HW03 Report

Original JPEG images with different exposure levels (for display purpose only):

Step 1. Merge LDR RAW into 32-bit HDR Image

`compute_mask` Function

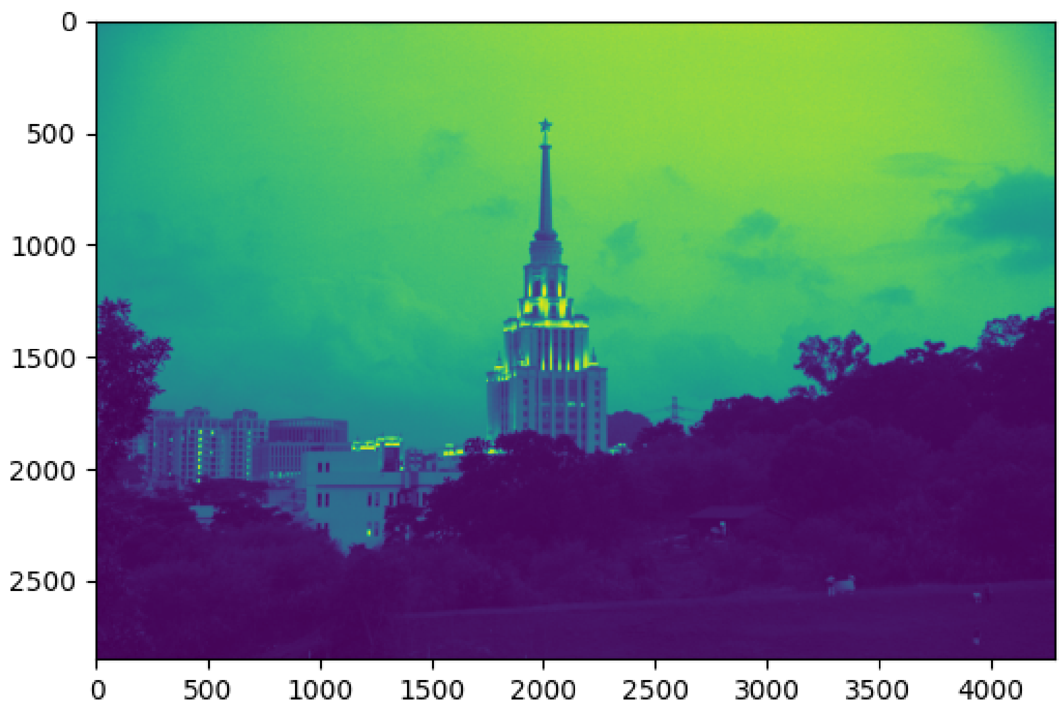
- **Purpose**: The function identifies pixels within each image that fall within the specified minimum (`min`) and maximum (`max`) intensity values.
- **Parameters**:
    - `min`: The lower threshold for pixel intensity. Pixels with intensity values equal to or greater than this threshold are considered for inclusion in the mask.
    - `max`: The upper threshold for pixel intensity. Pixels with intensity values equal to or less than this threshold are considered for inclusion in the mask.
    - `images`: A list of images. These are the images for which binary masks will be generated.
- **Process**: The function iterates through each pixel of the image. If a pixel's intensity falls within the specified `min` and `max` range (inclusive), the corresponding position in the mask is set to 1, indicating that the pixel meets the criteria for inclusion.

`get_fusion_weights` Function

- **Purpose**: Calculates weights for each pixel across images at different exposure levels.
- **Parameters**:
    - `images`: A list of images at different exposure levels.
    - `medians`: The median value of all pixel values for each image.
    - `masks`: Typically, masks are used to filter out overexposed or underexposed pixels, affecting the calculation of weights.
- **Process**: For each image, it calculates weights based on the difference between pixel values and the median of that image. Pixels closer to the median get higher weights.

