Improving low-light performance through burst photography on mobile cameras

Myles A. Gavic
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
gavic007@morris.umn.edu

ABSTRACT

The compact structure of today's smart phones has limited the size of cameras these devices can use. With most smart phones utilizing a small camera, this diminishes the amount of light gathered during capture. When taking an image in dimly lit scenes or when the amount of light captured is low, the resulting photograph will contain image noise and low dynamic range. This paper discusses methods for producing images through computational photography pipelines. Developed for mobile devices, these pipelines capture, align, and merge a burst of multiple images in order to produce a single high-dynamic-range and low noise image. Similar pipelines have existed for a while in the forms of High-Dynamic-Range (HDR) and Image Denoising algorithms. This paper will present a recent development in computational photography pipelines called Burst Photography for use in both high-dynamic-range and low-noise imagery.

Keywords

Computational photography, high dynamic range, RAW

1. INTRODUCTION

The ability to shoot in burst mode is a setting found on most newer cell phone cameras. Burst mode, or burst, allows the user to capture multiple images in rapid succession by either pressing or holding down the shutter button. Recently, burst capturing has become ubiquitous in many hand-held imaging devices (e.g., smartphone, compact and DSLR cameras). For example, the iPhone 5 supports a burst of up to 10 shots per second [7].

The purpose of burst mode is to allow users to select the best images of a subject in motion. Another common application of burst mode is its use in high-dynamic-range (HDR) photography. HDR photography consists of capturing multiple images at varying light settings (exposures) and merging them into a single image. As shown in Figure 1, HDR photography is useful when a scene contains large contrasts between highlights and shadows. This particular method of HDR is known as *bracketing* and is a setting used in most of today's smart phone cameras. While HDR can produce realistic looking images, the power needed to process the images can be strenuous on weaker processors [3] leading to a rapid loss in battery life. While burst mode accounts for scene motion and HDR, it still has the issue of causing graininess (image noise) when capturing in low light. An ideal solution would involve capturing the images in HDR while performing a noise reduction process. In recent years, Google has developed such a solution for their Nexus line of phones known as burst photography [4].

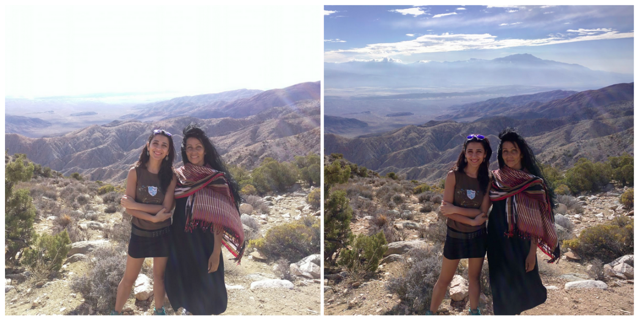
(a) (b)

Figure 1: An example of a scene taken with and without HDR enabled. (a) shows how without HDR, details in the sky and shadows are not visible. (b) shows that adding HDR brings out the detail throughout the scene and preserves the color. Image courtesy of Marc Levoy [6].

Burst Photography [4] is a computational photography pipeline developed as an improvement upon HDR+ capture mode found Googles' Nexus line of phones. *Burst photography* takes several images of a scene at the same exposure and merges them together. This process produces a single image that contains less noise and sharper detail than without HDR turned on as illustrated in Figure 1.

I will begin this paper by providing some background information on dynamic-range and image noise. Next, I will cover the general process of the burst photography pipeline and end by discussing a comparison in image performance between burst photography and a similar burst capture pipeline known as pixel fusion.

2. BACKGROUND

Dynamic range and *noise* are two major concerns to deal with when taking a photograph. If a scene is in a darker setting, then any image taken will contain noise. Noise occurs when the sample size of light photons hitting the camera's sensor is very small. For instance, one pixel might register

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/.
UMM CSci Senior Seminar Conference, April 2017 Morris, MN.

