EE367: A Processing Pipeline for Low-Light Burst Images

Cynthia Dalmady and Kristen Pownell

***Abstract*— A processing pipeline to align, merge, denoise, and enhance low-light images taken on iPhone 'burst' mode was proposed and tested. Patch-based and corner-based registered images were found to perform better than a single reference. Low-light illumination mapping, dehazing, and exposure fusion further enhanced results.**

I. INTRODUCTION

Phone photography has greatly improved as camera technology becomes smaller, faster, and cheaper, but it still suffers in conditions of low light. Images taken at night are often blurry, noisy, and too dark to make out important details. To improve these images, we implemented a processing pipeline including denoising and contrast enhancement algorithms. We tested different algorithms for each stage of the pipeline and determined which algorithms were best suited for different situations (indoors vs. outdoors, noisy vs. blurry, dark vs. hazy, etc).

However, even with these improvements, the final image will still fall short if the original image is too blurred to obtain any useful information. To address this problem, we took advantage of iPhone's burst mode, which captures several photos in quick succession at a low exposure. We registered the burst of images to create a whole greater than the sum of its parts. The registration process filters the images somewhat while merging them, so the merged image is already superior to the original reference image. The final merged image, therefore, is superior to the final single reference image.

II. IMAGE REGISTRATION

A. Introduction

Image registration aligns and merges a set of images into a single coherent image. It can be broadly broken down into two categories: rigid and non-rigid alignment. Rigid alignment assumes that the entire picture has been shifted by a set amount, say, three pixels in the x and y directions. Non-rigid alignment allows for warping caused by camera rotation that leads to nonuniform shifts across the image. Non-rigid alignment is much more robust but also has longer processing time than a quick global shift. We implemented both rigid and non-rigid algorithms and found that only non-rigid algorithms output an image superior to the original reference.

Registration algorithms must first choose a reference image within the burst to align other images to. We chose the sharpest image, measured as that with the highest green-channel gradient, as our reference, as in [10].

B. Rigid Alignment

For rigid alignment, we assumed a single global shift applied across an entire image. We calculated the shift by finding the minimum sum of squared distances (SSD) for different shift amounts, then shifting, adding, and averaging all images together. This performs moderately well, but if the camera has been rotated, any resulting image warp is not captured, The resulting image is still blurry, as seen in the image subsection Figure 1. Note that the full-size images are all available in the merge_results folder, or in the appendix at the end of this document.

C. Non-rigid Alignment

1) Pixel-based alignment: The simplest non-rigid alignment method loops through the entire image pixel-by-pixel, computes the minimum SSD between the window around the pixel and a series of shifted windows on the reference image, and uses those resulting shift values as a flow map to interpolate each image. We implemented a variant of this method from [7], which first globally aligned the images and used the resulting averaged image as a reference for non-rigid alignment. This algorithm is quite time-intensive because it iterates around each pixel. We reduced the window size to check around each pixel to 5x5, which does not allow for much shift and thus did not produce accurate results. A larger window size made run time untenable.

To correct for some of this misalignment, we weighted the images to favor well-aligned pixels. Here, we implemented principles from lucky imaging, based off [7]. Lucky imaging, often used in astronomy, trades off two contradictory goals: averaging over many pixels to smooth out noise, and sharpening the image by averaging over fewer, better aligned pixels. We scored each pixel on sharpness and "selectivity" to balance these goals. The sharpness weight was calculated by finding the magnitude of the Laplacian-filtered image. Specifically, for an input image $I_t(p)$, convolved with a Laplacian filter to form $L_t(p)$, $w^*_{sharp}(p) = |L_t(p)|$, where p represents a pixel. Similarly, the selectivity weight was calculated from the magnitude of the Laplacian-filtered mean image, where the mean image in a set of N images is given by $I_{mean}(p) = \frac{1}{N}\sum_{t=1}^{N} I_t(p)$. The selectivity weight was calculated from the magnitude of the Laplacian-filtered mean image: $w^*_{selectivity}(p) = |L_{mean}(p)|$. The sharpness weight was then normalized to a value between 0 and 1, and the selectivity was normalized to a weight between 0 and 10. The final weight and final image come from:

$$w(p) = w_{sharp}(p)^{w_{selectivity}(p)} \tag{1}$$

$$J(p) = \frac{\sum_{t=1}^{N} w(p) I_t(p)}{\sum_{t=1}^{N} w(p)} \quad (2)$$

However, the original misalignment was too great to be fully corrected with weighting, so the final image is still blurry. In fact, weighting adds additional artifacts that seem to outline the blurred edges. The resulting blurred image is shown in Figure 2. Overall, we cannot recommend pixel-based alignment as a usable registration technique.

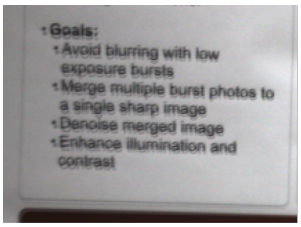


Fig. 1. Rigid registration

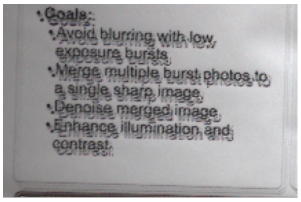


Fig. 2. Pixel-based Alignment

2) Patch-based alignment: A more sophisticated version of the simple pixel-by-pixel search comes from [10] and is currently implemented in the Google Pixel camera. This algorithm creates a pyramid by first downsampling the image by 32, aligning those patches for an initial estimate, and gradually upsampling back to a finer and more accurate alignment. The search radius starts out large and decreases as image size and computation time increase. This algorithm computes SSD normally for the smallest search radius, but for larger search radii, it instead computes the L2 norm with a linear equation to sharply decrease processing time. Indeed, the processing time for this algorithm was substantially faster than pixel-based alignment, allowing us to increase window size for more accuracy. To compute tile similarity between the reference tile T and a tile on image I shifted by (u, v) pixels, we used the distance measure:

$$D_p(u,v) = \sum_{y=0}^{n-1} \sum_{x=0}^{n-1} |T(x,y) - I(x+u+u_0, y+v+v_0)|^p \quad (3)$$

where (u_0, v_0) is the shift given by the previous pyramid level and p is the power of the norm. Rather than calculating D_p for each shift value on each tile, we instead use a single equation that calculates D_p for all shifts at once:

$$D_2 = ||T||_2^2 + box(I \circ I, n) - 2(\mathcal{F}^{-1}\{\mathcal{F}\{I\}^* \circ \mathcal{F}\{T\}\}) \quad (4)$$

To find the (u, v) value that minimizes this equation, we approximate this as a least square problem defined as:

$$D_2(u,v) \approx \frac{1}{2}[u\ v]\mathbf{A}\begin{bmatrix} u \\ v \end{bmatrix} + \mathbf{b}^T \begin{bmatrix} u \\ v \end{bmatrix} + c \quad (5)$$

We can use a set of 3x3 filters to efficiently solve this weighted least-squares system. Details are available in the supplement for [10]. The solution to the least-squares system gives the optimal shift distance for the tile.

To merge the tiles while correcting for misalignment, we used a temporal method similar to Weiner deconvolution. Merging the tiles in the frequency domain averages out less prominent frequencies likely to be noise. The weighting supplied by Weiner deconvolution further suppresses any mismatch that might come from noise or misalignment. Rather than simply averaging, then, we use the following equation to merge a given tile $T_x(\omega)$ with a reference tile $T_0(\omega)$:

$$T_0(\omega) = \frac{1}{N} \sum_{x=0}^{N-1} T_x(\omega) + A_x(\omega)[T_0(\omega) - T_x(\omega)] \quad (6)$$

$A_x(\omega)$ is defined similarly to the Weiner filter:

$$A_x(\omega) = \frac{|T_0(\omega) - T_z(\omega)|^2}{|T_0(\omega) - T_z(\omega)|^2 + c\sigma^2} \quad (7)$$

with σ^2 estimated at 0.1 and c fixed at a constant value. The choice of c greatly changes how similar the output is to the reference image. A low c will cause almost all non-reference tiles to be rejected and lead to a nearly exact copy of the reference image at the output. Increasing c increases tolerance to shifting errors, so more tiles are merged with the original image. The averaging of these additional tiles decreases noise but also increases blur. We tested several different values for c and found that, as in the original paper, c = 8 provided a good trade-off between blur and noise reduction.

This algorithm performs substantially better than the pixel-based alignment algorithm. The blur of the merged image is higher than the reference image, but the noise reduction increases image quality over the reference image. Indeed, for our seven tested image sets, patch-based alignment produced an average 9.3% quality increase. The resulting image is shown in Figure 3. Please see the results section for details on image metrics used.

