Report of Project IMA 206 Exposure Fusion

Fengli LIN and Yutong ZHAI

***Abstract*— We implement a technique which allows us to build a high quality image using a bracketed exposure sequence, without converting to HDR first. We blend multiple exposures by using simple quality measures like saturation and contrast. This is done in a multiresolution fashion to account for the brightness variation in the sequence. The resulting image quality is comparable to existing tone mapping operators. We try the naive Exposure Fusion algorithm firstly and get an artificial result. Then we combine the Exposure Fusion algorithm with the Laplacian Pyramid, our result is much improved, but there are still some phenomenons weird. After testing and modifying the code, we finally find a specific implementation trick, which is not explicitly mentioned in the paper, to improve the result image and obtain the same result as stated in the paper.**

I. INTRODUCTION

Most everyday scenes have a far greater dynamic range than can be recorded on a photographic film or electronic imaging apparatus (whether it be a digital still camera, video, etc.). [1]However, a set of pictures, that are identical except for their exposure, collectively show us much more dynamic range than any single picture. The dark pictures show us highlight details of the scene that would be washed out in a "properly exposed" picture, while the light pictures show us some shadow detail that would also not appear in a "properly exposed" picture.

A bracketed exposure sequence allows for acquiring the full dynamic range, and can be turned into a single high dynamic range image. Upon display, the intensities need to be remapped to match the typically low dynamic range of the display device, through a process called tone mapping.

In this report, we use a technique [3]which skips the step of computing a high dynamic range image, and immediately fuses the multiple exposure into a high-quality, low dynamic range image, ready to display(like a tone-mapped picture).The idea behind this approach is that we compute a perceptual quality measure for each pixel in the multi-exposure sequence, which encodes desirable qualities, like saturation and contrast. Guided by our quality measures, we select the good pixels from the sequence and combine them into the final result.

Exposure fusion has several advantages. First of all, the acquisition pipeline is simplified, no in-between HDR image needs to be computed. Since our technique is not physically-based, we do not need to worry about calibration of the camera response curve, and keeping track of each photographs exposure time. We can even add a flash image to the sequence to enrich the result with additional detail. This approach merely relies on simple quality measures, like saturation and contrast, which prove to be very effective. Also, results can be computed at near-interactive rates, as this technique mostly relies a pyramidal image decomposition. On the downside, we cannot extend the dynamic range of the original pictures, but instead we directly produce a well-exposed image for display purposes.

II. EXPOSURE FUSION

In this part, we introduce the principle of Exposure Fusion and our way to programmer it.

A. Quality Measures

There are three quality measures we take into account: contrast,saturation and well-exposedness

- **Contrast:** We apply a Laplacian filter to the grayscale version of each image, and take the absolute value of the filter response. This yields a simple indicator C for contrast. It tends to assign a high weight to important elements such as edges and texture. Here we use *ndimage.filters.laplace* in Scipy library to realize it.
- **Saturation:** As a photograph undergoes a longer exposure, the resulting colors become desaturated and eventually clipped. Saturated colors are desirable and make the image look vivid. We include a saturation measure S, which is computed as the standard deviation within the R, G and B channel, at each pixel. Here we calculate the average value of the three channel, then accumulate each channel's variation corresponding to the average value, then average these three variation, and finally we square root it.
- **Well-exposedness:** Looking at just the raw intensities within a channel, reveals how well a pixel is exposed. We want to keep intensities that are not near zero (underexposed) or one (overexposed). We weight each in- tensity i based on how close it is to 0.5 using a Gauss curve: $exp(-\frac{(i-0.5)^2}{2\sigma^2})$ where σ equals 0.2 in our implementation. To account for multiple color channels, we apply the Gauss curve to each channel separately, and multiply the results, yielding the measure E. For this part, we also do exponential euclidean with three channel separately at first, then we multiply the three results.

For each pixel, we combine the information from the three measures above into a scalar weight map using multiplication. The reason why we choose a product rather than a linear combination is that we can enforce all qualities defined by the measures at once. Similar to weighted terms of a linear combination, we can control the influence of each measure using a power function:

$$W_{ij,k} = (C_{ij,k})^{w_C} \times (S_{ij,k})^{w_S} \times (E_{ij,k})^{w_C}$$

with C, S and E, being contrast, saturation, and well-exposedness, resp., and corresponding weighting exponents w_C, w_S and w_E, in our code we set these three values all 1. The subscript ij, k refers to pixel(i, j) in the $k-$th image. If an exponent w equals to 0, the corresponding measure is not taken into account. In order to solve this little problem, we will set a lower bound to turn these 0 values into 1. Because the value 1 won't influence the result, but the value 0 will disturb other measure's information.