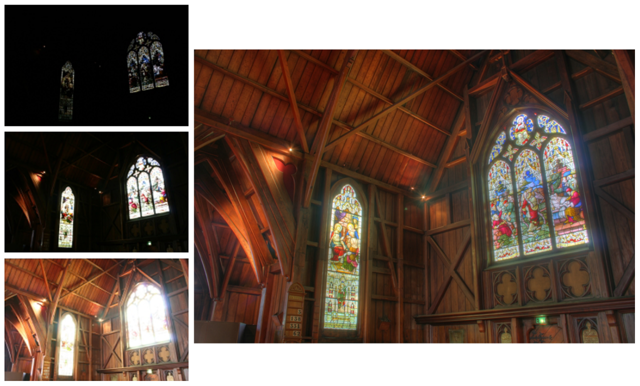

Figure 2: An example of bracketing exposure by taking multiple images beginning with an underexposed image (top left) and working towards an overexposed image (lower left) and then merging the images together. The purpose of this is to capture detail in the shadows and highlights of the frame. [10].

twice the light value of the pixels next to it, causing image noise. Due to the random nature of light photons hitting the sensor, noise in an image will be unique in every image from pixel to pixel [9]. Dynamic range is a span of light values ranging from the brightest highlights (full white) to the darkest shadows (full black). Similar to how a human eye opens and closes between bright and dark areas, a camera lens will open to pick up detail in the dark scenes and close to preserve detail in very bright scenes. However, unlike human eyes, current cameras lack the capability of capture both ends of the dynamic range spectrum. To account for this, some cameras use a stack biased High-Dynamic-Range imaging pipeline to widen the camera's light capture capabilities.

Stack-based High-Dynamic-Range imaging is a pipeline used to generate a single image while preserving detail in areas of extreme lights and shadows. Recent stack-based pipelines generally operate by capturing multiple images at varying exposures as shown in Figure 2, and then later merging each capture into a single HDR image [3]. While the results are compelling, methods such as HDR bracketing can be taxing on physical hardware and possibly produce occasional artifacts such as motion blur or alignment inconsistencies.

In recent years, Google developed HDR+, a feature in the Google Camera app for the Nexus 5 and Nexus 6. They found that, instead of bracketing photos, they can take a burst of shots with short exposure times and merge them algorithmically by replacing each pixel with the average color at that position across all the shots [6]. In this paper, I will address the HDR+ setting of burst photography.

3. CAPTURE AND PROCESS OVERVIEW

Figure 3 represents the burst photography process as a real-time pipeline. Once the camera app is opened, the camera begins recording and streaming raw frames to the viewfinder as shown in Figure 3 (top row). When the shutter is pressed, a burst of frames is captured at a constant exposure, formed into a stack and stored temporarily in

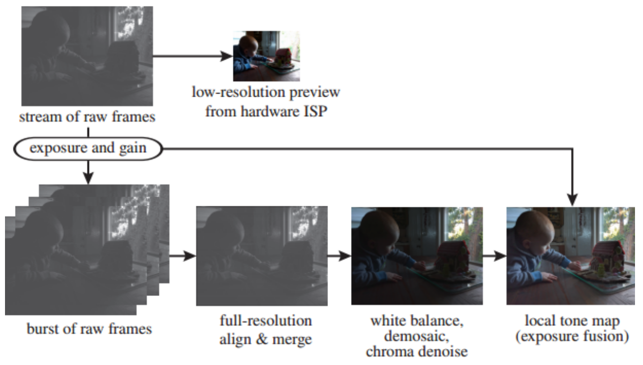
stream of raw frames
low-resolution preview from hardware ISP
exposure and gain
burst of raw frames
full-resolution align & merge
white balance, demosaic, chroma denoise
local tone map (exposure fusion)

Figure 3: An overview of Hasinoff, et al's., burst photography processing pipelines on the Nexus line of phones [4].

main memory. At this point the pipeline as shown in Figure 3 (bottom row) is activated and the stack of frames are aligned and merged together (Sections 5 and 6) into a single image. The pipeline then applies white balance adjustments and tone mapping to produce a single full-resolution photograph [4].