3) Corner-based alignment: Our final non-rigid algorithm, based off [8],was further optimized for fast phone applications by finding only a sparse correspondence between two images. Like the previous algorithm, it also divides the image up into tiles, but rather matching the entire tile, it identifies a single "corner" on the tile to match to the reference corner. "Cornerness" is computed as the difference in luminance around a point. Specifically, we choose 9 "grid points" on a tile and divide up the area around the grid point into four quadrants. We then compute the average change in luminance from quadrant to quadrant. For grid point p where the average luminance in each surrounding quadrant j is given by μ_j, the cornerness value is given by:

$$C(p) = \sum_{j=1}^{4} |\mu_{mod(j+1,4)} - \mu_j| \quad (8)$$

To save time, we only compute corners on the reference tile. We then search around the corresponding point on each additional image to find the closest match to the reference corner (evaluated with SSD). Like the previous algorithm, this algorithm also uses a pyramidal structure to identify

corners at each level and better initialize search locations at the next level of the pyramid.

This algorithm produces a very sparse flow. We used 21x21 tiles and chose three corners from a 3x3 grid in each, which only provides information for 1/7 of the pixels in the image. To compute displacement at every pixel, not only the corner pixels, we cross-bilaterally filtered the flow maps, P_u and P_v, to spread out the corner shifts. We weighted these filtered maps with a third map meant to give more precedence to original corner locations. This third map is given by:

$$N(p) = \begin{cases} 1, & p \text{ is a corner} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

For filtered corner map $\tilde{P}_f$, the final flow map F is given by $\tilde{P}_f/\tilde{N}$, where $\tilde{N}$ is the filtered normalization map. This dilutes the original shift, so we then scaled the flow map to have the same maximum shift as the original corner map. To spread the corner map out as far as possible, we used a very large spatial standard deviation, $\sigma_s = 400$. We also used a large search radius to ensure that every pixel would be able to access at least one corner, $r = 50$. This large search radius substantially slowed down the algorithm. The corner-matching was quite fast, but the bilateral filtering took nearly as long as the pixel-by-pixel search. The paper authors originally implemented their algorithm on an Nvidia microprocessor, which doubtless performs loops faster than Matlab. If the filtering could be sped up, the speed of this would hopefully be comparable to the patch-based search or even faster because it only chooses a few points to match from each tile.

After interpolating the images with the filtered flow map, we weighted them before the final merge. The weights were based off exposure fusion, discussed in detail later in this paper. Exposure fusion calculates weights from contrast, saturation, and well-exposedness. For this algorithm, we added a fourth weight, the structural similarity index (SSIM), to further weed out mismatch [13].

The corner-based algorithm was able to accurately recreate the reference image, but it was not able to substantially improve upon it as in patch-based alignment because the it does not add additional denoising. This algorithm was originally implemented for HDR processing, so the authors were mainly focused on matching an HDR stack taken at different exposures rather than improving upon an original image. Overall, it is not entirely suited for our application.

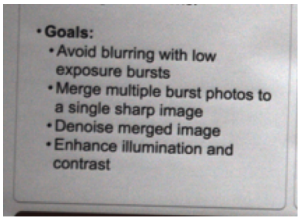


Fig. 3. Patch-based registration

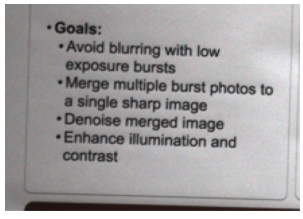


Fig. 4. Corner-based registration

D. Results

We had two main metrics to determine our image quality: blur and noise. Perfect alignment downweights non-reference images to result in low blur, but having fewer images to average over results in low noise reduction. Our registration algorithms in general produce slightly blurrier but significantly less noisy images than the reference. To measure noise, we used a Matlab implementation of [11], which finds low-textured patches of the image from the gradient and uses them to estimate overall image noise standard deviation. This assumes a white Gaussian noise model, which we assumed was close enough to accurate because of our low exposure time. To measure sharpness, we used the Tenengrad measure. This convolves the image with a Sobel kernel, which emphasizes edges, and finds the magnitude of the resulting gradient. A higher Tenengrad measure denotes a larger gradient and sharper edges. To express the trade-off between noise and blur, we use an overall image quality metric given by:

$$Q = \frac{T^2}{\sigma} \tag{10}$$

where Q is the overall quality, T is the Tenengrad measure, and σ is the estimated noise standard deviation. Results are shown in Table 1 below for several different types of images. (1) is a computationally blurred and shifted image; (2) is a high-detail indoor image taken with a RAW burst; (3) is a low-light indoor image taken with a regular, .jpg burst; and (4) is an outdoor low-light image taken with a RAW burst. We wanted to test both RAW and .jpg images because only iOS 10 has the capability to take RAW images, and they can currently only be captured from certain apps. Users without these apps would need to take a regular burst, which would be processed and stored as .jpg files. Fortunately, we saw similar performance on the RAW and .jpg images.

Clearly, patch-based performs the best from all our algorithms. The only image for which it does not exceed the reference is the computationally shifted image, which does not exactly match conditions of real life and thus should be of less importance than performance on natural images. Corner-based performs exactly as well or just slightly better than the reference because it is excellent at recreating the reference but does not include additional denoising. Going forward, we used the reference, patch-based, and corner-based images to test contrast enhancement.

TABLE I
REGISTERED IMAGE QUALITY

Algorithm	(1)	(2)	(3)	(4)
Reference	1.285	14.634	0.312	0.957
Rigid alignment	0.613	10.459	0.343	0.158
Pixel alignment	0.611	2.179	0.139	0.273
Patch-based	1.263	16.567	0.370	0.966
Corner-based	1.285	14.634	0.312	0.958

III. DENOISING

Although our registration methods smoothed out some noise, we needed further denoising to improve the images. Because all our images were captured with a short exposure time, we assumed that the noise was primarily Gaussian white noise. With more time, we would have also searched for denoising measures for signal-dependent Poisson noise, which becomes comparable to Gaussian noise in low-light situations.

A. Non-local Means

As done in class, the non-local means algorithm (NLM) finds similar windows across an image to average out noise rather than simply averaging the nearby pixels [1]. This preserves image detail and prevents over-smoothing. We used the fast implementation of non-local means from the previous homework for this filtering. It performed adequately well but still left some noise. The resulting denoised reference image is shown in Figure 5.

B. Collaborative Non-local Means

Collaborative non-local means is a variant on classic NLM that uses information from multiple images to better smooth out a reference image [3]. Rather than only finding windows on the reference image, it also finds windows on each additional image, and uses a combination of all windows to denoise the reference. This differs from our earlier registration algorithms because it does not actually try to combine the images; rather, it uses the combined information from all images to denoise only the reference. To implement this, we modified the previously mentioned fast NLM script. In most cases, it performed better than the original, non-denoised image, but additional denoising smoothed out too many details and made the image quality suffer. The resulting image after CNLM is shown in Figure 6.

C. BM3D

Our final and most effective denoising algorithm tested was Block-Matching and 3D Filtering (BM3D) [5]. Like NLM, BM3D searches for similar "blocks" within an image. It then stacks these blocks and filters them in 3D to obtain a sparse representation. This filtered block is further filtered with a shrinkage operator (such as Weiner filtering) to attenuate noise. The reconstructed blocks have lower noise but still preserve original detail.

To implement this, we used code released by the Tampere University of Technology. The results outperformed normal NLM processing and provided the least noisy images. However, for images that were less noisy to begin with, it over-smoothed details. We had to choose our denoising algorithm on a case-by-case basis to trade off detail preservation and noise attenuation. The resulting filtered reference image is shown in Figure 7.

D. Results

We used the same image quality measure described above to quantify the trade-off between blur increase and noise reduction for these images. We found that overall, BM3D greatly improved image performance for most low-detail images. CNLM did not perform as well as expected, most likely because it had already averaged out noise that was then re-averaged and blurred in future processing. Results are shown in Table 2 for the same set of images tested in Table 1: (1) computationally generated; (2) indoor, high detail; (3) indoor, low-light; (4) outdoor, low-light.

CNLM performs better than the non-denoised reference in most cases, but after going through BM3D, it is over-smoothed and lower-quality than the single-filtered reference. NLM and CNLM both cannot compare to BM3D because they would filter out too much detail for an equivalent loss in noise. For the extremely noise outdoor image (4), NLM actually makes the image worse, because the detail reduction is not matched by an adequate noise reduction. For high-detail images such as the computationally generated image, denoising does decrease the noise by a factor of more than 10, but the quality decrease is too high to justify the blurring. Overall, NLM provided an average quality increase of 32% across the reference, feature-based, and patch-based pictures, and BM3D provided an average quality increase of 190% across reference, patch-based, feature-based, and CNLM.

