CSE 659A: Advances in Computer Vision

Spring 2020: T-R: 1:00-2:20pm @ Zoom

Instructor: Ayan Chakrabarti (ayan@wustl.edu).

http://www.cse.wustl.edu/~ayan/courses/cse659a/

April 9, 2020
ILLUMINATION ESTIMATION

Last Time

- In the most general case, object shading is based on an "environment map".
- For AR and image-editing applications, we sometimes want to estimate this map to figure out how new objects will appear.
- So let's train a neural network that, given an image, predicts this map.
- But how do we get ground truth training data?
- To the rescue: 360 degree panorama cameras, and datasets collected using such cameras - SUN360: Xiao et al., CVPR 2012.
ILLUMINATION ESTIMATION

ILLUMINATION ESTIMATION

ILLUMINATION ESTIMATION

ILLUMINATION ESTIMATION

- Indoor Illumination: More complicated, possibly multiple light sources.
- But similar idea: Gardner et al., Learning to Predict Indoor Illumination from a Single Image
But similar idea: Gardner et al., Learning to Predict Indoor Illumination from a Single Image
ILLUMINATION COLOR ESTIMATION

- Recall the color constancy problem:

\[ I = W \times L \]

- Observed image is product of true surface color and light color.
- We are interested in "white-balancing" the image. Estimating and dividing by the 'chromaticity' of the light to get \( W \) from \( I \).
- Again, don't want to undo shading ... only the "color cast".
- In 559A, we talked about simple "priors": gray world (average true color is gray), white patch (brightest true color is achromatic).
- Let's use neural networks instead.
ILLUMINATION COLOR ESTIMATION

- Two versions: both train neural networks to predict chromaticity. But instead of regression, turn it into a classification problem.

- Discretize the space of chromaticity values $r^2 + g^2 + b^2 = 1$.
- Output a probability distribution over this space.
ILLUMINATION COLOR ESTIMATION

Ver 1: Chakrabarti, "Color Constancy by Learning to Predict Chromaticity from Luminance," NIPS 2015

Classify true chromaticity for every pixel independently from a grayscale version of the input image. Divide by observed color to get corresponding distribution over illuminant chromaticity. Assuming uniform illuminant, do global average pooling and estimate illuminant color.

Ver 2: Barron, "Convolutional Color Constancy," ICCV 2015

Train a CNN that looks at the observed color image, and outputs a distribution over illuminant color chromaticity.

Both methods work better than prior hand-crafted approaches. But both also make the assumption of a single illuminant color (ver1 does so in the second step).

What happens if you have multiple illuminants in the scene (indoor and outdoor lighting). Can you do white balance? Can you separate individual illuminants out?
Given an image with two light sources with two different colors ...

Learn to separate them into two images: each corresponding to what would have been observed under only one of the lights.

Can then use this to do light-editing: turn off one light, change relative brightness of different lights, change the color of one of the lights but not the other.
ILLUMINATION COLOR ESTIMATION


- Simplified description of their model:

\[ I[n] = \rho[n] \cdot (\alpha[n] \ell_1 + \beta[n] \ell_2) \]

- \( I[n], \rho[n], \ell_1, \ell_2 \) are all three channel RGB vectors. \( \cdot \) is element-wise multiply.

- \( \alpha[n] \) and \( \beta[n] \) are scalars / single-channel. Represent the corresponding "shading" from each light.

- Key Idea: take an image of the scene, and then take a second image with a flash, whose color \( \ell_f \) we know.

\[ I_2[n] = \rho[n] \cdot (\alpha[n] \ell_1 + \beta \ell_2[n] + \Delta[n] \ell_f) \]

- Now subtract \( I \) from \( I_2 \)

\[ I_2[n] - I[n] = \rho[n] \Delta[n] \ell_f \]

- Now, we can divide by the known light color \( \ell_f \):

\[ \Lambda[n] = (I_2[n] - I[n]) \cdot^{-1} \ell_f = \rho[n] \Delta[n] \]

- So \( \Lambda[n] \) now depends only on the true surface color \( \rho[n] \), but multiplied by a unknown shading scalar \( \Delta[n] \)
ILLUMINATION COLOR ESTIMATION

\[ \Lambda[n] = (I_2[n] - I[n]) \circ^{-1} \ell_f = \rho[n] \Delta[n] \]

- But notice that the "chromaticity" of \( \Lambda[n] \) is the same as the chromaticity of \( \rho[n] \)

\[ \hat{\Lambda}[n] = \frac{\Lambda[n]}{\|\Lambda[n]\|} = \frac{\rho[n]}{\|\rho[n]\|} = \hat{\rho}[n] \]

- Now go back to the original image:

\[ I[n] = \rho[n] \circ (\alpha[n] \ell_1 + \beta[n] \ell_2) \]