One of the key enabling technologies of burst photography is the Camera2 API [1]. Camera2 utilizes a request-based architecture that allows the camera to analyze a scene for exposure setting, capture images in *RAW* file format, and pre-record frames for burst capture. At the time of writing, Hasinoff, et al. [4] describes the only documented pipeline to capture in RAW natively. There are several reasons as to why burst photography captures in RAW rather than traditional JPEG. First, RAW files store roughly 3x more data than JPEG files, thus allowing for more detail and information about the image to be preserved. Second, RAW files use a combination of red, blue, and green filter patterns located on the camera's image sensor to capture high details of light and color. For any $2x2$ area of pixels, there will be two green, one blue, and one red. This format (know as RGGB) and is beneficial because our eyes distinguish luminance (brightness) with greater intensity within the green channel [2]. Third, RAW files contain the entire dynamic range spectrum for the photo. This means that the camera can capture at a set exposure and adjust highlights or shadows afterwards if need be.

The ability to adjust the dynamic range settings of an image after capture means that there is no need to bracket the exposure between frames as shown in Figure 2. Instead, burst photography captures every frame at a the same exposure. Removing the need to alter exposure is one of the main shortcuts in reducing the processing time.

4. EXPOSURE METERING

It's a difficult step in any photography pipeline to set a cameras exposure settings. Because exposure setting are scene dependent, most pipelines rely on an auto-exposure algorithm to make the proper adjustments. These algorithms usually operate by finding a balance between lights and shadows within the dynamic range [3]. While a traditional auto-exposure algorithm will produce compelling results, there are some scenarios that indicate room for improvement.

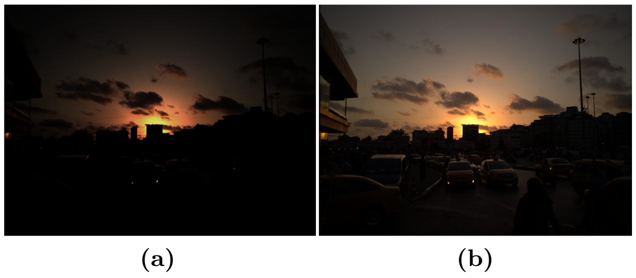

Figure 4: Above is an example of a sunset image taken from Google's example set along with the Hasinoff, et al. [4] The image on the left (a) is an illustration of the exposure setting from standard auto exposure algorithm. The image on the right (b) was taken using burst photography database to set the exposure.

As shown in Figure 4 (a), most traditional auto-exposure algorithms would select the bright light of the sunset and lower the exposure in order to balance out the dynamic-range. However, it's usually acceptable in a scene like this to leave the sun slightly over-exposed to preserve color throughout the sky and maintain detail within the foreground as shown in Figure 4 (b).

Another issue with setting the auto-exposure involves the edits that are applied to the image throughout the burst photography pipeline as mentioned in section 3. It is possible that what a user sees through the viewfinder could be a misrepresentation of the final image. To address this, Hasinoff, et al., [4] introduce the development of their custom auto-exposure algorithm that pre computes future image processing and is responsible for adjusting the exposure settings.

4.1 Exposure database

To account for auto-exposure failures, Hasinoff, et al., constructed a database of scenes for the camera to constantly compare what it sees to. This database consists of about 5,000 images (at the time of writing) that have been taken with the burst photography pipeline. The goal is to contain enough images so that most scenes consumers are likely to encounter will have a matching set of example images [4].

Using this database, the camera takes input frames from the viewfinder and searches for images with matching descriptors such as exposure, composition, and colors as shown in Figure 4 (b). This way the sun in Figure 4 can be slightly over exposed in order to bring out some detail in the streets. Once a matching set of candidate images is found, then the settings from that set is applied to the camera and the burst is ready for capture.

4.2 Burst size

While setting were being made, the camera was constantly recording through the viewfinder at 30 fps and storing frames in temporary memory. This is performed alongside exposure adjustments as a way to reduce the overall processing time. Once the shutter is pressed, the camera captures the current frame and saves the previous 2 – 8 frames. The number of frames used is dependent on the exposure settings that were used. Once the frames are captured, they are put into a stack where one frame is selected to be the reference frame of the burst [4]. To account for shutter lag, the reference frame is chosen only from the first 3 frames in the burst. With the reference frame selected the stack is ready to move on to alignment and merging.

