Leveraging Pixel Value Certainty in Pixel-Shift and Other Multi-Shot Super-Resolution Processing

Henry Gordon Dietz; Department of Electrical and Computer Engineering, University of Kentucky; Lexington, Kentucky

Abstract

Traditional super-resolution processing computes sub-pixel alignment over a sequence of image captures to allow sampling at a finer spatial resolution. Alternatively, the mechanism intended to implement in-body image stabilization (IBIS) can be used to shift the sensor in a stationary camera by precise fractions of a pixel between exposures. The implicitly perfect alignment of pixel-shift images reduces post-processing to interleaving of raw data, but motion of camera or scene elements produces disturbing artifacts. Determining misalignments on raw images from cameras using color filter array (CFA) sensors is potentially problematic, so the synthesized super-resolution image is instead typically built from already-interpolated image data, with a reduction in tonal quality.

The current work instead directly models the certainty, or confidence, with which pixel values are known. Sub-pixel alignment may be computed on either raw or interpolated image data. Still, only the underlying raw samples have precisely known values, so only they are used to compute the super-resolution image. However, primarily due to motion, even raw pixel values can have variable value certainty. Thus, a confidence metric is calculated for each raw pixel value and used as a weighting factor in computing the best estimate for the value of each super-resolution pixel.

Introduction

Television shows and movies often assume that an image from any camera can be "zoom and enhance" processed to dramatically improve image resolution. The improvement is imagined to happen in both spatial and tonal dimensions: not only are incredibly fine details made visible, but the noise level of the image is reduced so that finer tonal gradations can be perceived. Processing to increase the spatial and/or tonal resolution of a single captured image is generically referred to as single-image super-resolution processing. In recent years, a variety of artificial intelligence (AI) methods have emerged that appear to enhance spatial and tonal resolution significantly [1] [2] [3], but they are fundamentally hallucinating credible details that might or might not resemble the actual scene detail. The only way to create a known-accurate image of a scene with higher spatial or tonal resolution than the sensor of the camera being used is to combine images from multiple captures[4][5], which can be accomplished in various ways as summarized in Table 1.

Table 1: Multi-Shot Methods for Improving Resolution

Method	What Moves?	Capture Overlap	Tonal or Spatial?
Stitched Panorama	Camera	Small	Spatial
Scanning	Sensor	Small	Spatial
Averaging	Nothing	Large	Tonal
Image Stacking	Camera	Large	Both
Pixel Shift	Sensor	Large	Both

Using film and photochemical processing, the emphasis is on improving spatial resolution and multiple captures are generally shot as a panorama sequence in which the camera's view is changed with just enough overlap between captured images to facilitate aligning the seams between prints. *Panorama stitching* also is commonly used with electronic sensors[6]. Unlike physical splicing of printed photographs, digital processing can computationally warp images so that distortions caused by the lens and re-aiming of the camera need not result in severely artifacted seams. As an alternative to panning the camera, *scanning*[7][8] the coverage circle of the taking lens can be implemented by moving the sensor, eliminating the potential for seam misalignments due to lens distortion.

Given the ability to precisely align captures with sub-pixel accuracy, keeping overlap between digital image captures very high offers the significant benefit of potentially improving both spatial and tonal resolution. Many cameras incorporate multi-shot averaging modes for reducing high-ISO noise or as an anti-shake mechanism. *Image stacking*, widely used in astrophotography, computes precise alignments by analyzing the images captured and aligning features. In contrast, *pixel shift* uses computer motion control mechanisms to implement tiny, very precise, movements of the sensor, typically in units of 1/2 or 1 pixel. The deterministic positioning should allow fewer captures to yield higher-quality data because the scene is more regularly sampled, and it should not be necessary to determine alignment computationally.

While all the above techniques are somewhat effective, problems like motion artifacting limit the quality of the combined images. The focus of this paper is the use of *pixel value certainty*, or *confidence* in processing pixel-shift captures so that artifacted pixel value samples are given less weight in determining the final image pixel values.

Pixel Shift

Pixel shifting, sometimes called *microscanning*, has been known for at least several decades as an evolutionary step beyond line scanners. In the 1980s, the Jenoptic ProgRes (programmable resolution) 3012 commercial microscope camera[9] moved a rel-

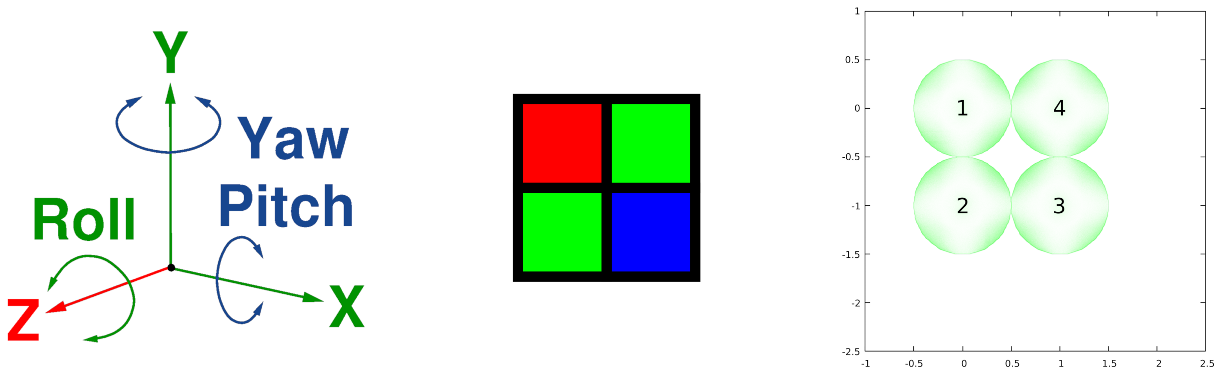

Figure 1. *IBIS supports X, Y, and Roll movements; with a 2×2 RGGB CFA, a 4-shot sequence using X, Y pixel offsets can provide full color at each pixel*

atively small and low resolution CCD sensor behind the lens using piezo-electric actuators so that a series of exposures could produce images as large as 4608×3480 uninterpolated pixels.

