recommendation of the faculty, do therefore confer the degree of

Master of Laugh

Chief Executive Officer

Dr. Mustermann

Figure 1. The system takes the position of the viewer's head (two positions shown) and the illumination into account to simulate metallic effects.

technique that was already outlined by Miller [Mil95]. Daniel [Dan05a] employs a camera with a fisheye lens to capture an environment map and illuminate 3D objects displayed on the screen through pre-computed radiance transfer. This is limited to diffuse lighting. The exposure time of the camera alternates between a long and a short setting to synthesize a higher dynamic range. Nayar et al. [Nay04a] describe a display that makes similar use of a fisheye lens but employs a large data-compressed collection of pre-rendered or pre-captured images for full relighting including specular highlights.

Fuchs et al. [Fuc08a] discuss options to build passive reflectance field displays, that is: displays that react to illumination—in this case illumination from behind. Using microlens arrays and LC display panels in a similar fashion, Koike and Naemura [Koi08a] demonstrate a "BRDF display," in which the directional response to the incoming illumination can be controlled digitally. Reproducing metallic effects with such a system would, however, require a huge angular resolution to produce appropriately sharp reflections.

In a patent application [Ker09] that has been published after the submission of this paper, Kerr and King of Apple, Inc., propose to track the user's head—for instance through a camera—to simulate 3D effects on a 2D screen. The user may for instance "look around" the edges of window in the foreground to see what is behind; including reflections of the environment is mentioned, too. Mannerheim and Nelson [Man08] propose using a camera to track the location of the user's head in order to adjust a binaural audio signal presented through loudspeakers.

Many goggle-free (i.e., autostereoscopic) 3D virtual reality displays employ head-tracking to project the left and right partial images onto the respective eye of the user; for an example, see [San05a]. The data thus gained can in principle be employed to render specular and mirroring reflections based on the actual position of the viewer.

Color management systems [Sto04a] are a standard amenity of current computer operating systems. They operate on the basic principle of converting colors from device-dependent spaces such as RGB and CMYK to device-independent spaces such as XYZ or CIELAB. This conversion is described through profiles for each input and output device such as camera, scanner, display, or printer. Current color management systems only support perfectly diffuse reflection

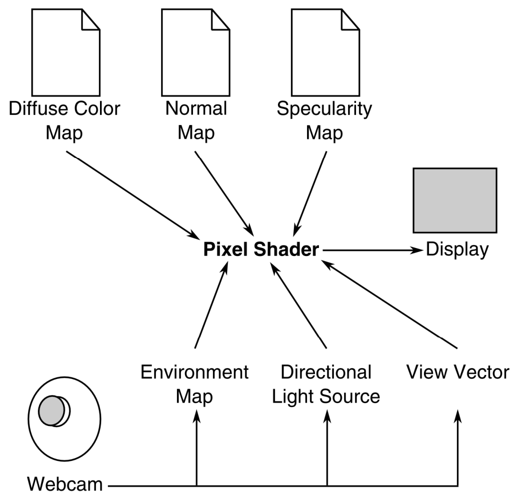


Figure 2. The system reads three two-dimensional maps and the image stream from the webcam to feed the pixel shader used for rendering.

models. Whereas color models for metallic inks have been researched into [Her03a], they have not yet found their way into off-the-shelf prepress software solutions.

3. ARCHITECTURE

This work focuses on rendering a sheet of paper or a single view of an object and trying to create as lean and hence Web-compliant a system as possible. Hence, we confine ourselves to working with two-dimensional maps instead of operating on complete three-dimensional meshes as has been done in former work on Mixed Reality. The input to the system consists of several maps, which typically are stored on the server side: the color data for diffuse reflection (a standard RGB image), a normal map (encoded as RGB image), and a specularity map (encoded as grayscale image) that defines the blend between matte and metallic behavior per pixel.

The non-metallic part of the model is rendered with the Lambertian model [Bli77]. The metallic part employs the Cook-Torrance model [Coo82] with fully editable parameters. In the software prototype, these are offered as controls on the graphical user interface. In an actual application, however, they would be set and frozen during the authoring phase and then be stored as part of the media file or in a configuration file.