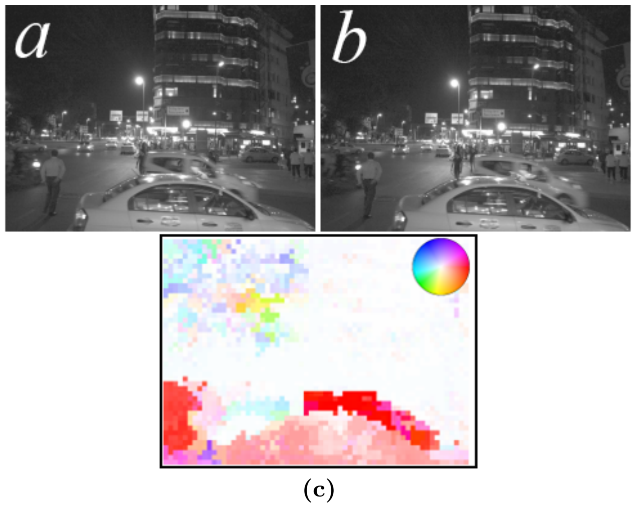

Figure 5: The upper images (a) and (b) are a pair of 3 Mpix grayscale frames. Where the lower image is a colorized representation of the motion between the frames. In the lower image (c), hue and saturation (depicted in the color wheel) indicate where movement in each frame is present and to what degree [4].

5. ALIGNMENT

The alignment phase of the burst photography prepares the stack for merging by aligning each of the non-reference frames to the chosen reference frame. This section will provide a brief overview of the *hierarchical alignment* process for the burst photography pipeline. The details of this topic are beyond the scope of this paper and can be found in the alignment sections of Hasinoff, et al. [4], and Liu, et al. [7]

Before alignment occurs, the frames in the stack are converted from color to grayscale in order to measure the average brightness of light at each pixel. The grayscaled frames are then downsized from 12 mpixs to 3 mpixs [4]. At this point each frame is divided up into multiple 16 x 16 pixel *tiles* as shown in Figure 5 (c).

Alignment is broken up into two processes. First, hierarchical alignment is performed on the downsized frames matching up non-reference frame tiles to reference frame tiles. This provides an initial guess as to how the frames will align to the reference frame. The frames are then upscaled back to their original 12 mpix resolution and fast subpixel L2 alignment is performed. Fast subpixel L2 alignment analyzes and aligns every pixel in each frame with the corresponding pixels in the reference frame. The goal is to align each of the frames 16 x 16 pixel tiles as close as possible to the reference frames tiles. Any misaligned tiles will be handled in the merging process.

6. MERGING FRAMES

A key goal when merging frames produce a high-dynamic range image with reduced levels of noise. The core function-

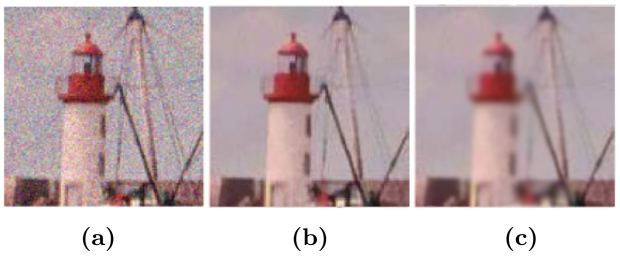

Figure 6: Image (a) is an example of a noisy image. Image (b) is the same image after denoising has been applied. Image (c) is an example of a over denoising.

ality of a merging algorithm is accomplished by combining non-reference frame tiles with reference frame tiles. This way, noisy pixels should get overlapped by non-noisy pixels. However, because noise is random, it's bad practice to only rely on the over-laying of tiles to denoise an image. Burst photography accounts for this by implementing a denoising algorithm known as Discrete Fourier Transformations or DFTs.

6.1 DFT

Before going into detail about the burst photography merging method, I am going to take a moment to introduce a denoising process called Discrete Fourier Transformations. Discrete Fourier transformations are a key component used in the burst photography merging method to prepare images for denoising. DFTs use the RAW file format to locate large quantities of noise data. Once selected, the noise values are reduced in order to blend noisy pixels into the image.

Figure 6 provides an example of a DFTs impact on a noisy image. One thing to note is that data used to represent noise is also what defines edges and outlines within the image. Therefore, the noise values can only be reduced to a certain point before edge detail also begins to deteriorate. Figure 6 (c) shows what can happen if the noise values were removed entirely from the image data. For more information on DFTs reference [8].