TABLE II
DENOISED IMAGE QUALITY

Method	Ref	Patch	Corner	CNLM
(1)				
-	1.285	1.263	1.285	1.280
NLM	0.750	0.744	0.750	0.750
BM3D	1.921	1.965	1.921	1.923
(2)				
-	14.634	16.567	14.634	17.872
NLM	17.535	18.143	17.535	17.398
BM3D	44.970	48.477	44.970	43.730
(3)				
-	0.312	0.370	0.312	0.671
NLM	0.873	1.052	0.873	1.013
BM3D	1.700	1.754	1.699	1.683
(4)				
-	0.957	0.966	0.958	0.792
NLM	0.594	0.610	0.596	0.581
BM3D	3.986	4.305	3.982	3.959



Fig. 5. Noisy reference image



Fig. 6. Denoised with NLM



Fig. 7. Denoised with CNLM



Fig. 8. Denoised with BM3D

From the results on the registration and denoising algorithms, we determine that the patch algorithm with BM3D denoising produces the best results in regards to sharpness and noise. Therefore, we use the merged image formed by the patch algorithm and BM3D for the image enhancement section of our pipeline.

IV. IMAGE ENHANCEMENT

A. Introduction

Once we have merged and denoised the burst images, the resulting low exposure image is dark and has low dynamic range. To improve the overall image quality, we first enhance the overall of illumination of the low-light image to obtain a high exposure image of the scene. We then combine this enhanced exposure with the low exposure, merged image to generate a high quality, tone-mapped LDR image. Next, we apply a dehazing algorithm and a global gamma correction to improve the saturation and contrast of hazy and washed-out images.

B. LIME

Directly amplifying low light images may improve the visibility of dark regions, but it will also oversaturate the bright regions in the image, losing contrast and detail. Gamma correction is another possibility since it nonlinearly amplifies intensities, but it does not take into consideration the relationship between the pixels, resulting in somewhat inconsistent scenes. Instead, we investigated the Retinex theory, which models an image (L) as the element-wise multiplication of two factors, reflectance (R) and illumination (T). We can then recover a light-enhanced image by solving for the reflectance through element-wise division.

$$\mathbf{L} = \mathbf{R} \circ \mathbf{T}, \tag{11}$$

The Low-Light Image Enhancement via Illumination Map Estimation, or LIME, is the algorithm we implemented for estimating the illumination map (T)[12]. As opposed to other Retinex algorithms, LIME only estimates one factor, the illumination, shrinking computational costs. The paper additional describes a faster estimation of the solution to further improve speed. While the sped up algorithm was not implemented here, this algorithm shows promise for mobile phone applications.

We initially estimate T by finding the maximum value of each color channel for each pixel. This estimation guarantees that the recovery will not be saturated.

$$\hat{\mathbf{T}}(x) \leftarrow \max_{c \in \{R,G,B\}} \mathbf{L}^c(x), \tag{12}$$

The algorithm then iteratively solves for the minimization problem in Eq. 12 by using an augmented Lagrangian multiplier (ALM) based algorithm. The goal of the solution is to minimize the error of the illumination map while simultaneously preserving the overall structure through the weighted gradient matrix.

$$\min_{\mathbf{T},\,\mathbf{G}} \|\hat{\mathbf{T}} - \mathbf{T}\|_F^2 + \alpha \|\mathbf{W} \circ \mathbf{G}\|_1 \quad \text{s.t. } \nabla \mathbf{T} = \mathbf{G}. \tag{13}$$

The augmented Lagrangian can be written as follows. Each term can be individually updated to converge to a solution.

$$\mathcal{L} = \|\hat{\mathbf{T}} - \mathbf{T}\|_F^2 + \alpha \|\mathbf{W} \circ \mathbf{G}\|_1 + \Phi(\mathbf{Z}, \nabla \mathbf{T} - \mathbf{G}), \tag{14}$$

While there are a few options for weighing the illumination gradient, for the best compromise of performance and simplicity, we utilize the gradient of the initial illumination map estimate $\hat{T}$ as our weights.

$$\mathbf{W}_{dx}(x) \leftarrow \frac{1}{|\nabla_{dx} \hat{\mathbf{T}}(x)| + \epsilon}; \; \mathbf{W}_{dy}(x) \leftarrow \frac{1}{|\nabla_{dy} \hat{\mathbf{T}}(x)| + \epsilon}. \tag{15}$$

A summary of the algorithm is shown here, and further details of the derivation may be found in Guo's paper [12].

1) Initialize α, $\hat{T}$, $u(0) > 0, \rho > 1$ and weights W
While (not converged) do:
2) Update T^{t+1}

$$\mathbf{T}^{t+1} \leftarrow \mathcal{F}^{-1}\left(\frac{\mathcal{F}\left(2\hat{\mathbf{T}} + \mu^{(t)}\mathbf{D}^T\left(\mathbf{G}^{(t)} - \frac{\mathbf{Z}^{(t)}}{\mu^{(t)}}\right)\right)}{\mathbf{2} + \mu^{(t)} \sum_{d \in \{h,v\}} \overline{\mathcal{F}(\mathbf{D}_d)} \circ \mathcal{F}(\mathbf{D}_d)}\right) \tag{16}$$

3) Update G^{t+1}

$$\mathbf{G}^{(t+1)} = \mathcal{S}_{\frac{\alpha \mathbf{W}}{\mu^{(t)}}}\left[\nabla \mathbf{T}^{(t+1)} + \frac{\mathbf{Z}^{(t)}}{\mu^{(t)}}\right]. \tag{17}$$

4) Update Z^{t+1}

$$\begin{aligned} \mathbf{Z}^{(t+1)} &\leftarrow \mathbf{Z}^{(t)} + \mu^{(t)}(\nabla \mathbf{T}^{(t+1)} - \mathbf{G}^{(t+1)}); \\ \mu^{(t+1)} &\leftarrow \mu^{(t)} \end{aligned} \tag{18}$$

5) Gamma correction on T via $T = T^\gamma$. The higher the Gamma, the greater the enhancement. Values from 0.3-0.5 produce the most natural looking results.
6) Recover enhanced image

$$\mathbf{R} = \mathbf{L}/\mathbf{T}, \tag{19}$$

We measure the quality of the enhancement by calculating the lightness order error (LOE). The relative order of lightness represents the light source directions and the lightness variation, so the naturalness of an enhanced image is related to the relative order of lightness in different local areas. A lower number indicates better image quality. We calculate LOE using the merged image as the reference image. In Eq.20, the function $U(p, q)$ returns 1 if $p \leq q$, 0 otherwise,

and the circle-and-cross symbol stand for the exclusive-or logic operation:

$$LOE = \frac{1}{m}\sum_{x=1}^{m}\sum_{y=1}^{m}\Big(U(\mathbf{Q}(x),\mathbf{Q}(y)) \oplus U(\mathbf{Q}_r(x),\mathbf{Q}_r(y))\Big), \tag{20}$$

The performance of the LIME algorithm, shown in Figure 9, is quite impressive. Even in mostly dark images, the original scene can still be recovered. However due to the lack of information in ultra-low light scenes, the maximum color value in dark areas is amplified, resulting in some enhanced images with unnatural color saturation. The images of the cliff, lamp, and chair are single-image test images, while the bicycle image is the enhancement on a merged image using the patch algorithm and BM3D denoising.

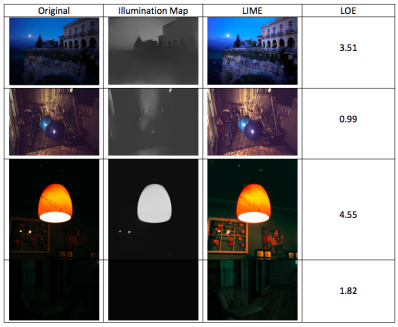


Fig. 9. lightness order error measure comparison of LIME algorithm

C. Exposure Fusion

Even though we take the burst images with the same exposure, we can combine our high-exposure, light-enhanced image with the low-exposure merged image to generate an image with high dynamic range. Instead of using a traditional HDR algorithm which combines the different exposures into an HDR image that must be later tone-mapped to a low dynamic range for displays, the exposure fusion algorithm skips the intermediate HDR image and immediately results in a well-exposed, LDR image. Therefore, it can be computationally efficient for mobile phone applications and is currently used for Google's HDR+. This exposure fusion algorithm was also used to calculate weights for corner-based image merging.

We weight each pixel in each image corresponding to three metrics: contrast, saturation, and well-exposedness. We calculate each as follows:

- **Contrast**: Apply a Laplacian filter to the grayscale version of the image to assign high weights to edges and textures for contrast.
- **Saturation**: Saturation is computed as the standard deviation within each color channel. This will apply low weights to over-exposed areas with desaturated colors and clipping.
- **Well-exposedness**: We assume a well-exposed pixel has an intensity of approximately 0.5, so we weight each intensity based on a Gaussian curve with mean 0.5 and sigma 0.5: $\exp\frac{-(i-0.5)^2}{2\sigma^2}$. We apply this to each color channel individually and multiply them together.