To adapt the sharpness of the reflected environment to the selected sharpness of the highlights, the environment map can be sampled down by an adjustable power of two. Figure 2 shows the overall architecture.

To provide a system that works over the Web with a typical computer on the client side we elected to employ a standard webcam image instead of an image shot through a fisheye lens. The image taken by the webcam is used for three purposes:

First, a cube map of a plausible environment is build from the image. The front face of the cube is formed by the camera image as such, cropped from both the left and right side by the eighth part of its width. The remaining parts on the left and right are put into the left and right faces of the cube map and extended through repetition of the last pixel column. The bottom and the top face of the cube map are formed through repetition of the first row or the last row of the camera image, respectively. To partially hide the repetitions, the lateral faces are feathered toward black at their ends, see Figure 3.

Second, a coarse-grained version of the camera image is searched for the brightest spot. For the rendering, a light source with this color is placed accordingly. Thus, one strong specular reflection is taken into account without high dynamic range imaging and without complex rendering algorithms, see Figure 4.

Third, the user's head is found in the camera image using an existing software library (see Section 4). The center of the head is used to define the view direction for the rendering, see Figure 5. In case no webcam is available—for example, out of privacy concerns—, the user can choose to steer the viewing position

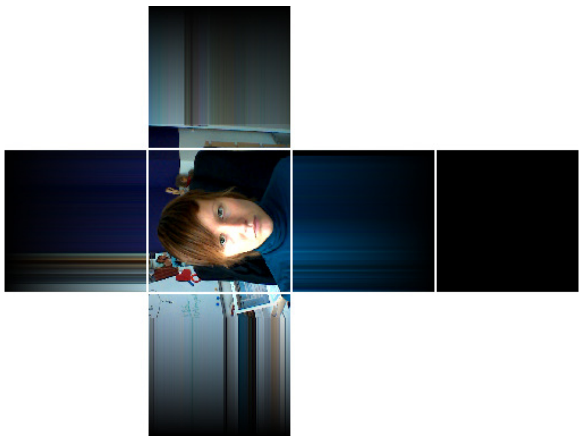


Figure 3. The environment cube map is built from the camera's input through cropping, the repetition of the final rows and columns, and feathering.

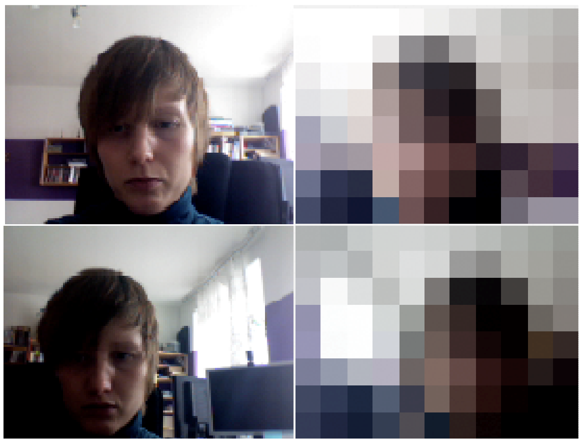


Figure 4. To determine a position for a single dominant light source, the camera image (left) is sampled down to large blocks (right).

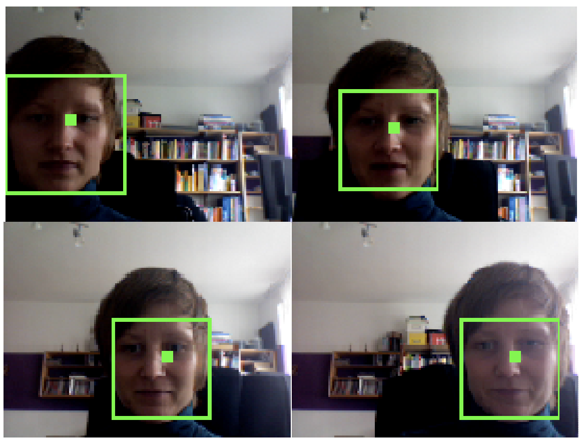


Figure 5. A point slightly above the centroid of the largest rectangle returned by the face detector controls the view direction of the lighting model.

through the mouse and apply one of several environment maps included with the software.

