Non-photorealistic rendering

UCSD CSE 167
Tzu-Mao Li
So far: photorealistic rendering

- we get our image by simulating the interaction between lights and surfaces
- a style long established by Pixar since 1980s

Motion blur

Fur materials

Defocus blur

Specular highlights

Shadow

Global illumination

Subsurface scattering

Monster University [2013] @ Pixar

Red Dead Redemption 2 [2018] @ Rockstar Games
Pros and cons of photorealistic rendering

• pros
  • automatically get “good” images
  • blend well with real photographs
  • safe and well-established
  • can be used for tasks outside of entertainment

• cons
  • expensive to render
  • kind of…get old?

Transformers [2007] @ Industrial Light and Magic

Gran Turismo 7 [2022] @ Polyphony Digital
Non-photorealistic rendering (NPR)

- techniques that deviate from traditional physically-based approaches

Return of the Obra Dinn [2018] © Lucas Pope

Okami [2006] © Clover Studio
The film industry is moving towards NPR?

Spider-Man: Into the Spider-Verse [2018] @ Sony

The Mitchells vs. The Machines [2021] @ Sony

Puss in Boots: The Last Wish [2022] @ Dreamworks
There is no single way to achieve non-photorealistic looks

• no “rendering equation”

• not a very well-researched area

• I’m very interested in this!!!

• currently, can be seen as a bunch of “hacks”

https://www.gamedeveloper.com/production/art-design-deep-dive-using-a-3d-pipeline-for-2d-animation-in-i-dead-celle-i-
NPR often takes inspiration from art

Galileo’s drawing of moon
Today: a bunch of examples

- shading
- outlines
- painterly rendering
- hatching
- half-toning
Today: a bunch of examples

- shading
- outlines
- painterly rendering
- hatching
- half-toning
Toon shading/cel shading

https://www.reddit.com/r/Unity3D/comments/afygr0/i_wrote_a_tutorial_for_tooncel_shading_linksource/
Toon shading/cel shading

idea: apply standard shader -> **quantize** the result

\[ K_d \max (N \cdot L, 0) \]

\[ K_d (N \cdot L > 0 ? 1 : 0) \]

https://roystan.net/articles/toon-shader/
Toon shading/cel shading

idea: apply standard shader -> **quantize** the result

\[ K_d (N \cdot L > 0 ? 1 : 0) \]

\[ K_a + K_d (N \cdot L > 0 ? 1 : 0) \]

[https://roystan.net/articles/toon-shader/](https://roystan.net/articles/toon-shader/)
Toon shading/cel shading

idea: apply standard shader -> **quantize** the result

\[ K_a + K_d (N \cdot L > 0 \? 1 : 0) \]

add specular highlight

https://roystan.net/articles/toon-shader/
Toon shading / cel shading

idea: apply standard shader -> **quantize** the result

add specular highlight

add “rim lighting” \((1 - V \cdot N) (N \cdot L)^{0.1}\)

https://roystan.net/articles/toon-shader/
Toon shading/cel shading

idea: apply standard shader -> **quantize** the result

add “rim lighting”

add shadow

https://roystan.net/articles/toon-shader/
Toon shading/cel shading is widely used

The Legend of Zelda: Breath of the Wild [2017] @ Nintendo

https://roystan.net/articles/toon-shader/
Toon shading/cel shading is widely used in animated films. For example, in *The Great Mouse Detective* (1986) by Disney, the gears are CGs and the characters are hand drawn.

https://www.youtube.com/watch?v=FLjj0SJTuSM
Toon shading/cel shading is widely used

Okami [2006] @ Clover Studio
Toon shading/cel shading is widely used.

Dead Cells [2018] @ Motion Twin

https://www.gamedeveloper.com/production/art-design-deep-dive-using-a-3d-pipeline-for-2d-animation-in-i-dead-cells-i-
Toon shading/cel shading is widely used

Guilty Gear Xrd [2014]  
@ Arc System Works
Toon shading requires heavily tuned lighting.
Gooch shading [1998]

\[ K_d \max (N \cdot L, 0) \]

\[ \left( \frac{1 + L \cdot N}{2} \right) K_{\text{blue}} + \left( \frac{1 - L \cdot N}{2} \right) K_{\text{yellow}} \]
Today: a bunch of examples

- shading
- outlines
- painterly rendering
- hatching
- half-toning
Artists use lines to convey shapes.

https://www-sop.inria.fr/reves/Basilic/2019/GSHPDB19/
https://people.csail.mit.edu/tjudd/apparentridges.html
Goal: 3D models + camera -> lines

http://people.csail.mit.edu/tjudd/apparentridges.html
Occluding contours are important cues for line drawing

Occluding contours are important cues for line drawing.