For each pixel, the weights are the products of the weighted quality metrics. We can adjust the weights to emphasize one metric over the other. We use $\omega_C = 2$, $\omega_S = 1$, and $\omega_E = 1.5$. The weights are then normalized to obtain consistent results.

$$W_{ij,k} = (C_{ij,k})^{\omega_C} \times (S_{ij,k})^{\omega_S} \times (E_{ij,k})^{\omega_E}$$

$$\hat{W}_{ij,k} = [\sum_{k'=1}^{N} W_{ij,k'}]^{-1} W_{ij,k} \tag{21}$$

Once we have weights for each pixel in each image, we fuse the images utilizing Gaussian and Laplacian pyramids as described [2]. First, we calculate the Gaussian weight and image pyramids at each level by downsampling the images by a factor of 2 for each pyramid level. Next, we calculate the Laplacian image pyramids by calculating the difference between each pyramid level and the upsampled interpolation of the pyramid one level above. We fuse the images into a new Laplacian pyramid by multiplying the Gaussian weights and Laplacian image pyramids and summing over the images for each pyramid level. We can then reconstruct this pyramid by reversing the procedure used to generate the Laplacian pyramids, thus obtaining a high quality LDR image.

$$\mathbf{L}\{R\}_{ij}^{l} = \sum_{k=1}^{N} \mathbf{G}\{\hat{W}\}_{ij,k}^{l} \mathbf{L}\{I\}_{ij,k}^{l} \tag{22}$$

The resulting HDR image improves upon the contrast and increases the dynamic range of the original image. The images of the cliff, and street-light are single-image test images, while the bicycle image is the enhancement on a merged image using the patch algorithm and BM3D denoising. However, since we are artificially creating an exposure bracket, there are sometimes dark halo artifacts in the merged image, as can been see in the bicycle HDR image. We have been able to reduce or eliminate the dark halos by limiting the gamma of the light enhancement to 0.3-0.5. If the light-enhanced image is too bright compared to the merged image, though, the images will not blend as nicely. A summary of the results is shown in Figure 10. We notice that the Tenengrad measure actually decreases from the LIME enhanced image to the HDR image, even though the image quality seems to improve. It is possible that the fusion of the pyramids is introducing a small error or blurring that is not perceivable.

D. Dehazing

Due to the low light conditions, the resulting image may still look washed out. In order to improve the contrast and saturation of the image, we applied a dehazing algorithm to the image. Dehazing algorithms are normally applied on

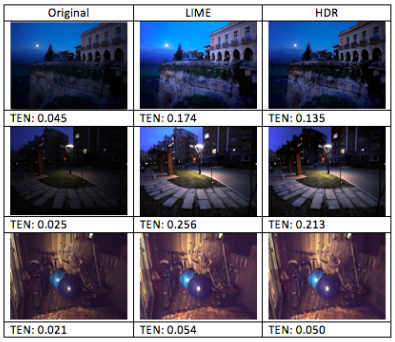


Fig. 10. Tenengrad measure comparison of HDR fusion algorithm

hazy-outdoor images, where haze, fog or smoke, can grey out sections of the image at greater distances. However, we wanted to investigate whether the same techniques could be applied to the enhanced but low-contrast low-light images to similarly improve the saturation and contrast. The original dehazing algorithm, described in [4], implemented a dark channel prior based on outdoor haze statistics. These statistics showed that hazy regions often have very low intensity in at least one color channel and that these "dark pixels" can be used to estimate the haze transmission.

A hazy image I can be modeled as a combination of the scene radiance J, the global atmospheric light A, and the medium transmission t. The goal of dehazing is to recover $J, A,$ and t.

$$\mathbf{I}(\mathbf{x}) = \mathbf{J}(\mathbf{x})t(\mathbf{x}) + \mathbf{A}(1 - t(\mathbf{x})) \tag{23}$$

We first calculate the dark channel prior by iterating over small windows centered at pixel x and finding the minimum color intensity within the color channels in this window. If the image is haze free, the dark channel should be zero. However, if there is haze in the image, these sections will appear as grey in the dark channel. We then estimate the atmospheric light by picking the top 0.1% brightest pixels in the dark channel. From these pixels we then pick the pixel with the highest intensity in the input image. With this algorithm, we should be able to pick out the brightest haze, while ignoring white-colored objects in the scene.

$$J^{\text{dark}}(\mathbf{x}) = \min_{\mathbf{y}\in\Omega(\mathbf{x})} \left(\min_{c\in\{r,g,b\}} J^c(\mathbf{y}) \right) \tag{24}$$

After calculating the dark channel prior and atmospheric light, we can estimate the transmission using Eq.24 and use this transmission to recover the radiance J, which we use as the final, dehazed image.

$$\tilde{t}(\mathbf{x}) = 1 - \omega \min_{\mathbf{y}\in\Omega(\mathbf{x})} \left(\min_{c} \frac{I^c(\mathbf{y})}{A^c} \right) \tag{25}$$

$$\mathbf{J}(\mathbf{x}) = \frac{\mathbf{I}(\mathbf{x}) - \mathbf{A}}{\max(t(\mathbf{x}), t_0)} + \mathbf{A} \tag{26}$$

However, this algorithm is limited by two main issues. First, the window size when calculating the dark channel prior greatly determines the image quality. If the window is too small, the image becomes oversaturated. If the windows are too large, the resulting image may have halos near the depth edge. The paper [4] introduces an algorithm that will filter these halos to produce a well saturated image with little to no artifacts. However, this algorithm is fairly time consuming. Instead, we reduce the window size to the minimum 3x3 to reduce or eliminate the halos and adjust for the oversaturation by applying a global gamma correction of around 0.85 to each pixel in the image. The gamma correction desaturates the image slightly to help return the image saturation to more natural look.



Fig. 11. Tenengrad measure comparison of dehazing algorithm

A summary of both hazy and low light images is shown in Figure 11. The images of the canyon and tree are single-image test images, while the bicycle and robot images are the enhancements on merged images using the patch algorithm and BM3D denoising. The algorithm works great on hazy, outdoor images and some low-light images. However, if the low-light image contains a large white region (as in the robot image), it will incorrectly assume that this section is haze and will try to correct it, resulting in unnatural saturation of the image. Since the dehazing algorithm is fine-tuned to work with outside, hazy images rather than dark, indoor images, we must conclude that it is not a reliable algorithm to improve the saturation and contrast of our low light images. Further research must be done on tone mapping algorithms that will suit our needs better.

V. RESULTS

Overall, we were successful in creating a processing pipeline for low-light burst images. We found that merging multiple images, while slightly increasing blur, provided a

beneficial noise reduction. Further denoising produced an image suitable for contrast enhancement. We were consistently able to illuminate the image without over-saturating it, although further enhancements such as dehazing and exposure fusion were only successful under certain conditions.

For this project, we focused primarily on fast applications that could be run on a phone. For future research, we could look into slower but more accurate merging methods that might provide sharpness enhancement along with noise reduction. Our pipeline would also benefit from a sharpness enhancement stage, which would improve registered and non-registered images. Additionally, we could extend our algorithm to cover live photos as well as burst photos, thus taking advantage of another little-used iPhone feature.

REFERENCES

[1] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel, "A non-local algorithm for image denoising." IEEE Computer Science Conference on Computer Vision and Pattern Recognition, June 2005.

[2] Burt and Adelson, "The Laplacian Pyramid as a Compact Image Code," IEEE Transactions on Communications, Vol. COM-31, no. 4, April 1983, pp. 532-540.

[3] Geng Chen, Pei Zhang, Yafeng Wu, Dinggang Shen, and Pew-Thian Yap, "Collaborative non-local means denoising of magnetic resonance images." IEEE International Symposium on Biomedical Imaging, 2015.

[4] Kaiming He, Jian Sun, Xiaoou Tang, "Single-image haze removal using dark channel prior." IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(12), Dec 2011.

[5] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian, "Image denoising with block-matching and 3D filtering." IEEE Transactions on Image Processing, 16(8), 2007.

[6] Tom Mertens, Jan Kautz, and Frank Van Reeth, "Exposure Fusion." Pacific Conference on Computer Graphics and Applications, 2007.

[7] Neel Joshi and Michael Cohen, "Seeing Mt. Rainier: Lucky imaging for multi-image denoising, sharpening, and haze removal." IEEE International Conference on Computational Photography, 2010.

[8] Orazio Gallo, Alejandro Troccoli, Jun Hul, Kari Pulli, and Jan Kautz, "Locally Non-rigid Registration for Mobile HDR Photography." IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015.

[9] Said Pertuza, Domenec Puiga, and Miguel Angel Garcia, "Analysis of focus measure operators for shape-from-focus." Journal of Pattern Recognition 46(5), May 2013.

[10] Samuel Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan Barron, Florian Kainz, Jiawen Chen, and Marc Levoy, "Burst photography for high dynamic range and low-light imaging on mobile cameras." ACM Transactions on Graphics, 35(6), Nov 2016.

[11] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi, "Noise level estimation using weak textured patches of a single noisy image." IEEE International Conference on Image Processing, 2012.