6.2 Burst Merging

In this section, I will cover the merging method developed for the Burst Photography pipeline. I will also be referring to their method as burst merging for duration of this section.

Burst merging operates by taking each tiles light values within the stack and constructing a new single image from their weighted averages. The process starts by selecting a tile in the reference frame and locating all corresponding tiles in the non-reference frames. Once all tiles have be selected, DFTs are performed on each tile to compute their noise values as output (ω). The next step is to take the weighted average coefficients of the noise values and apply them to the corresponding tile in the new image. To compare the noise values and construct the new image, burst merging implements Pairwise temporal filtering.

6.2.1 *Pairwise Temporal Filter*

The main goal of this merging process is to reduce noise in the final image. Burst merging handles this task by implementing a pairwise temporal filter with aims to attenuate noise within each non-reference tile. Let $T_z(\omega)$ be the noise

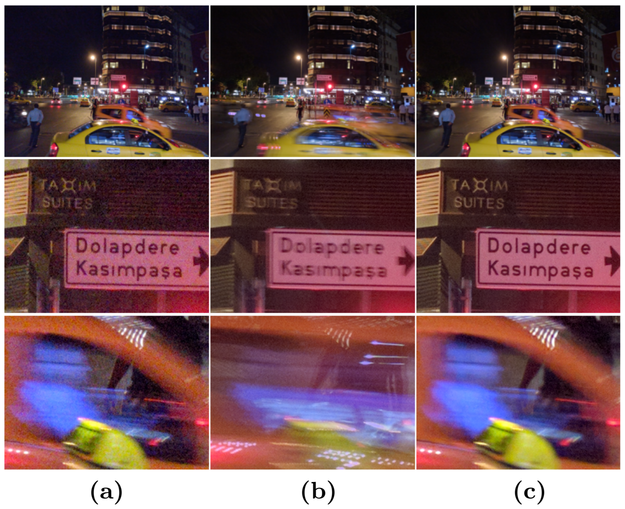

Figure 7: An example from Hasinoff, et al. [4] of merging 8 frames from a moving scene. From top to bottom, the rows consist of the input frames (top row), a crop where alignment succeeds (middle row), and a crop where alignment partially fails (bottom row). (a) A reference frame is chosen. (b) Averaging all 8 frames without alignment produces ghosts in regions exhibiting motion. (c) The result of the robust merge resemble the reference frames but with less noise.

value output for the z^{th} frame, $w = (\omega_x, \omega_y)$ be coordinate locations of noise in each tile, and $\tilde{T}_0(\omega)$ be the output value of the alternate tile that is applied to the new image [4]. Because noise is randomly generated, the position of noise will vary from frame to frame. Thus, once the output $T_z(\omega)$ is obtained, the alternate tile can be created and applied to the new image:

$$\tilde{T}_0(\omega) = \frac{1}{N} \sum_{z=0}^{N-1} T_z(\omega)$$

While this method is common practice for most frequency-based denoising methods, it does not account for alignment failure as shown in Figure 7(b). To add robustness, burst merging uses an expression similar to the first equation with the addition of a filter that controls the contribution of the alternate tiles [4]:

$$\tilde{T}_0(\omega) = \frac{1}{N} \sum_{z=0}^{N-1} T_z(\omega) + A_z(\omega)[T_c(\omega) - T_z(\omega)]$$

$A_z(\omega)$ controls the degree of usage between the alternate and reference tile in the new image. The body of this sum can be rewritten as $(1 - A_z) \cdot T_z + A_z \cdot T_0$ to emphasize that A_z controls a linear interpolation between T_z and T_0 [4]. This added robustness will ignore any tiles identified to contain alignment failures so that those failures will be avoided and not applied to the new image. This can be seen in Figure 7 where the bottom row shows that, if alignment failure is present, the reference tile will be used in the final result.