4. IMPLEMENTATION

The prototype has been developed in ActionScript 3 using the Adobe Flex Builder 3 development environment based on the Flex software development kit 3.4 targeting Adobe Flash Player version 10 or newer.

The face-tracking component employs the “Marilena” port [Mas09a] of the face detection in the OpenCV library. This detector [Vio01] employs a cascade of simple classifiers that use the contrast between averages over rectangular parts of the image. These averages can be computed quickly through a summed-area table. Consequently, the features being detected resemble Haar wavelets. The training data that has been generated upfront is based on a variant of the AdaBoost. In this case, during training the best features (that is: sets of rectangles) are found and the classifiers are adjusted, whereas a classical AdaBoost would only concern the latter step.

The rendering has been realized through a shader routine developed with Adobe’s Pixel Bender Toolkit 1.5 [Ado09a]. PixelBender comprises of a basic interactive development environment to build image processing routines (called “kernels”) in a programming language resembling the OpenGL Shading Language GLSL. The range of available functions corresponds to a pixel shader in standard GPU programming. The kernels thus created can be connected into dataflow graphs and can be compiled to byte-code to be loaded and executed in Flash Player 10.

As the Pixel Bender Toolkit as such offers GPU acceleration for the kernels, it is foreseeable that future versions of Flash Player also execute kernels on the GPU instead of running them on the CPU as the current version does. Then a vital part of the acceleration offered by the graphics processor can be leveraged even in this Web-based software. The circumstance that Pixel Bender only targets pixel processing but not mesh processing fits nicely to the scope of our application.

5. RESULTS

We measured the performance of the system on an Apple MacBook computer, which runs Mac OS X 10.5.8., is equipped with an Intel Core 2 Duo processor running at 2.2 GHz and an integrated webcam, It does not contain a dedicated graphics chip but uses the Intel GMA X3100 chipset graphics instead.

At an image size of 512 x 384 pixels, the software prototype with all functions applied runs at 15 frames per second; at an image size of 615 x 461 pixels, this rate decreases to 8 frames per seconds. With all camera functions switched off, the rendering alone easily achieves 30 frames or more per second. This shows that the processing of the camera image is the step that limits the performance.

One may hope that future application frameworks grant direct access to head-tracking data and thus relieve the application from such computations. Most popular webcams already come with robust and computationally lean integrated head-tracking to add 3D items such as hats or sunglasses to the user’s face. Currently, however, there is no official way to access these head-tracking data from other software.

The .swf file that is transferred to the client computer and contains the complete code of the application possesses a size of 260 KB. The three maps (diffuse color, normal, specularity) add to this size; their byte count depends heavily on the compression used.

The pixel repetitions used to build the environment cube map (see Figure 3) may become visible in extreme situations, namely if large, flat and perfectly mirroring surfaces are viewed from head positions that are strongly off-center. In all other cases, the details of the texture and specularity maps and/or the blurriness of the reflection hide these artifacts. This becomes apparent in Figure 6, which also demonstrates the use of our system with two-dimensional normal maps of three-dimensional meshes: Even though the object does not rotate, the look of polished metal is reproduced faithfully.

For speed and simplicity, the color computations are executed in RGB space and employ the automatic clamping of the RGB components. Thus, bright highlights—as they are more or less required for metallic effects—appear color-shifted toward white. For instance, the internally computed color (1.9, 1.5, 0.9) does not appear on the display screen as reddish orange but as (1.0, 1.0, 0.9), which is a slightly yellowish white. Even though this effect is only apparent to the trained eye, a color clamping that restricts the hue of the color to its original value could suppress it, at the cost of less brighter highlights.

6. CONCLUSION AND OUTLOOK

We have demonstrated a system that plausibly simulates of metallic colors but remains inexpensive in terms of computer hardware and computational effort. In particular, the system leverages standard Internet technology and can thus be employed in Web shops, electronic advertisements, etc.