[12] Xiaojie Guo, Yu Li, and Haibin Ling, "LIME: Low-Light Image Enhancement via Illumination Map Estimation," IEEE Transactions on Image Processing, 26(2), Feb 2017.

[13] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity." IEEE Transactions on Image Processing 13(4), April 2004.

APPENDIX 1

Image Quality Metrics

Tenegrad Measure, Registration

	I_ref	I_dumb_merge	I_global	I_pixel_unweight	I_pixel_weighted	I_cnlm	I_feature	I_patch
SKY								
Normal	0.172	0.094	0.151	0.104	0.156	0.170	0.172	0.170
NLM	0.169	0.092	0.149	0.099	0.147	0.168	0.169	0.167
BM3D	0.166	0.090	0.147	0.092	0.134	0.165	0.166	0.164
ROBOT								
Normal	0.055	0.027	0.042	0.029	0.039	0.049	0.055	0.053
NLM	0.047	0.024	0.038	0.025	0.033	0.045	0.047	0.046
BM3D	0.042	0.022	0.035	0.023	0.030	0.041	0.042	0.042
KETTLE								
Normal	0.067	0.024	0.051	0.033	0.055	0.057	0.067	0.066
NLM	0.054	0.018	0.043	0.027	0.045	0.051	0.054	0.054
BM3D	0.049	0.017	0.039	0.025	0.040	0.046	0.049	0.048
BIKE								
Normal	0.189	0.043	0.051	0.049	0.107	0.168	0.189	0.186
NLM	0.078	0.012	0.020	0.018	0.035	0.074	0.078	0.077
BM3D	0.031	0.010	0.017	0.015	0.020	0.031	0.031	0.031
BENCH								
Normal	0.204	0.035	0.041	0.038	0.100	0.204	0.204	0.200
NLM	0.080	0.004	0.011	0.008	0.024	0.080	0.080	0.078
BM3D	0.021	0.003	0.009	0.007	0.010	0.021	0.021	0.021
POSTER								
Normal	0.368	0.136	0.228	0.224	0.262	0.358	0.368	0.365
NLM	0.350	0.125	0.216	0.212	0.238	0.345	0.350	0.347
BM3D	0.335	0.117	0.207	0.195	0.220	0.330	0.335	0.332
MEMCHU								
Normal	0.346	0.062	0.142	0.101	0.222	0.345	0.346	0.340
NLM	0.196	0.028	0.104	0.063	0.126	0.196	0.196	0.193
BM3D	0.118	0.024	0.090	0.053	0.083	0.118	0.118	0.117

Noise, Registration

	I_ref	I_dumb_merge	I_global	I_pixel_unweight	I_pixel_weighted	I_cnlm	I_feature	I_patch
SKY								
Normal	0.005433	0.002934	0.003130	0.009239	0.026963	0.003857	0.005433	0.004399
NLM	0.003334	0.001504	0.001899	0.005042	0.015843	0.003185	0.003334	0.002856
BM3D	0.001677	0.000851	0.001023	0.001900	0.011678	0.001586	0.001677	0.001512
ROBOT								
Normal	0.009693	0.005172	0.005105	0.005968	0.011190	0.003525	0.009693	0.007542
NLM	0.002522	0.001228	0.001480	0.001700	0.005532	0.002041	0.002522	0.002015
BM3D	0.001049	0.000843	0.000866	0.000853	0.003224	0.001012	0.001049	0.000989
KETTLE								
Normal	0.006451	0.003930	0.003902	0.005168	0.012608	0.002564	0.006451	0.005509
NLM	0.002061	0.000933	0.001313	0.001578	0.008457	0.001816	0.002061	0.001863
BM3D	0.000793	0.000709	0.000715	0.000818	0.005881	0.000783	0.000793	0.000764
BIKE								
Normal	0.037200	0.016993	0.016616	0.017072	0.041596	0.035731	0.037164	0.035637
NLM	0.010300	0.000964	0.001112	0.001284	0.012998	0.009540	0.010256	0.009725
BM3D	0.000240	0.000016	0.000038	0.000112	0.002358	0.000242	0.000240	0.000220
BENCH								
Normal	0.040288	0.016870	0.016531	0.016854	0.042431	0.040219	0.040288	0.038610
NLM	0.012442	0.000793	0.000972	0.001337	0.015282	0.012422	0.012442	0.011727
BM3D	0.000743	0.000014	0.000042	0.000249	0.005357	0.000743	0.000743	0.000705
POSTER								
Normal	0.009252	0.004918	0.004986	0.007992	0.031380	0.007169	0.009252	0.008040
NLM	0.006981	0.001968	0.003028	0.002874	0.020652	0.006860	0.006981	0.006646
BM3D	0.002493	0.001038	0.001144	0.004706	0.010814	0.002493	0.002493	0.002280
MEMCHU								
Normal	0.093066	0.032302	0.033126	0.032909	0.080402	0.092912	0.093066	0.091750
NLM	0.051439	0.003485	0.007604	0.006654	0.046363	0.051369	0.051439	0.050158
BM3D	0.007213	0.000065	0.001388	0.001168	0.013235	0.007195	0.007214	0.006958

Quality, Registration

	I_ref	I_dumb_merge	I_global	I_pixel_unweight	I_pixel_weighted	I_cnlm	I_feature	I_patch
SKY								
Normal	5.431	3.011	7.310	1.170	0.897	7.454	5.431	6.578
NLM	8.531	5.631	11.690	1.930	1.356	8.821	8.531	9.793
BM3D	16.352	9.521	20.994	4.427	1.537	17.093	16.351	17.869
ROBOT								
Normal	0.312	0.138	0.343	0.145	0.139	0.671	0.312	0.370
NLM	0.873	0.453	0.965	0.382	0.198	1.013	0.873	1.052
BM3D	1.700	0.582	1.399	0.644	0.272	1.683	1.699	1.754
KETTLE								
Normal	0.696	0.142	0.667	0.207	0.237	1.286	0.696	0.781
NLM	1.433	0.356	1.403	0.451	0.240	1.444	1.433	1.544
BM3D	2.968	0.396	2.170	0.736	0.274	2.720	2.966	3.017
BIKE								
Normal	0.957	0.110	0.158	0.141	0.273	0.792	0.958	0.966
NLM	0.594	0.147	0.372	0.251	0.094	0.581	0.596	0.610
BM3D	3.986	5.776	7.856	1.965	0.176	3.959	3.982	4.305
BENCH								
Normal	1.032	0.071	0.102	0.086	0.234	1.031	1.032	1.038
NLM	0.511	0.020	0.116	0.048	0.037	0.510	0.511	0.519
BM3D	0.606	0.491	1.884	0.170	0.019	0.606	0.606	0.627
POSTER								
Normal	14.634	3.763	10.459	6.302	2.179	17.872	14.634	16.567
NLM	17.535	7.916	15.475	15.649	2.749	17.398	17.535	18.143
BM3D	44.970	13.282	37.585	8.050	4.463	43.730	44.970	48.477
MEMCHU								
Normal	1.285	0.120	0.613	0.308	0.611	1.280	1.285	1.263
NLM	0.750	0.226	1.415	0.605	0.340	0.750	0.750	0.744
BM3D	1.921	8.762	5.771	2.414	0.523	1.923	1.921	1.965

Light Order Error, LIME

	l_ref	l_feature	l_patch
SKY			
Normal	2.822	2.822	2.806
NLM	2.799	2.803	2.796
BM3D	2.646	2.651	2.653
ROBOT			
Normal	2.248	2.247	2.183
NLM	1.881	1.885	1.884
BM3D	1.791	1.790	1.788
KETTLE			
Normal	1.061	1.061	1.024
NLM	1.051	1.051	1.053
BM3D	1.128	1.129	1.128
BIKE			
Normal	2.576	2.576	2.577
NLM	2.006	2.006	1.988
BM3D	1.009	0.999	0.996
BENCH			
Normal	3.733	3.732	3.733
NLM	2.518	2.518	2.479
BM3D	0.358	0.361	0.355
POSTER			
Normal	3.038	3.038	3.044
NLM	3.285	3.285	3.295
BM3D	3.310	3.310	3.304
MEMCHU			
Normal	2.648	2.645	2.641
NLM	3.037	3.036	3.035
BM3D	2.949	2.956	2.954

Tenegrad, HDR

	I_ref	I_feature	I_patch
SKY			
Normal	0.241	0.241	0.238
NLM	0.234	-	0.231
BM3D	0.229	0.229	0.227
ROBOT			
Normal	0.119	0.119	0.113
NLM	0.095	0.095	0.092
BM3D	0.085	0.085	0.083
KETTLE			
Normal	0.094	0.094	0.092
NLM	0.074	0.074	0.073
BM3D	0.067	0.067	0.066
BIKE			
Normal	0.358	0.358	0.352
NLM	0.140	0.140	0.138
BM3D	0.050	0.050	0.049
BENCH			
Normal	0.406	0.406	0.400
NLM	0.161	0.161	0.158
BM3D	0.034	0.034	0.034
POSTER			
Normal	0.422	0.422	0.419
NLM	0.400	0.400	0.397
BM3D	0.382	0.382	0.379
MEMCHU			
Normal	0.540	0.540	0.532
NLM	0.288	0.288	0.283
BM3D	0.157	0.157	0.156