- If I divide \( I[n] \) by \( \hat{\rho}[n] \) and then normalize each pixel to be unit norm, I have the effective chromaticity of shading:

\[ \Gamma[n] = \frac{\alpha[n] \ell_1 + \beta[n] \ell_2}{\|\alpha[n] \ell_1 + \beta[n] \ell_2\|} \]

- If I plot the \( \Gamma[n] \) of all points on the chromaticity space, they will all lie along an arc. (Because the un-normalized versions all lie along a line in RGB space).
- Given these observed values, fit them to an arc, and the end-points of the arc will give you the chromaticity of the two lights \( \hat{\ell}_1 \) and \( \hat{\ell}_2 \).
ILLUMINATION COLOR ESTIMATION

\[ \Lambda[n] = (I_2[n] - I[n]) \ast^{-1} \ell_f = \rho[n] \Delta[n] \]

\[ \hat{\Lambda}[n] = \frac{\Lambda[n]}{\|\Lambda[n]\|} = \frac{\rho[n]}{\|\rho[n]\|} = \hat{\rho}[n] \]

\[ I[n] = \rho[n] \ast (\alpha[n] \ell_1 + \beta[n] \ell_2) \]

\[ \Gamma[n] = \frac{\alpha[n] \ell_1 + \beta[n] \ell_2}{\|\alpha[n] \ell_1 + \beta[n] \ell_2\|} \]

- The position along the arc will tell you the ratio between \(\alpha[n] \ell_1\) and \(\beta[n] \ell_2\).
- This means, you can split \(I[n]/\hat{\rho}[n]\) into \(\alpha[n] \ell_1\) and \(\beta[n] \ell_2\), and then multiply it with \(\hat{\rho}[n]\) to get the two separated images.
ILLUMINATION COLOR ESTIMATION

No-flash/flash images  Pure flash image  Histogram of $\Gamma$

Separated images

Image of $\frac{\alpha}{||\alpha||}$  Image of $\Gamma$
ILLUMINATION COLOR ESTIMATION

- Works great, but we need a flash / no-flash pair.
- In-convenient, and also, won't work in scenes with objects too far for the flash to reach, or a very bright ambient light source that drowns out the flash.
- But the only place we use the flash image is to compute the reflectance chromaticity $\hat{\rho}[n]$. 
- And we've solved that exact same problem from a single image, with neural network for color-constancy!
- Basic idea, train a network to output $\hat{\rho}[n]$, and then apply the above separation method. (But try to do this end-to-end, so that the loss on $\rho[n]$ is based on accuracy of final separation).

ILLUMINATION COLOR ESTIMATION
ILLUMINATION COLOR ESTIMATION

(a) Input photographs

(b) Hui et al. [24]

(c) Ours
ILLUMINATION COLOR ESTIMATION

(a) Input
(b) Hui et al. [24]
(c) Our

(a) Input image
(b) Output separated images
So far, we have mostly talked about algorithms for images, taken from traditional cameras. But traditional cameras were designed to capture images "directly":

- Very similar to film camera designs, where the final photograph is directly exposed by a chemical reaction on the film.

But now, we have significant more computational resources available on the camera.
- Can rely on more advanced algorithms to produce images that "look good".

In many cases, how the images look doesn't even matter. Will be using images to solve vision tasks, not show them to humans.

Given all this, should our cameras be making different kinds of measurements?
Goal: Generate "good looking" Images

There are fundamental trade-offs in quality one must make when taking an image: given that there is a fixed quantity of available light.

- Short Exposure, Small Aperture: Very little light reaching pixels, image is noisy.  
  Denoising
- Long Exposure, Small Aperture: Camera or scene might be moving, image has motion blur.  
  Motion De-blurring
- Short Exposure, Large Aperture: Not all points in scene in focus, image has defocus blur.  
  Defocus De-blurring
- Other problems: A/D converters in sensors have limited bit-depth. Lead to low-dynamic range.
- Even more fundamental: Can not capture multiple color channels at the same pixel location.
- Let's begin by talking about solving the color imaging problem.
Place a different color filter in front of each pixel.
DIGITAL COLOR IMAGING

Place a different color filter in front of each pixel.

Multiplexed Measurements

Interpolation

Loss/distortion of high-frequencies.
Loss of light due to color filters.
Multiplexed Measurements
DIGITAL COLOR IMAGING

Still use Bayer Pattern (from 1970s)

Place a different color filter in front of each pixel.

Multiplexed Measurements

Loss/distortion of high-frequencies.
Loss of light due to color filters.

Interpolation
DIGITAL COLOR IMAGING

Measure un-filtered light everywhere
Grayscale image: at higher SNR
Measure color at sparse set of locations.

[ICCP 2014]
Chakrabarti, Freeman, and Zickler, Rethinking Color Cameras
Interpolation

Fill in holes for Luminance
(easier than interpolation, much more "context")

Propagate Chromaticities Guided by Luminance Image Edges
DIGITAL COLOR IMAGING

Traditional

Proposed

Reconstructions, brightness / SNR settings.
DIGITAL COLOR IMAGING

[ICCP 2014]

Hand-designed, want to learn automatically from data!
So, thinking jointly about measurement and reconstruction helps us get to a better camera system.