Since A_z controls how involved alternative frames are in creating the final image, it is important to determine which alternative tile coefficients are lower in quality than the reference tile. The value T_z should only contribute to the final image when its difference from T_0 can be ascribed to as noise. The contribution of T_0 should also be reduced when it differs from T_0 due to poor alignment [4]. The following equation defines A_z as a shrinkage operator and is a variant of the classic Wiener filter:

$$A_z(\omega) = \frac{|D_z(\omega)|^2}{|D_z(\omega)|^2 + c}$$

where $D_z(\omega) = T_0(\omega) - T_z(\omega)$, and c is a constant (set to 8 for Figure 7) that tunes the impact of A_z. Because the degree of contribution varies from tile to tile, the image is allowed to degrade in a graceful manner when alignment failure is present as shown in Figure 7 (c) row 3.

6.2.2 *Overlap Adjustment and Post Processing*

Up to this point, I have covered how burst merging operates by creating a single image from a burst of frames. However, before the image can be sent to finishing, the image needs to be corrected for any border overlap. I will provide a brief overview of both processes. Further detail can be found in Hasinoff, et al. [4] Tiles that have overlapped from the aliment process are corrected by applying a *windowed function* that acts a as a crop for the final image [4].

The image finishing process consists of 13 different operations that are applied with the goal of making the image match what was initially seen by the user. To get a basic idea of the resulting image after finishing, refer to Figure 4(b). More information can be found in Hasinoff, et al. [4]

7. FAST DENOISING

In Hasinoff, et al.'s, study [4] a comparison was performed between burst merging and other merging methods, one of them being Liu, et al.'s [7] pixel fusion method. In this section, I will compare images generated via the burst merging and pixel fusion pipelines. However, first I will provide an overview of how the pixel fusion pipeline works.

Fast denoising was developed by Lui, et al., under Microsoft Research Technologies in 2014 [7]. Similar to burst merging, fast denoising produces a single image from a burst of noisy images. Fast denoising handles scene motion by identifying consistent pixels throughout each of the frames. In the pipeline, the resulting images are aggregated through the use of a processes called Temporal fusion.

Temporal fusion begins by assuming that x_t are the color values of consistent pixels from the t^{th} frame of the input set. Where *consistent pixels* refers to pixels that are considered to be correctly aligned across all frames. Let $\tilde{x}$ be the resulting value of pixel fusion:

$$\tilde{x} = u + \frac{\sigma_c^2}{\sigma_c^2 + \sigma^2}(x_t - u)$$

where u is the mean of pixel values $\{x_t\}$, and the variance of *true pixels* (non-noisy pixels) is approximated by max(0, $\sigma_t^2 - \sigma^2$) [7]. σ_t is the standard deviation of $\{x_t\}$ and σ is the standard deviation of noise across all frames. If the variance of noise is greater than the variance of color, then the approximation of true pixels is set to 0 for the resulting image. Temporal fusion runs at every for every group of

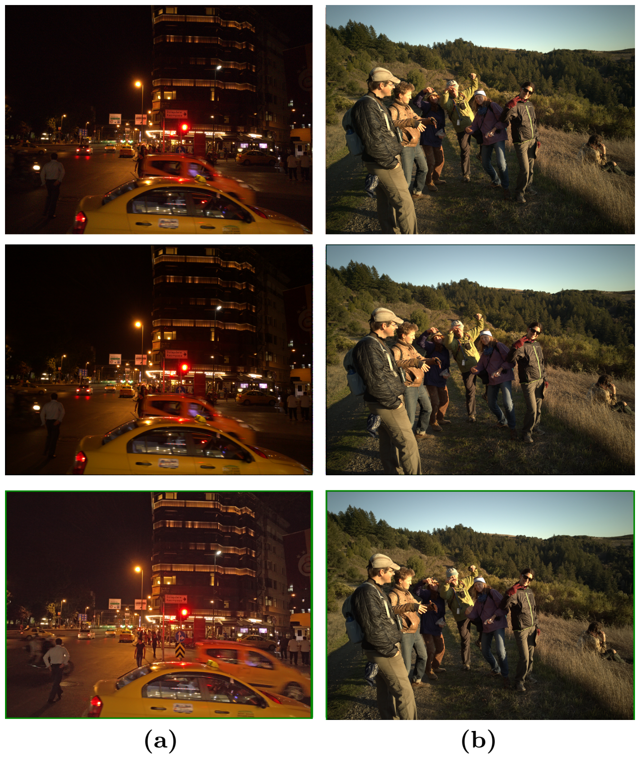

Figure 8: A comparison of two low-light photos produced by Hasinoff, et al.'s [4] burst merging (middle row) and Liu, et al.'s [7] pixel fusion (bottom row). Also, note that the green outlines indicate where cropping occurred within the pipeline. Taken from [5].