Tenengrad, Dehazed

	I_ref	I_feature	I_patch
SKY			
Normal	0.202	0.202	0.199
NLM	0.200	-	0.197
BM3D	0.189	0.189	0.187
ROBOT			
Normal	0.102	0.101	0.095
NLM	0.077	0.077	0.075
BM3D	0.066	0.066	0.065
KETTLE			
Normal	0.095	0.095	0.090
NLM	0.065	0.065	0.064
BM3D	0.054	0.054	0.053
BIKE			
Normal	0.385	0.385	0.379
NLM	0.157	0.157	0.154
BM3D	0.039	0.039	0.039
BENCH			
Normal	0.448	0.448	0.442
NLM	0.195	0.195	0.191
BM3D	0.033	0.033	0.032
POSTER			
Normal	0.451	0.451	0.446
NLM	0.423	0.423	0.419
BM3D	0.401	0.401	0.400
MEMCHU			
Normal	0.539	0.539	0.532
NLM	0.301	0.301	0.296
BM3D	0.162	0.162	0.161

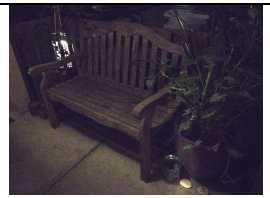
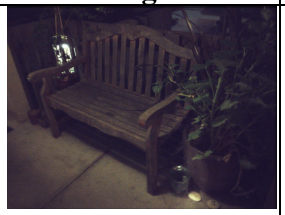
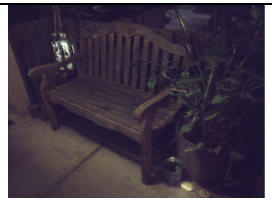
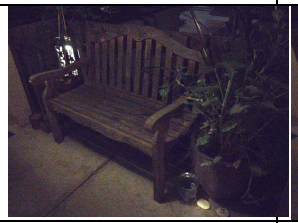
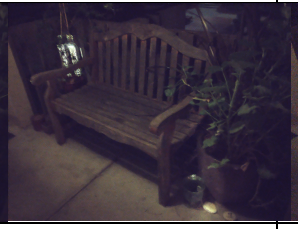
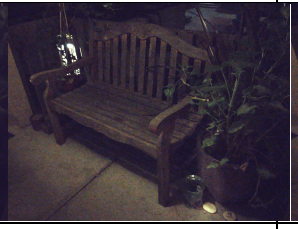
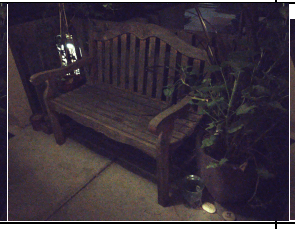
Tenengrad, Tone-mapping

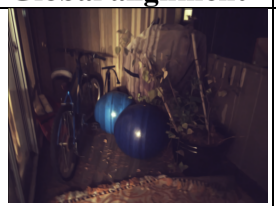
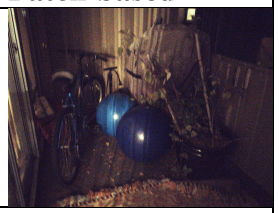
	I_ref	I_feature	I_patch
SKY			
Normal	0.210	0.210	0.207
NLM	0.210	-	0.207
BM3D	0.196	0.196	0.194
ROBOT			
Normal	0.117	0.117	0.109
NLM	0.087	0.087	0.085
BM3D	0.076	0.076	0.074
KETTLE			
Normal	0.097	0.097	0.093
NLM	0.066	0.066	0.065
BM3D	0.055	0.055	0.055
BIKE			
Normal	0.426	0.426	0.420
NLM	0.173	0.173	0.170
BM3D	0.041	0.041	0.041
BENCH			
Normal	0.490	0.490	0.485
NLM	0.215	0.215	0.211
BM3D	0.033	0.033	0.033
POSTER			
Normal	0.442	0.442	0.437
NLM	0.415	0.415	0.410
BM3D	0.392	0.392	0.391
MEMCHU			
Normal	0.587	0.587	0.578
NLM	0.316	0.316	0.310
BM3D	0.162	0.162	0.161

APPENDIX 2

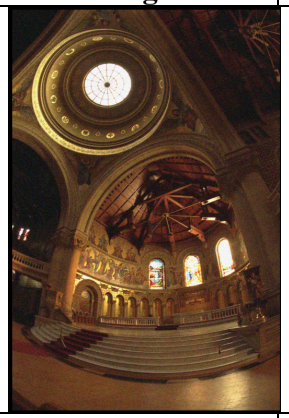
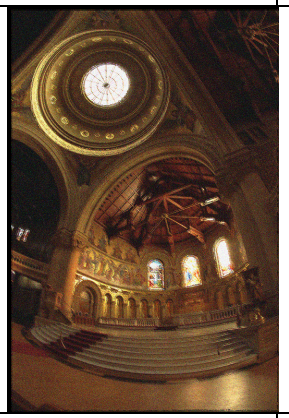
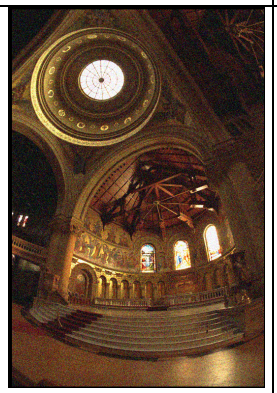
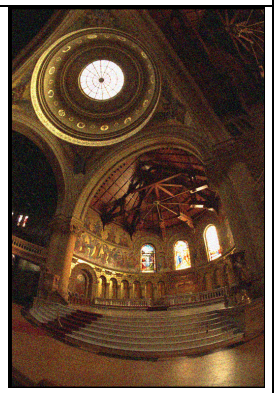
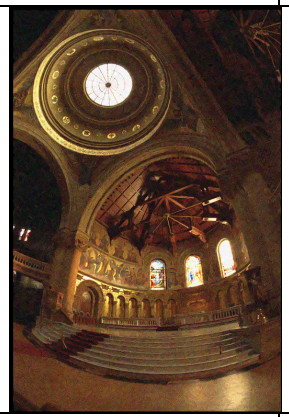
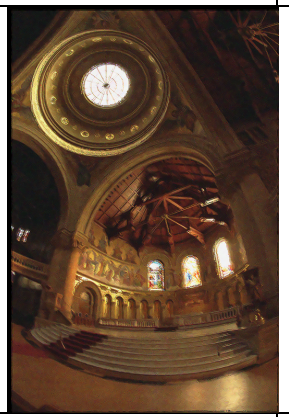
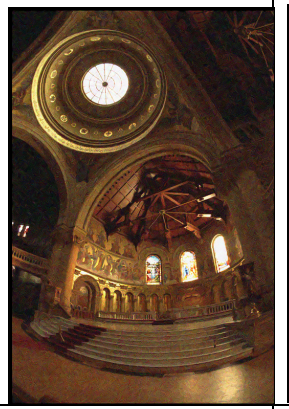
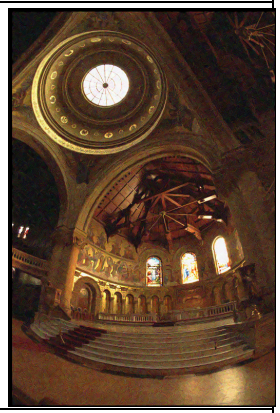
Merged Images

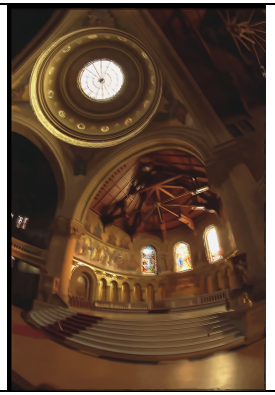
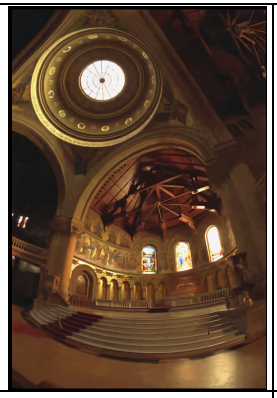
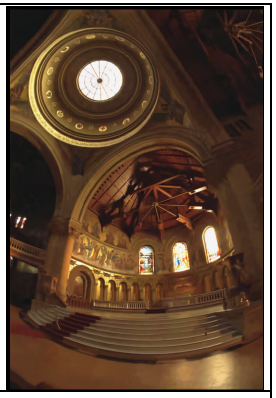
Note: For quality metrics shown, N = noise standard deviation, T = Tenengrad measure, and Q = quality index