Definition: set of points on the surface s.t. $N \cdot V = 0$.

Two ways to extract occluding contours

**image-space methods:**
find pixels with large depth variations

**3D methods:**
directly find them on 3D mesh given a camera

Two ways to extract occluding contours

read more here

LINE DRAWINGS FROM 3D MODELS: A TUTORIAL

Pierre Bénard
LaBRI (UMR 5800, CNRS, Univ. Bordeaux)
Inria Bordeaux Sud-Ouest
pierre.benard@labri.fr

Aaron Hertzmann
Adobe Research
hertzman@dgp.toronto.edu

Preprint

ConTesse: Accurate Occluding Contours for Subdivision Surfaces
Chenxi Liu1 Pierre Bénard2 Aaron Hertzmann3 Shayan Hoshuyar4
1 University of British Columbia 2 Univ.Bordeaux, CNRS, Bordeaux INP, INRIA, LaBRI, UMR 5800
3 Adobe Research 4 Adobe

Algebraic Smooth Occluding Contours
Ryan Capezzel
New York University
USA
tj@cs.nyu.edu

Aaron Hertzmann
Adobe
USA
hertzman@dgp.toronto.edu

Jaicheng Dai
New York University
USA
jld700@nyu.edu

Denis Zorin
New York University
USA
deniz@cs.nyu.edu

Occluding Contour Breakthroughs, Part 1: A Surprisingly Hard Problem

Aaron Hertzmann's blog
Occlusion contour alone is not sufficient!
A few options to add contours

suggestive contour [DeCarlo 2003]: draw contours when curvature in the image space = 0
(draw at dark regions under Lambertian shading)
A few options to add contours

apparent ridges [Judd 2007]: draw contours when image space curvature is large
(draw at places where color changes very fast)
A few options to add contours

neural contours [Liu 2020]: just train a neural network to do it

3D model
occluding contours
apparent ridges
neural contours
Where do people draw lines? [Cole 2008]

- invited 29 artists (students + professionals) to draw line drawings of 3D models
- occluding contours explain 50-65% of lines
- occluding contours + suggestive contours + apparent ridges explain 80-90% of lines
We can use lines to enhance shape perception!
We can use lines to enhance shape perception!

Spider-Man: Across the Spider-Verse [2023] @ Sony
We can use lines to enhance shape perception!
We can use lines to enhance shape perception!

A Scannerly Dark [2006]
Today: a bunch of examples

- shading
- outlines
- painterly rendering
- hatching
- half-toning
NPR by image processing

Painterly Rendering with Curved Brush Strokes of Multiple Sizes

2000

Aaron Hertzmann
Media Research Laboratory
Department of Computer Science
New York University
Algorithm from Hertzmann: greedily make a “stroke” to minimize the error between the current painting and the target image

NPR by image processing

for \( x=0 \) to \( \text{imageWidth} \) stepsize grid do
\[ \text{for } y=0 \text{ to } \text{imageHeight} \text{ stepsize grid do} \]
\[
\begin{aligned}
&// \text{ sum the error near } (x,y) \\
&M := \text{the region } (x-\text{grid}/2..x+\text{grid}/2, \\
&y-\text{grid}/2..y+\text{grid}/2) \\
\text{areaError} := \sum_{i,j\in M} D_{i,j} / \text{grid}^2 \\
\text{if } \text{(areaError > } T) \text{ then} \\
&\begin{aligned}
&// \text{ find the largest error point} \\
&(x_1, y_1) := \text{arg max}_{i,j\in M} D_{i,j} \\
&s := \text{makeStroke}(R, x_1, y_1, \text{referenceImage}) \\
&\text{add } s \text{ to } S \\
&\end{aligned}
\end{aligned}
\]
\[ \text{paint all strokes in } S \text{ on the canvas,} \]
\[ \text{in random order} \]
NPR by image processing

Algorithm from me [2020]: gradient descent over stroke parameters

optimization (vector)  

target
NPR by image processing

Algorithm from me [2020]:
gradient descent over stroke parameters
NPR by image processing

can also be done using some form of hill climbing algorithm

https://github.com/fogleman/primitive
NPR by roto-scoping

artists hand draw frames directly on top of a raw footage

A Scanner Darkly [2006]

