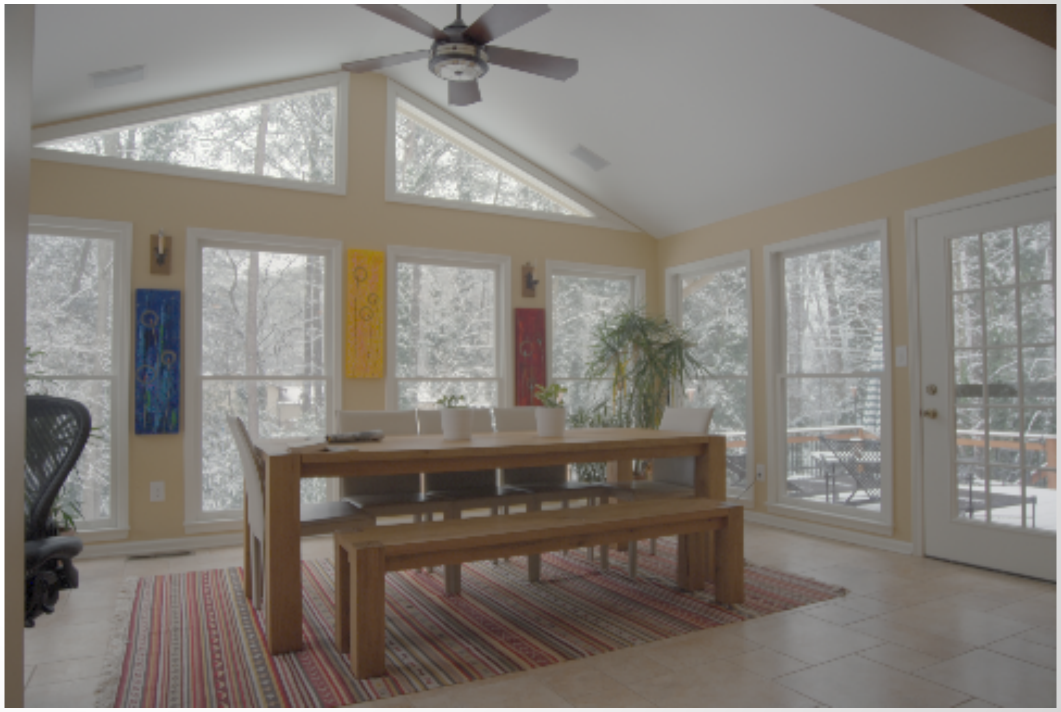
Computational Photography
Assignment #9
HDR

Ram Subramanian
Fall 2016

Result from Example Input

- Demonstrate your HDR output from your basic code, using the provided images in the input folder.

- Yes! This is pretty much the same as the reference output provided.



HDR Image Components Thumbnails

- Demonstrate thumbnails of the HDR images you chose to try (either from web or your own). **Use <u>at least</u> five images.** Use additional slides as necessary.

```
e: 1/2500.0
a: 3.615
i: 800
```

```
e: 1/160.0
a: 3.615
i: 800
```

```
e: 1/10.0
a: 3.615
i: 800
```

```
e: 1/640.0
a: 3.615
i: 800
```

```
e: 1/40.0
a: 3.615
i: 800
```

"The Marble Hall"

http://farbspiel-photo.com/learn/hdr-pics-to-play-with/marble-hall-ppw

HDR Image

- Demonstrate the result for your HDR images. If you had to correct for saturation or overflow, what did you do? *(not all image sets have this)*

Implementation: linearWeight()

```
max_intensity = 255.  # maximum intensity value of a uint8 picture
return np.min([pixel_value, max_intensity - pixel_value])
```

linearWeight(), given a pixel intensity, returns how “far away” the pixel is from the center of the intensity range.

For the valid range of intensity values (0-255), the center (or the median) is 127. So we simply return either the intensity passed in as the argument or, (255 – intensity), whichever is lower.