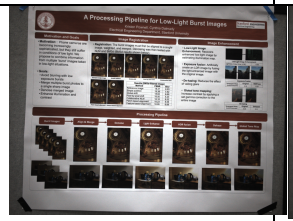
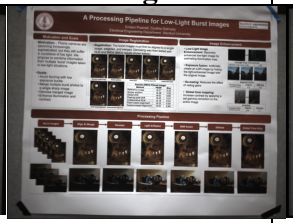
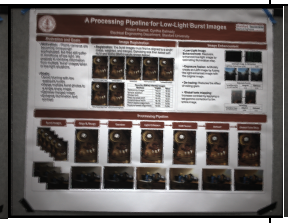
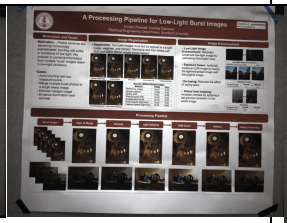
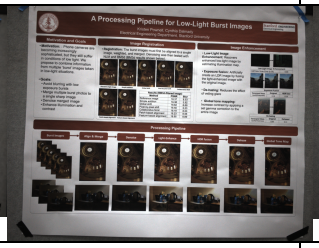
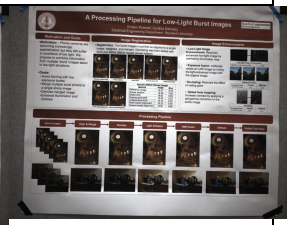
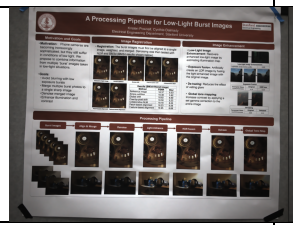
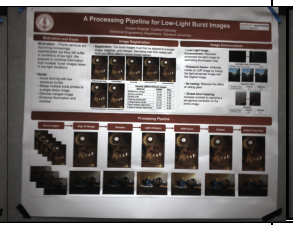
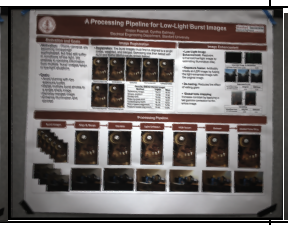
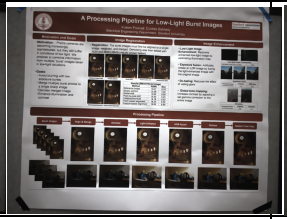
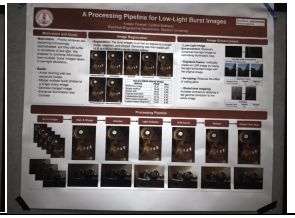
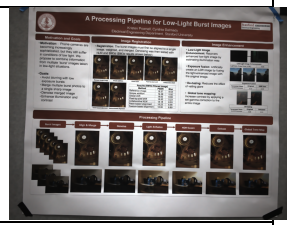
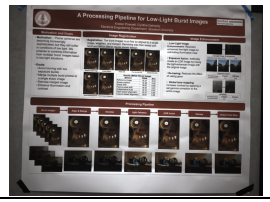
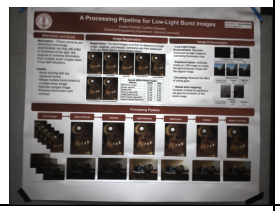
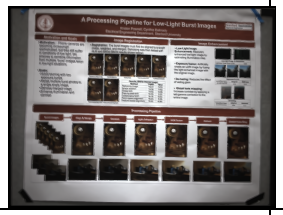
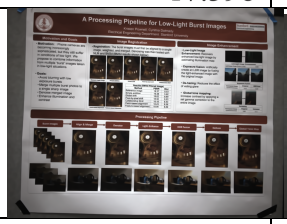
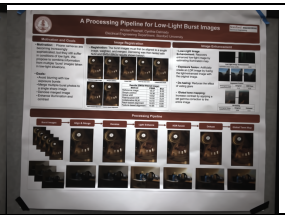
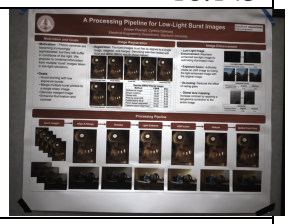
BENCH IMAGE (Outdoor, low-light)						
	Reference	**Global alignment**	**Pixel-based**	**CNLM**	**Feature-based**	**Patch-based**
Merged						
N	0.040288	0.016531	0.042431	0.040219	0.040288	0.038610
T	0.204	0.041	0.100	0.204	0.204	0.200
Q	1.032	0.102	0.234	1.031	1.032	1.038
NLM						
N	0.012442	0.000972	0.015282	0.012422	0.012442	0.011727
T	0.080	0.011	0.024	0.080	0.080	0.078
Q	0.511	0.116	0.037	0.510	0.511	0.519
BM3D						
N	0.000743	0.000042	0.005357	0.000743	0.000743	0.000705
T	0.021	0.009	0.010	0.021	0.021	0.021
Q	0.606	1.884	0.019	0.606	0.606	0.627

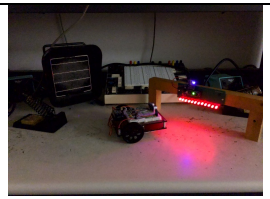
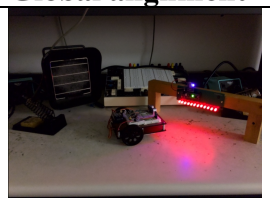
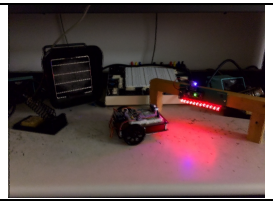
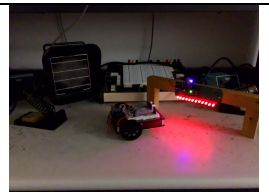
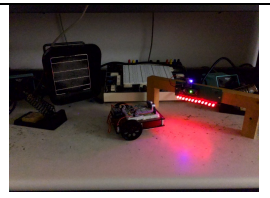
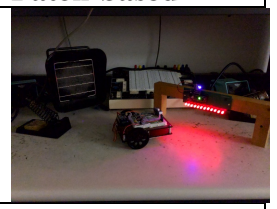
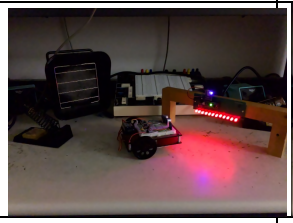
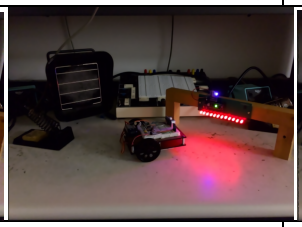
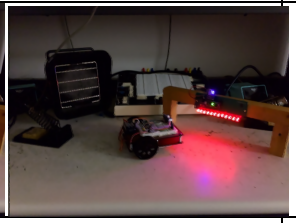
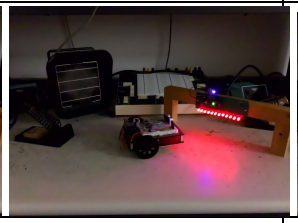
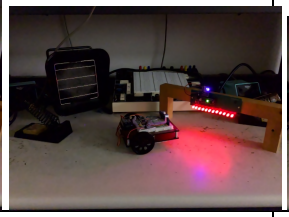
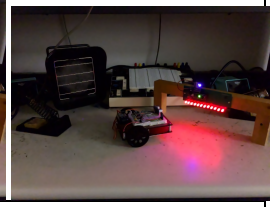
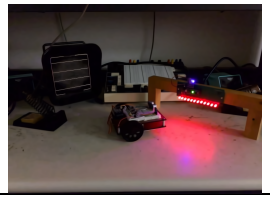
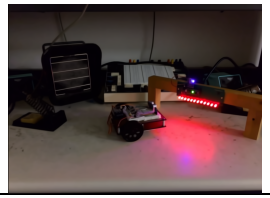
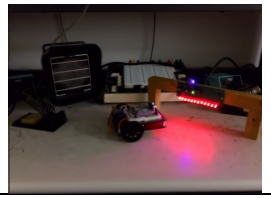
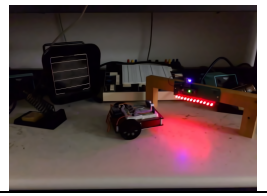
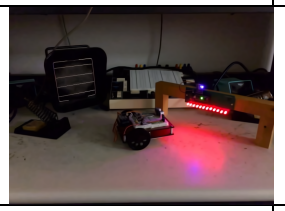
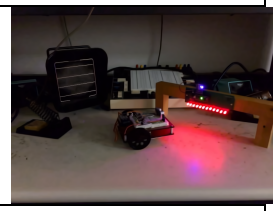
BIKE IMAGE (Outdoor, low-light)						
	Reference	**Global alignment**	**Pixel-based**	**CNLM**	**Feature-based**	**Patch-based**
Merged						
N	0.0372	0.0166	0.0416	0.0357	0.0372	0.0356
T	0.1887	0.0512	0.1066	0.1682	0.1887	0.1856
Q	0.957	0.158	0.273	0.792	0.958	0.966
NLM						
N	0.0103	0.00111189	0.01299808	0.00954019	0.01025575	0.00972524
T	0.0782	0.0203	0.0350	0.0744	0.0782	0.0770
Q	0.594	0.372	0.094	0.581	0.596	0.610
BM3D						
N	0.0002398	0.00003837	0.00235761	0.00024192	0.00024009	0.00021975
T	0.0309	0.0174	0.0203	0.0309	0.0309	0.0308
Q	3.986	7.856	0.176	3.959	3.982	4.305