Future developments can target the precision of the simulation of the lighting, possibly turning the plausible result into an almost visually exact one. Doing so would require dealing with camera calibration, generating environment maps with a high dynamic range from a standard camera [Dan05a], and creating cheap but precise ancillary lenses to turn a standard webcam into a fisheye camera. The reproduction of perfectly mirrored reflections on extended flat surfaces could be improved through replacing the pixel repetition in the cube map by a synthesized texture. Strong highlights would benefit from bloom effects based on high-dynamic range computations of the colors.

An integration of 3D meshes looks straightforward from the algorithmic side. In terms of performance, however, Adobe Flash—running on the computer's CPU—may be overcharged with such a task. In future, a more general approach that requires no browser plug-in may become possible through the advent of WebGL [Mar09].

A second avenue of development would be to focus on strengthening the connection to color management systems in their present form. A standard color management system could handle the diffuse illumination and a system similar to the prototype we have described could add gloss and mirror effects.

7. REFERENCES

[Ado09a] Adobe. Adobe Pixel Bender technology. http://labs.adobe.com/technologies/pixelbender/, last accessed 2009-10-24.

[Bli77] Blinn, J.F. Models of light reflection for computer synthesized pictures. SIGGRAPH Computer Graphics 11, No. 2, pp. 192–198, 1977.

[Coo82] Cook, R.L., and Torrance, K.E. A reflectance model for computer graphics. ACM Transactions on Graphics 1, No. 1, pp. 7–24, 1982.

[Dan05a] Daniel, T. Real-time video lighting. ACM SIGGRAPH '05 Posters, Los Angeles CA, ACM Press, p. 50, 2005.

[Fuc08a] Fuchs, M., Raskar, R., Seidel, H.-P., and Lensch, H.P.A. Towards passive 6D reflectance field displays. ACM Transactions on Graphics 27, No. 3, Article No. 58, 2008.

[Her03a] Hersch, R.D., Collaud, F., and Emmel, P. Reproducing color images with embedded metallic patterns. Proceedings of ACM SIGGRAPH '03, Los Angeles CA, ACM Press, pp. 427–434, 2003.

[Ker09] Kerr, D.R., and King, N.V. Systems and methods for adjusting a display based on the user's position. United States Patent Application 20090313584, 2009.

[Koi08a] Koike, T., and Naemura, T. BRDF display: interactive view dependent texture display using integral photography. Proceedings of IPT/EDT '08, Los Angeles CA, ACM Press, Article No. 6, 2008.

[Mas09a] Masakazu, O. Marilena: port of the OpenCV face to ActionScript 3. http://www.libspark.org/wiki/mash/Marilena, last accessed 2009-10-24.

[Man08] Mannerheim, P., and Nelson, P.A. Virtual sound imaging using visually adaptive loudspeakers. Acta Acustica united with Acustica 94, No. 16, pp. 1024–1039, 2008.

[Mar09] Marrin, C. (editor). WebGL specification, working draft 10 December 2009. http://www.khronos.org/webgl/, last accessed 2010-01-02.