Implementation: sampleIntensities()

```
images = np.array(images)
num_intensities = 256
num_images = len(images)
intensity_values = np.zeros((num_intensities, num_images), dtype=np.uint8)
mid = np.round(num_images // 2)
mid_img = images[mid]
for i in range(num_intensities):
    idx = np.where(mid_img == i)
    if idx[0].size == 0:
        continue
    pick = np.random.randint(0, idx[0].size)
    intensity_values[i] = images[:, idx[0][pick], idx[1][pick]]
  return intensity_values
```

sampleIntensities(), given an image, take a sample from corresponding points in the image stack, for each valid intensity (0-255) – this is then used to build a log(exposure) to pixel value map and eventually generate a response curve to fit them together.

For convenience sake (3d array splicing), I convert images to a np.ndarray. For every possible intensity value (0-255) that I find in the mid-image, I simply find the corresponding intensity values in the other images (selecting only one at random if multiple pixels have the same intensity).

Implementation: computeResponseCurve()

```
# PART 1a
idx = 0
for i, ir in enumerate(intensity_samples):
    for j, jc in enumerate(ir):
    w_ij = weighting_function(jc)
    mat_A[idx, jc] = w_ij
    mat_A[idx, num_samples + i] = -w_ij
    mat_b[idx, 0] = w_ij * log_exposures[j]
    idx = idx + 1
# PART 1c
mat_A[-1, intensity_range // 2] = 1
```

```
# PART 1b
offset = num_images * num_samples
for idx in range(1, intensity_range):
    w = weighting_function(idx)
    mat_A[offset + idx - 1, idx - 1] = smoothing_lambda * w
    mat_A[offset + idx - 1, idx] = -2 * smoothing_lambda * w
    mat_A[offset + idx - 1, idx + 1] = smoothing_lambda * w


# PART 2
x = np.linalg.lstsq(mat_A, mat_b)[0]
```

computeResponseCurve(), given log_exposures and intensity sample, fits the exposure-intensity lines (by adjusting the radiance) for each intensity value (0-255) together to to form a smooth curve.

In order to do this we set-up two matrices A and b, A with different constraints (such as fitting, smoothing, color) and b with the weighted intensities from each image in proportion to their log(exposure) values. Finally, we solve for the response curve using least-squares (Ax=b)

Implementation: computeRadianceMap()

```
for i in range(img_shape[0]):
    for j in range(img_shape[1]):
        px = np.array([images[k][i, j] for k in range(images.shape[0])])
        wt = map(weighting_function, images[:, i, j])
        wtsum = sum(wt)
        if wtsum > 0:
            img_rad_map[i, j] = sum(wt * (response_curve[px] - log_exposure_times)) / wtsum
return img_rad_map
```

computeRadianceMap() finally computes the map of each of the pixels in the image stack to the adjusted radiance in the final HDR image. This map is used to weight pixels from each individual image from the stack to make the HDR output.

HDR Image Requirements

- Ideally all variables, except exposure time should remain constant for the greatest chance in making a good HDR image – this includes: ISO, aperture, focus, white-balance, metering, the scene (no large moving objects), etc.

- My images were taken with the bracket (-4, -2, 0, 2, 4). Essentially this means each image captured 4 times ‘as much light’ (radiance) as the previous image. This is important, especially when using linear weighting.

- The scene itself needs to be relatively constant. Having large objects or objects with high contrast to the rest of the scene moving can create problems in getting a good hdr result.

Discussion of Results

REMEMBER: We want to see **thorough answers** for each of these questions! Use complete sentences and include images/sketches (as needed) to help in your discussion & analysis.

- I think my output HDR represents the image stack very well (the window beams and the individual lights are sharp and clearly visible along with the rest of the image, such as the floor patterns, walls, chairs, etc.)
- The image could definitely be a little brighter, except in the areas where sunlight hits the floors (which could be a bit dimmer).
- The watermark at the bottom left corner didn’t come out as well – I suspect this is a combination of down sampling the original image (>95 MB each) and slight misalignments between each image. And interestingly, the intensity of the letters decreases as you move left.
- Given more time, I definitely would try to sample my own images with my camera - much of the difficulty in fact was shooting the image with the correct settings (the auto exposure bracketing feature in chdk didn’t quite play well and I didn’t have time to debug the lua scripts for it :/)

Resources

HDR picture set: *“The Marble Hall”* (http://farbspiel-photo.com/learn/hdr-pics-to-play-with/marble-hall-ppw)

Piazza post @437