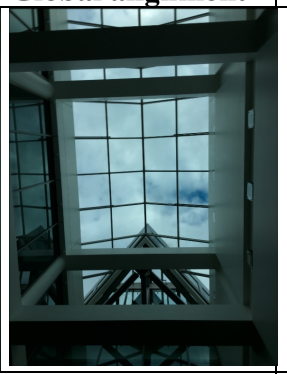
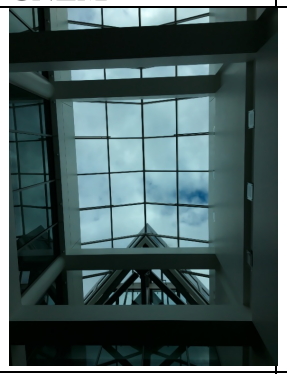
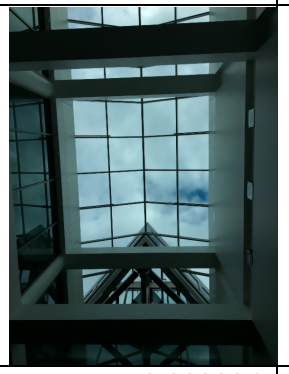
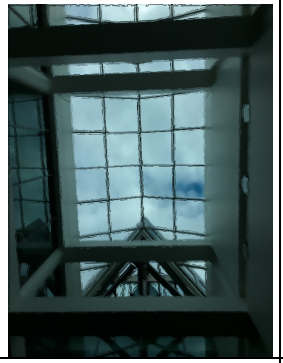
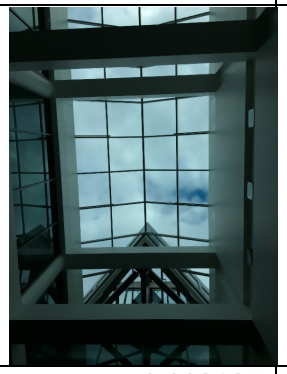
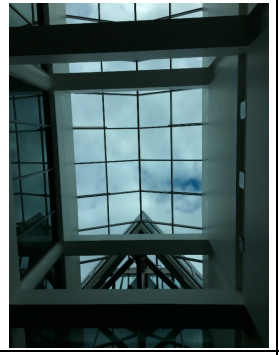
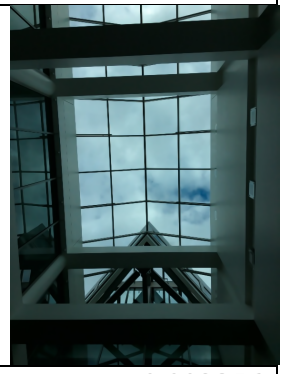
KETTLE IMAGE (Indoor, low-light)						
	Reference	**Global alignment**	**Pixel-based**	**CNLM**	**Feature-based**	**Patch-based**
Merged						
N	0.006451	0.003902	0.012608	0.002564	0.006451	0.005509
T	0.067	0.051	0.055	0.057	0.067	0.066
Q	0.696	0.667	0.237	1.286	0.696	0.781
NLM						
N	0.002061	0.001313	0.008457	0.001816	0.002061	0.001863
T	0.054	0.043	0.045	0.051	0.054	0.054
Q	1.433	1.403	0.240	1.444	1.433	1.544
BM3D						
N	0.000793	0.000715	0.005881	0.000783	0.000793	0.000764
T	0.049	0.039	0.040	0.046	0.049	0.048
Q	2.968	2.170	0.274	2.720	2.966	3.017

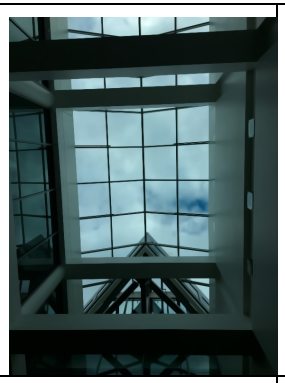
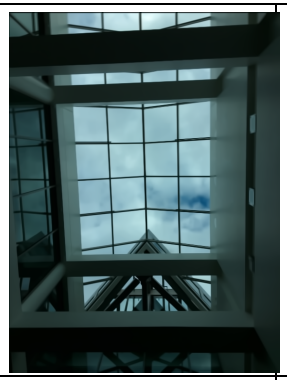
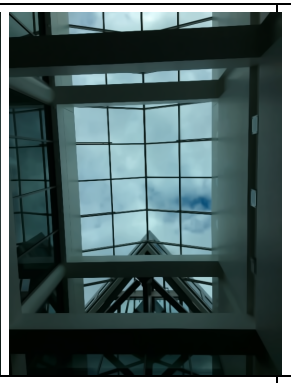
MEMORIAL CHURCH IMAGE (Computationally generated)						
	Reference	**Global alignment**	**Pixel-based**	**CNLM**	**Feature-based**	**Patch-based**
Merged						
N	0.093066	0.033126	0.080402	0.092912	0.093066	0.091750
T	0.346	0.142	0.222	0.345	0.346	0.340
Q	1.285	0.613	0.611	1.280	1.285	1.263
NLM						
N	0.051439	0.007604	0.046363	0.051369	0.051439	0.050158
T	0.196	0.104	0.126	0.196	0.196	0.193
Q	0.750	1.415	0.340	0.750	0.750	0.744

BM3D						
N	0.007213	0.001388	0.013235	0.007195	0.007214	0.006958
T	0.118	0.090	0.083	0.118	0.118	0.117
Q	1.921	5.771	0.523	1.923	1.921	1.965

POSTER IMAGE (Indoors, high-detail)						
	Reference	**Global alignment**	**Pixel-based**	**CNLM**	**Feature-based**	**Patch-based**
Merged						
N	0.009252	0.004986	0.031380	0.007169	0.009252	0.008040
T	0.368	0.228	0.262	0.358	0.368	0.365
Q	14.634	10.459	2.179	17.872	14.634	16.567
NLM						
N	0.006981	0.003028	0.020652	0.006860	0.006981	0.006646
T	0.350	0.216	0.238	0.345	0.350	0.347
Q	17.535	15.475	2.749	17.398	17.535	18.143
BM3D						
N	0.002493	0.001144	0.010814	0.002493	0.002493	0.002280
T	0.335	0.207	0.220	0.330	0.335	0.332
Q	44.970	37.585	4.463	43.730	44.970	48.477

ROBOT IMAGE (Indoors .jpg burst, low-light)						
	Reference	**Global alignment**	**Pixel-based**	**CNLM**	**Feature-based**	**Patch-based**
Merged						
N	0.009693	0.005105	0.011190	0.003525	0.009693	0.007542
T	0.055	0.042	0.039	0.049	0.055	0.053
Q	0.312	0.343	0.139	0.671	0.312	0.370
NLM						
N	0.002522	0.001480	0.005532	0.002041	0.002522	0.002015
T	0.047	0.038	0.033	0.045	0.047	0.046
Q	0.873	0.965	0.198	1.013	0.873	1.052
BM3D						
N	0.001049	0.000866	0.003224	0.001012	0.001049	0.000989
T	0.042	0.035	0.030	0.041	0.042	0.042
Q	1.700	1.399	0.272	1.683	1.699	1.754

SKY IMAGE (Indoors .jpg burst, well-lit)						
	Reference	**Global alignment**	**Pixel-based**	**CNLM**	**Feature-based**	**Patch-based**
Merged						
N	0.005433	0.003130	0.026963	0.003857	0.005433	0.004399
T	0.172	0.151	0.156	0.170	0.172	0.170
Q	5.431	7.310	0.897	7.454	5.431	6.578
NLM						
N	0.003334	0.001899	0.015843	0.003185	0.003334	0.002856
T	0.169	0.149	0.147	0.168	0.169	0.167
Q	8.531	11.690	1.356	8.821	8.531	9.793

BM3D						
N	0.001677	0.001023	0.011678	0.001586	0.001677	0.001512
T	0.166	0.147	0.134	0.165	0.166	0.164
Q	16.352	20.994	1.537	17.093	16.351	17.869