NPR by rotoscoping
NPR by rotoscoping

Loving Vincent [2017]
NPR by rotoscoping

How I Animated This Video
[2021] @ Joel Haver
Today: a bunch of examples

- Shading
- Outlines
- Painterly rendering
- Hatching
- Half-toning
Hatching: a drawing technique for conveying both shape and shading

https://paintbasket.com/cross-hatching-for-beginners/
Automatic hatching algorithm

For each point on the surface, we can generate two perpendicular vectors that correspond to the fastest and slowest changing normals; these are called the “principle curvature directions.”
Automatic hatching algorithm

rotate and stretch a "hatching" texture based on the principle curvature directions, use denser hatching if the surface is darker
Automatic hatching algorithm
From “Into the Spider-Verse”

https://www.youtube.com/watch?v=l96IgQmXmhM
Today: a bunch of examples

- shading
- outlines
- painterly rendering
- hatching
- half-toning
Half-toning: a printing technique

using black/white dots to represent gray colors
Early video games use half-toning to increase color variations.

Uncharted Waters [1990] @ KOEI (大航海時代)
NEC PC-98 version

https://twitter.com/ruuupu1/status/1425412567392944131
Algorithm to produce half-toning: dithering

Heavily used in Into the Spider-Verse for recreating the comic book feel
Used in modern video games to create unique art styles

Return of the Obra Dinn

Return of the Obra Dinn [2018] @ Lucas Pope
Check out Lucas Pope’s article for how to achieve temporally stable dithering!

Today: a bunch of examples

- shading
- outlines
- painterly rendering
- hatching
- half-toning
Overall, NPR is highly related to human perception. Real scene (possibly imaginary) → additional data, emotion, etc → similar impression, data well transmitted → illustration & idea inspired by Fredo Durand → additional data, emotion, etc.
Overall, NPR is highly related to human perception

Designing Perceptual Puzzles by Differentiating Probabilistic Programs

KARTIK CHANDRA, MIT, USA
TZU-MAO LI, UCSD, USA
JOSHUA TENENBAUM, MIT, USA
JONATHAN ROGAN-KELLEY, MIT, USA

We design new visual illusions by finding “adversarial examples” for pre-trained models of human perception—gradually, for probabilistic models, which treat vision as Bayesian inference. To perform this search efficiently, we design a differentiable probabilistic programming language, whose API exposes HMC as a first-class differentiable feature. We demonstrate our method by systematically creating illusions for three features of human vision: color constancy, size constancy, and fusion perception.

ACM Reference Format:

1 INTRODUCTION

Human visual perception does not always match objective reality. Consider “The Dress,” a viral illusion discovered by chance in 2015. This photograph of a blue and black striped dress elicits a strange perceptual effect: while most viewers indeed describe the dress as blue and black, roughly a third instead confidently describe it as white and gold. [Lai/Sorescu et al. 2015]. Because these two perceptual modes are so dramatically different, viewers are often baffled to learn that others see the same image differently. Illusions like “The Dress” have long been studied for insight into perception. This paper is motivated by the question: Can we systematically generate new such illusions in a principled manner?

Following [Ozden et al. 2019], we would like to think of such patterns as “image synthesis” or “inverse vision rendering.” Given a model of human visual perception that infers a scene from an image (inverse rendering), we wish to search for input images that elicit interesting responses as output from the model (inverse vision rendering).

What model should we choose? A long line of cognitive science research suggests that human visual perception is enabled well by probabilistic models, which treat perception as Bayesian inference. Under such a model, a viewer has some prior belief about the statistical distribution of scenes (objects, their colors, and lighting condition) in the world—tuned with an observation (i.e., image). The viewer performs evidence to update that prior belief into a posterior belief (which can be used to generate images of scenes).

Probabilistic models have already been used for inverse rendering in a variety of settings. Here, we instead are interested in inverse vision modeling: we seek to elicit such a behavior that the model yields a posterior belief, given it the distribution of scenes (i.e., images), is conditionally incorrect. In this way, images are like “adversarial examples” — not for a deep neural network, but instead for a small, principled model of one aspect of human perception

In this paper, we present a general purpose tool for solving such inverse problems: a differentiable probabilistic programming language. In particular, probabilistic programming languages (PPLs) allow users to express probabilistic models as structured generative processes, which are then converted into efficient programs for performing Bayesian inference with respect to a given observation. By enabling differentiation through the inference process, a differentiable PPL allows users to optimize (by gradient descent) an observation to evoke some desired property in the inferred posterior distribution. Using this tool, we can model a variety of perceptual phenomena with simple generative models—a few lines of code—and then search for stimuli that elicit interesting behaviors from the models.

check out our recent paper on a potential way to do NPR!
Next: procedural modeling