The initial camera measurements are less "direct" measurements of what we want (RGB at every pixel). But are more optimal for eventual inference algorithm.

But this design is hand-crafted. Can there be better designs?

Can we "learn" these designs automatically from data?

LEARNING MULTIPLEXING PATTERN

Incident Light → Sensor Layer → Captured Image → Inference Network → Output Loss

Multiplexing Pattern

Multiplexed Measurements
LEARNING MULTIPLEXING PATTERN

Incident Light -> Sensor Layer -> Captured Image -> Inference Network

Multiplexing Pattern -> Multiplexed Measurements
Interpolate to full RGB

Output Loss
LEARNING MULTIPLEXING PATTERN

- Incident Light
- Sensor Layer
- Captured Image
- Inference Network
- Output Loss
At each pixel, measure one of four colors (red, green, blue, white).

Input Light as 4 channel Image

Sensor CFA Pattern (PxP Repeated)

P=8

Sensor Output fed to inference network
LEARNING MULTIPLEXING PATTERN

At each pixel, measure one of four colors (red, green, blue, white).

\[ x(n) \in \mathbb{R}^C \]

Pixel Intensities for C filters

Inner Product

\[ y(n) = \langle x(n), I(n) \rangle \]

Selected Intensity

\[ I(n) \in \{0, 1\}^C, \|I(n)\| = 1 \]

Selection Vector

Block for each pixel
LEARNING MULTIPLEXING PATTERN

At each pixel, measure one of four colors (red, green, blue, white).

\[ I(n) \in \{0, 1\}^C, \|I(n)\| = 1 \]

Not Differentiable!

Block for each pixel
At each pixel, measure one of four colors (red, green, blue, white). 

Learn parameters $w(n) \in \mathbb{R}^C$ 

$I(n) = \text{Soft-Max} (w(n))$ 

Block for each pixel
LEARNING MULTIPLEXING PATTERN

At each pixel, measure one of four colors (red, green, blue, white).

Learn parameters $w(n) \in \mathbb{R}^C$

$$I(n) = \text{Soft-Max} (\alpha w(n))$$

Becomes 0-1 as $\alpha \to \infty$

Increase $\alpha$ across training iterations.

Block for each pixel
LEARNING MULTIPLEXING PATTERN

Sensor Output

(3P) x (3P)

Multiplicative FC
exp( FC[log(.)] )
P x P x 3K

Conv 1x1
f(·, k)
P x P x 3K

λ(·, k)
P x P x 3K

FC + ReLU
1x1xF

Conv + ReLU
2x2xF

Conv + ReLU
3x3xF

Conv + ReLU
P x P (stride P)

K

∑
k=1

λ(·, k)f(·, k)

Reconstructed Full-color Patch

We use:

K = 24
F = 128
Experiments

Gehler-Shi Database

- RAW sensor images from camera with anti-aliasing filter
  Downsample by 2, and treat as ground-truth RGB at that resolution.
  Simulate white as sum of RGB

- 568 Images (each image has 40k 8x8 patches).
- Use 56 for testing, remaining for training & validation.

- Train / test with different levels of noise (AWGN added to sensor output).
Results

- First evaluate inference network architecture. Train it on fixed standard patterns and compare to standard interpolation methods.
LEARNING MULTIPLEXING PATTERN

Results

- First evaluate inference network architecture. Train it on fixed standard patterns and compare to standard interpolation methods.
Results

- First evaluate inference network architecture. Train it on fixed standard patterns and compare to standard interpolation methods.

- Network actually does better than traditional methods

- Also much faster. For 3MP image:
  200ms (GPU) / 9s (CPU) vs 20-60 s for traditional methods.

- Same architecture able to reason with two very different sensing strategies.
LEARNING MULTIPLEXING PATTERN

Training Pattern

Trained at specific (moderate) noise level.
LEARNING MULTIPLEXING PATTERN

Training Pattern

[ICCP 2014] VS

It # 1,500,000 (Final)
Results

Now compare learned pattern with traditional patterns, using neural network inference for all.
Results

Test Set Reconstruction Quality (median PSNR in dB)

Noise STD

Bayer
ICCP
Learned

Level used for training pattern.
Results

Ground Truth

Bayer

Learned

[ICCP]
LEARNING MULTIPLEXING PATTERN

Results

Ground Truth

Bayer

Learned

[ICCP]
LEARNING MULTIPLEXING PATTERN

Results

Ground Truth

Bayer

Learned

[ICCP]
Results

Ground Truth

Bayer

Learned

[ICCP]
LEARNING MULTIPLEXING PATTERN

Results

Bayer

Learned

[ICCP]

Ground Truth