B. Fusion

We will compute a weighted average along each pixel to fuse the N images, using weights computed from our quality measures. To obtain a consistent result, we normalize the values of the N weight maps such that they sum to one at each pixel (i, j) :

$$\hat{W_{ij,k}} = [\sum_{k'=1}^{N} W_{ij,k'}]^{-1} W_{ij,k}$$

The resulting image R can then be obtained by a weighted blending of the input images:

$$R_{ij} = \sum_{k=1}^{N} \hat{W_{ij,k}} I_{ij,k}$$

with Ik the k-th input image in the sequence.

III. Laplacian Pyramid

A. Intuition

As we can see from the naive fusion results, the boundary details in the naive version are not satisfactory. Wherever weights vary quickly, disturbing seams will appear. This happens because the images we are combining, contain different absolute intensities due to their different exposure times. We could avoid sharp weight map transitions by smoothing the weight map with a Gaussian filter, but this results in undesirable halos around edges, and spills information across object boundaries.

To address the problem, the authors propose to fuse differently exposed images using a Laplacian decomposition of the images and a Gaussian pyramid of the weight maps. This method turns out to avoiding seams, because it blends image features instead of intensities. Since the blending equation is computed at each scale separately, sharp transitions in the weight map can only affect sharp transitions appear in the original images (e.g. edges). Conversely, flat regions in the original images will always have negligible coefficient magnitude, and are thus not affected by possibly sharp variations in the weight function, even though the absolute intensities among the inputs could be different there.

For dealing with color images, the authors say that carrying out the blending each color channel separately produces good results.

B. Gaussian pyramid

The Gaussian pyramid is computed as follows. The original image is convolved with a Gaussian kernel. The resulting image is a low pass filtered version of the original image.

Suppose the image is represented initially by the array g_0. This image becomes the bottom or zero level of the Gaussian pyramid. Pyramid level 1 contains image g_1, which is a reduced or low-pass filtered version of g_0. Each value within level 1 is computed as a weighted average of values in level 0 within a 5-by-5 window. Each value within level 2, representing g_2, is then obtained from values within level 1 by applying the same pattern of weights. Note that in implementation, we use the $[0.05, 0.25, 0.4, 0.25, 0.05]$ kernel.

The level-to-level averaging process is performed by the function REDUCE.

$$G_{i+1}(x, y) = REDUCE(G_i(x, y))$$

which means, for levels $0 < l < N$ and nodes $i, j, 0 < i < C_1, 0 < j < R_l$,

$$g_l(i, j) = \sum_{m=2}^{2} \sum_{n=2}^{2} w(m, n) g_{l-1}(2i + m, 2j + n).$$

We now define a function EXPAND as the reverse of REDUCE. Its effect is to expand an (M + 1)-by-(N + 1) array into a (2M + 1)-by-(2N + 1) array by interpolating new node values between the given values. Thus, EXPAND applied to array g_l of the Gaussian pyramid would yield an array g_l,1 which is the same size as g_{l1}.

$$gl, n = EXPAND(g_l, n1)$$

The EXPAND operation is defined as follows:

$$G_{i+1}(x, y) = 4 \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m, n) G_i(\frac{x - m}{2}, \frac{y - n}{2})$$

C. Laplacian pyramid

[]The Laplacian [2] is then computed as the difference between the original image and the low pass filtered image. This process is continued to obtain a set of band-pass filtered images (since each is the difference between two levels of the Gaussian pyramid). Thus the Laplacian pyramid is a set of band pass filters.

$$L_i(x, y) = G_i(x, y) - EXPAND(G_{i+1}(x, y))$$

D. Fusion

Now with the help of Gaussian pyramid and Laplacian pyramid, the new fusion formula is defined as follows:

$$LR_{ij}^{l} = \sum_{k=1}^{N} G\{W\}_{ij}^{l} L\{I\}_{ij,k}^{l}$$

where $L\{A\}^l$ is defined as the l-th level in a Laplacian pyramid decomposition of an image A, $G\{B\}^l$ is defined as the l-th level in a Gaussian pyramid decomposition of an

Fig. 1. **Input images with corresponding weight maps**

image B, I is the input image and W is the corresponding weight map.