consistent pixels which is followed up by multi-scale fusion to aggregate the results into a single image. More information about multi-scale fusion and the rest of the fast denoising pipeline can be found in Lui, et al. [7]

8. COMPARISON AND RESULTS

For comparison test between the pipelines, 30 raw bursts sets were used from Google reference image database (Section 4.1). The photos chosen for the bursts were picked biased on of their coverage of motion levels and brightness. As shown in Figure 8 (top row), the the two images displayed illustrate performance on low-light scenes. Before the test, all images where converted to JPEG because the burst fusion pipeline does not operate with RAW files. This also ensured that the experiment only focused on the performance of aligning and merging algorithms.

Looking at the examples in Figure 8, both pipelines produce a denoised image with an appealing dynamic range. These examples also show that the main differences between burst merging (middle row) and burst fusion (bottom row), is in the alignment. Burst merging produced an image that is nearly identical to the input due to the robustness of the pairwise temporal filter. Any motion that was misaligned was not used in the final image. In comparison, burst fusion produced a result that is slightly different from the input image. In Figure 8 (bottom row), image (a) shows a taxi behind the car at a completely different position compared to the input frame, and in image (b) some of the hiker's arms are repositioned in the final image. This can be attributed to the alternate forms of handling alignment for the final photo. More information can be found in Lui, et al. [7]

9. CONCLUSIONS

As demonstrated in this paper, solutions for capturing detailed photos in low-light with high dynamic range have existed for a while but fail to operate in a timely manner on mobile devices [3]. With these hardware constraints in mind, Hasinoff, et al., describe a system for capturing a burst of underexposed frames, aligning and merging them together to produce a high-resolution photograph on mobile devices. The system was deployed on several mass-produced cell phones, marketed as HDR+ in the Nexus 6, 5X, 6P, and Pixel phone [4]. While other merging methods such as Liu, et al.'s [7] pixel fusion produce similar results, from the burst processing tests we can see that burst merging produces better results especially in low-light scenes.

There are still limitations to future development of burst photography. Hasinoff, et al., point out [4] that a major hurdle is the computing time of the images and the differences between what's shown in the viewfinder and how the final image appears. In extreme circumstances, users might abandon taking an image altogether due to the lack in quality displayed on the viewfinder, or miss out on capturing a moment in rapid burst due to the HDR+ small processing delay. These problems are possible to solve at the time of writing, but might take time in order to bring the needed technology down to a consumer level.

Acknowledgments

Thanks to Nic McPhee for help, time, and advice, to Elena Machasova for her corrections, advice, and improvements, to Peter Dolan for his advice and insight, and to Maria Medlyn for her insight and corrections.

10. REFERENCES

[1] Android. `android.hardware.camera2` documentation.

[2] Computerphile. Youtube. `https://www.youtube.com/watch?v=LWxu4rkZBLw`, Feb 2015.

[3] O. Gallo and P. Sen. *Stack-Based Algorithms for HDR Capture and Reconstruction*, chapter 3, pages 85–119. Academic Press, 2016.

[4] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Trans. Graph.*, 35(6):192:1–192:12, Nov. 2016.

[5] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras, supplemental material. *ACM Trans. Graph.*, 35(6):192:1–192:12, Nov. 2016.

[6] M. Levoy. HDR : Low light and high dynamic range photography in the google camera app, Oct 2014.

[7] Z. Liu, L. Yuan, X. Tang, M. Uyttendaele, and J. Sun. Fast burst images denoising. *ACM Trans. Graph.*, 33(6):232:1–232:9, Nov. 2014.

[8] F. Weinhaus. ImageMagick v6 examples – Fourier transforms, July 2000.

[9] Wikipedia. Image noise — Wikipedia, The Free Encyclopedia, 2016. [Online; accessed 6-April-2017].

[10] Wikipedia. Tone mapping — wikipedia, the free encyclopedia, 2017. [Online; accessed 4-April-2017].