`raw_exposure_fusion` Function

- **Purpose**: Creates an HDR image using a set of images at different exposure levels.
- **Parameters**:
    - `images`: A list of images at different exposure levels.
    - `medians`: The median value of all pixel values for each image.
    - `exposure_times`: Exposure times corresponding to the images in the `images` list.
    - `mask_percent`: Percentage used to compute the mask thresholds, filtering overexposed or underexposed pixels.
- **Process**:
    1. Calculates fusion weights based on the difference between pixel values and their respective image medians.
    2. Applies the equation in the lecture slide: applies weights to each image, combining logarithmic transformation and exposure time correction to composite the HDR image. This involves multiplying the logarithmic brightness values of each pixel by its weight, summing them up, and normalizing based on the sum of weights.
    3. Utilized the composited logarithmic brightness image to produce the final HDR image.



Following the raw exposure fusion process, it's evident that details are meticulously preserved in both the highlight and shadow regions, enhancing the overall depth and clarity of the image.

Step 2. Demosaic the fused raw data. Save your 32-bit HDR image into 32- bit *.EXR * file

`writeEXR` Function

- **Purpose**: Saves image data in the OpenEXR format, a high dynamic range (HDR) image file format, ensuring the preservation of high fidelity color and brightness information.
- **Parameters**:
    - `image_data`: A NumPy array containing the image data to be saved. The function expects this data to be in a 3-channel format (e.g., RGB).
    - `file_name`: The desired path and name of the output EXR file. This specifies where the file will be saved on the disk.
- **Process**:
    1. Converts the `image_data` to a 32-bit floating point format (`np.float32`), which is suitable for storing HDR content in the EXR format.
    2. Determines the height, width, and number of channels from the `image_data`. These dimensions are crucial for configuring the EXR file's header correctly.

3. Creates a new EXR file header, specifying the image dimensions and setting up each color channel (R, G, B) to use 32-bit float values. This step ensures that the EXR file can accurately represent HDR images.
4. Separates the R, G, and B channels from the `image_data`, converts each channel into a byte string, and then maps these channels correctly within the EXR file.
5. Writes the prepared pixel data to the EXR file and closes the file to finalize the saving process.

From the result, we can observe that:

EXR format supports storing a wider range of brightness and colors than traditional 8-bit or 16-bit image formats, allowing for more accurate representation of extremely bright and dark scene details. EXR files can store pixel data in 32-bit full-float numbers, offering extremely high color precision and dynamic range.

Step 3. Tone Mapping with Bilateral Filter

Tone mapping is conducted in the YUV domain, where the Y channel undergoes bilateral filtering.