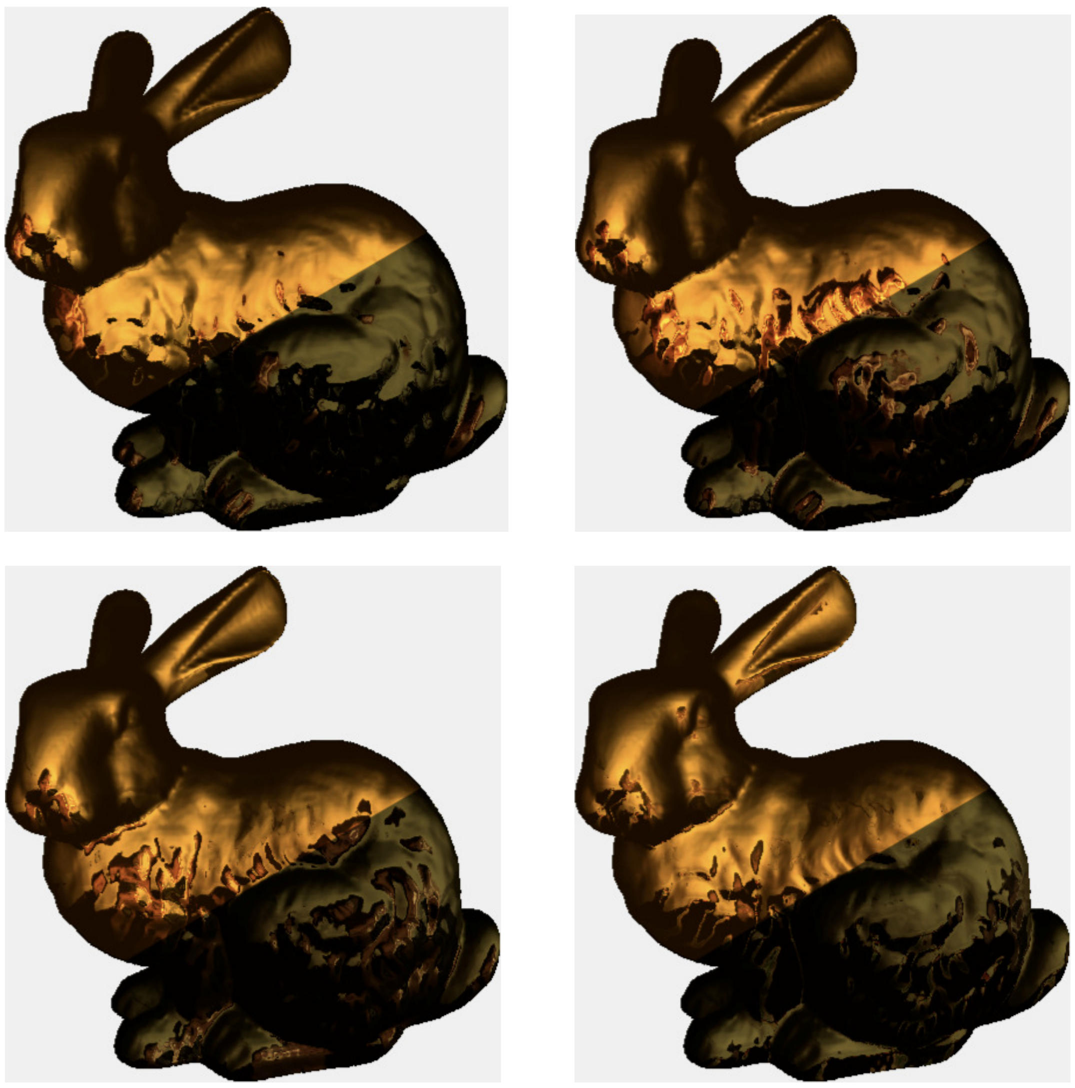


Figure 6. In addition to visualizing two-dimensional relief prints on paper, the system can also plausibly convey the look of metallic 3D objects as described through a 2D normal map. In this image sequence, the user's head has moved from left to right. For demonstration, a specularity map with less metallicity below the diagonal has been applied. (Stanford Bunny courtesy of http://graphics.stanford.edu/data/3Dscanrep/)

[Mil95] Miller, G. Volumetric hyper-reality, a computer graphics holy grail for the 21st century? Proceedings of Graphics Interface '95, Québec Québec, Canadian Information Processing Society, pp. 56–64, 1995.

[Nay04a] Nayar, S.K., Belhumeur, P.N., and Boult, T.E. Lighting sensitive display. ACM Transactions on Graphics 23, No. 4, pp. 963–979, 2004.

[Rop04] Ropinski, T., Wachenfeld, S., and Hinrichs, K.H. Virtual reflections for augmented reality environments. Proceedings of Artificial Reality and Telexistence (ICAT04), Seoul, Korea, pp. 311–318, 2004.

[San05a] Sandin, D.J., Margolis, T., Ge, J., Girado, J., Peterka, T., and DeFanti, T. A. The Varrier autostereoscopic virtual reality display. ACM Transactions on Graphics 24, No. 3, pp. 894–903, 2005.

[Sto04a] Stone, M.C. Color in information display: principles, perception, and models. ACM SIGGRAPH '04 Course Notes, Los Angeles CA, ACM Press, Course 21, 2004.

[Vio01] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. Proceedings of Computer Vision and Pattern Recognition (CVPR 2001), Kauai HI, IEEE Press, pp. I-551–I-518, 2001.