Finally, the pyramid $L\{R\}^l$ is collapsed to obtain the result R.

IV. Result and Dissuasion

A. Original Result

After programming the naive fusion, we can see from Fig.1, the input images with corresponding weight maps. A high weight means that a pixel should appear in the final image. These weights reflect desired image qualities, such as high contrast and saturation.

From Fig.2(a), we can see the result of Naive Fusion is very artificial, especially in the part of sky, there are many grids. What's worse, the color of the total image is very rough, and lots of little black points distributed all around the image, so it looks very noisy.

After applying the Laplacian Pyramid, the result is greatly improved. It is no longer noisy and the color of the whole image looks smooth and the edge's information about the figure and buildings are well preserved. However, there is a significant drawback in this image that the part of sky is so strange, which obviously isn't the final result we expect.

B. Improvement in Method

According to the algorithm proposed in the paper, we verified that the code of Laplacian Pyramid is correct, so we try to figure out if there exists a trick in the algorithm of Naive Fusion. By searching in the Internet and testing the code, we find out that it really needs a small modification in our code. Because $C_{ij,k}, S_{ij,k}, E_{ij,k}$ could be very small and when these three values are all close to 0, then the weight value of the pixel will be very unstable. In order to solve this problem, we add a constant to these three measure values. In our code we add 0.5 to each value and obtain a content result. Compared to Naive Fusion, the Fusion with Laplacian Pyramid holds more vivid color and it has more information about the edges, for example, we can see more information in the sky in the Fusion with Laplacian Pyramid.

In order to test our code, we try another set of image, in Fig.3, it shows the corresponding weight map of each original image. From Fig. 4 we see that in the version of Exposure Fusion with Laplacian Pyramid, the contrast is stronger and the edges information is clearer.

ACKNOWLEDGMENT

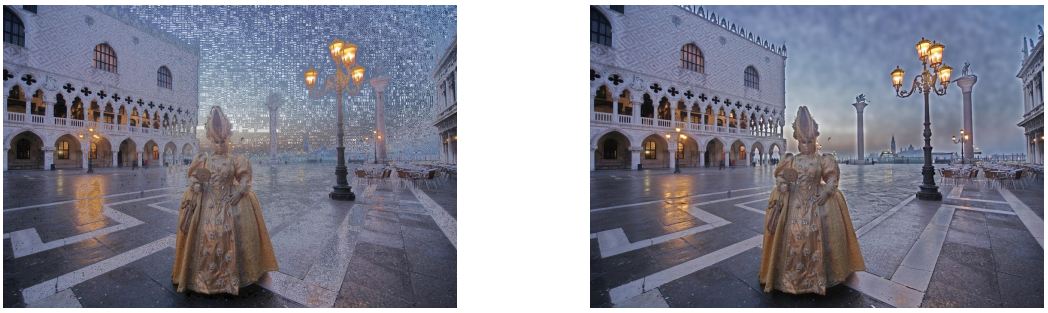
Thanks to professor Yohann Tendero for his instructions and helps.

References

[1] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. In *ACM Transactions on Graphics (ToG)*, volume 23, pages 294–302. ACM, 2004.

[2] Peter Burt and Edward Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540, 1983.

[3] Tom Mertens, Jan Kautz, and Frank Van Reeth. Exposure fusion: A simple and practical alternative to high dynamic range photography. In *Computer graphics forum*, volume 28, pages 161–171. Wiley Online Library, 2009.



(a) Original: Navie Fusion

(b) Original: Exposure Fusion with Laplacian Pyramid

(c) Improved: Navie Fusion

(d) Improved: Exposure Fusion with Laplacian Pyramid

Fig. 2. **The compare of initial method and the improved method**

Fig. 3. **Input images with corresponding weight maps**

(a) Navie Fusion

(b) Exposure Fusion with Laplacian Pyramid

Fig. 4. **The comparsion of Naive Fusion and the Exposure Fusion with Laplacian Pyramid**