`bilateral_filtering` Function

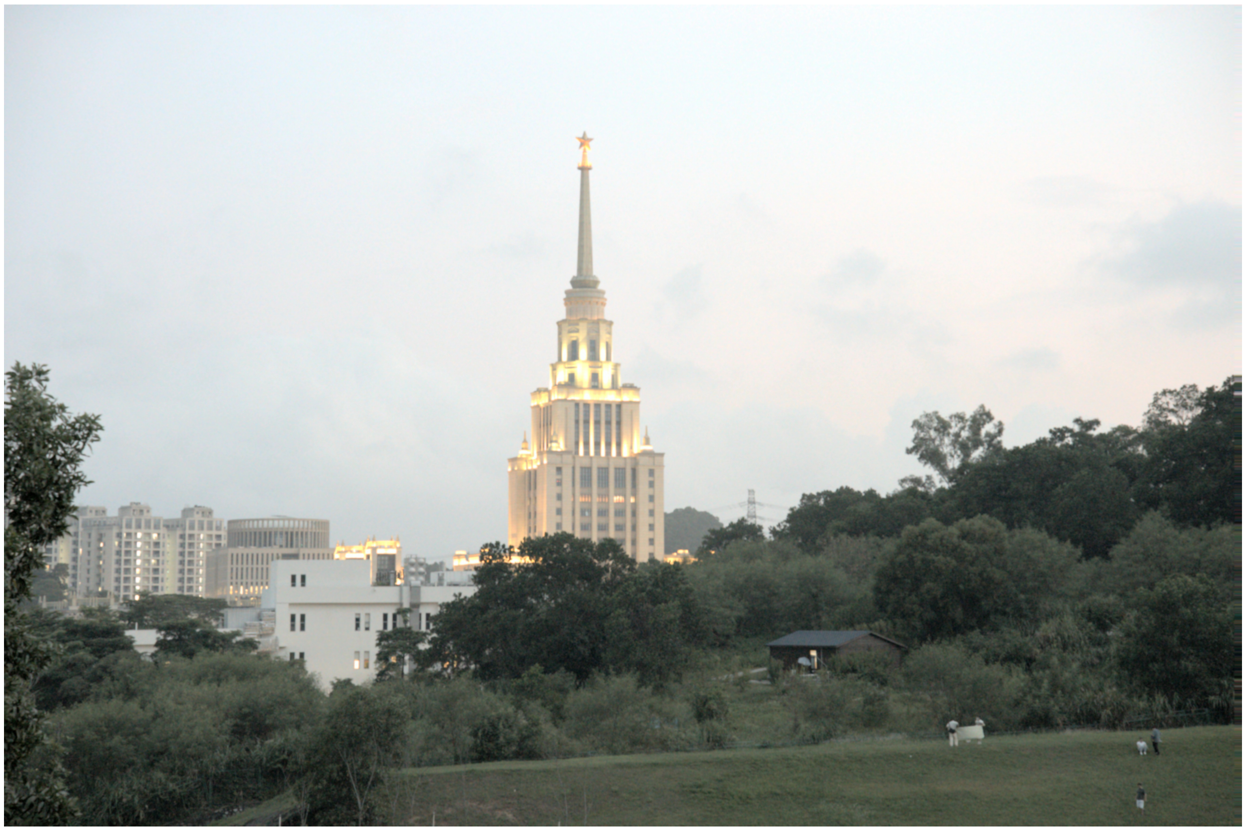
- **Purpose**: Applies bilateral filtering to an image to separate it into base and detail layers, then recalculates the image intensity using a specified gamma correction factor.
- **Parameters**:
  - `image`: The input image to be processed.
  - `range_sigma`: The sigma value for the range filter, affecting how the filter considers differences in pixel values (color/intensity).
  - `spatial_sigma`: The sigma value for the spatial filter, influencing how pixel proximity impacts the filtering.
  - `gamma`: The gamma correction factor used for adjusting the intensity of the base layer.
- **Process**:
  1. The function first applies a fast bilateral filter to separate the image into a base layer, preserving edges while smoothing other areas.
  2. It computes a detail layer by dividing the original image by the base layer.
  3. It then recalculates the new intensity of the image by applying gamma correction to the base layer and adding back the details.
  4. The resulting new intensity image is returned.

`compute_new_intensity` Function

- **Purpose**: Combines the base and detail layers of an image with gamma correction, then normalizes and scales the result to fit within a specific intensity range.
- **Parameters**:
  - `base`: The base layer of the image, obtained from bilateral filtering.
  - `detail`: The detail layer of the image, representing the ratio between the original image and its base layer.
  - `gamma`: The gamma correction factor to be applied to the base layer.
- **Process**:
  1. Applies gamma correction to the base layer and adds the detail layer to obtain the gamma-corrected intensity.
  2. Normalizes the gamma-corrected intensity to a 0-1 range.
  3. Scales the normalized intensity to the target range (16-235) for display purposes.
  4. Rounds the scaled intensity to integers and converts it to 8-bit unsigned integers (`uint8`), resulting in the final intensity image.

`fastbilateral2d` Function

- **Purpose**: Performs a fast bilateral filter on an image, effectively separating it into base and detail layers by smoothing the image while preserving edges.
- **Parameters**:
  - `image`: The input image to be filtered.
  - `range_sigma`: The sigma value for the range filter, dictating the filter's sensitivity to differences in pixel values.
  - `spatial_sigma`: The sigma value for the spatial filter, determining the influence of pixel proximity on the filtering effect.
- **Process**:
  - Utilizes OpenCV's `bilateralFilter` function to apply bilateral filtering to the input image, smoothing it while maintaining edge integrity. The filtered image (base layer) is then returned.



From the result, we can observe that:

HDR images capture a wide range of luminance levels, from deep shadows to bright highlights, which often exceed the display capabilities of standard monitors and printing devices. Tone mapping compresses this wide range into a narrower range that can be displayed on standard dynamic range (SDR) devices while still preserving visible details in both dark and bright areas.