The thing that has made pixel shift suddenly much more appealing is the common implementation of IBIS (in-body image stabilization) in interchangeable-lens mirrorless cameras. As sensors reached ever higher pixel counts, tiny amounts of camera shake became the fundamental limit on the spatial resolution that could be delivered. Because moving the sensor to compensate for this shake is done within the body, IBIS avoids adding complexity to the lenses. OIS (optical image stabilization) in lenses is also unable to compensate for as many types of motion. The leftmost diagram in Figure 1 shows the six dimensions of motion which may be present in camera shake. OIS is only able to easily correct for Pitch and Yaw movements. In contrast, 3-axis IBIS, which directly implements motion in the X, Y, and Roll dimensions, can approximately correct for small-angle Pitch and Yaw movements by combining them in the computation of Y and X offsets. Except perhaps at macro focus distances, motion in the Z axis has negligible impact on the image and is thus ignored.

The IBIS motion control is normally computed in response to data from an IMU (inertial measurement unit) sampled many times during an exposure. To track camera shake, typical sample rates are in the kilohertz range, and the actuators that move the sensor can only be effective if they are capable of responding to shake on that timescale with positioning accuracy that is at least comparable to the dimensions of a single pixel. Thus, given the assumption that the subject and camera are in fixed relative positions, it is actually much simpler to implement precise pixel shifts between captures than to implement shake-compensation movements. In pixel shift sequences, the Roll dimension position is fixed and the X, Y offsets are changed between exposures.

Pixel Shift Modes

There are two ways in which pixel shifting potentially improves both spatial and tonal resolution:

Full pixel steps: Nearly all cameras use a CFA (color filter array), such as the 2×2 RGGB Bayer pattern shown in the center of Figure 1. A single shot does not sample all color channels at each pixel site. By moving the sensor in pixel-sized steps in the X and Y directions, it is possible to position each color channel in every pixel location over a series of captures. The result is primarily improved color accuracy and reduction of artifacts like moire – improvements to tonal resolution. In theory, spatial resolution of luminance scene details might be unaffected by full-pixel shifts, but resolution of color details could be doubled in both the X and Y dimensions. Typical capture sequences for this use just four exposures, one for each of the color placements within the CFA, making exposures offset in a pattern like that shown in the right side of Figure 1.

Fractional pixel steps: Moving the sensor in smaller increments, it becomes possible to increase resolution by sampling between pixels. Most commonly, 1/2-pixel steps are used, thus doubling linear pixel density and quadrupling the total pixel count delivered. A typical capture sequence will sample each 1/2-pixel position with all four channels of a 2×2 CFA, thus requiring a rather long sequence of 16 exposures. Beyond the awkwardness of such a long sequence, a high effective pixel fill factor implies that the fractionally-moved pixels overlap the areas sampled in neighboring positions, reducing contrast. Similarly, many sensors incorporate filter stacks that include AA (anti-alias) filters that spread focused light over multiple pixels, and this also can reduce contrast. Of course, a lens that does not project a sufficiently sharp image will also limit the spatial resolution that can be achieved by pixel shifting.

Generally, pixel shift modes require that the camera be in a fixed position relative to the subject for the entire pixel-shift capture sequence. There have been attempts to implement pixel shifts while correcting for camera shake: i.e., handheld pixel shift. However, that would require precisely correcting for shake during the entire sequence of captures rather than just within a single capture. Combining that longer correction period with somewhat higher positioning accuracy requirements, such as positioning in 1/2-pixel steps, is challenging. For example, Pentax has implemented handheld pixel shift[10], but only for a 4-shot sequence.

Standard Pixel Shift Processing

Pixel shift is based on the idea of precisely-controlled motion of the sensor. Given that precise motion, the software for combining pixel-shift captures is logically very straightforward:

Full pixel steps: The merging rule is simply that, instead of the usual demosaicing interpolation to spread color information, each pixel uses only the data for the color channel that it sampled

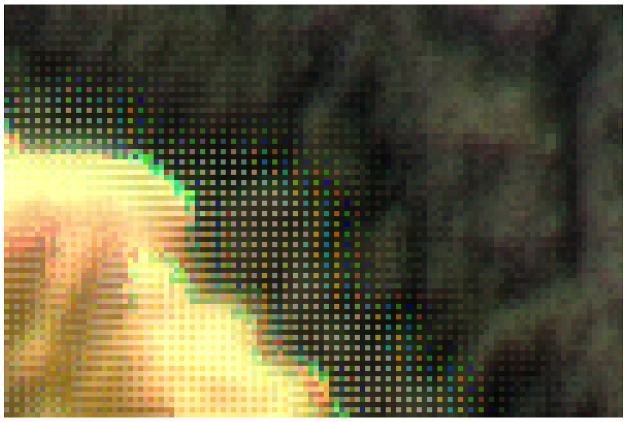
Figure 2. *Fragment of* `PixelShift2DNG` *output showing motion artifacts*

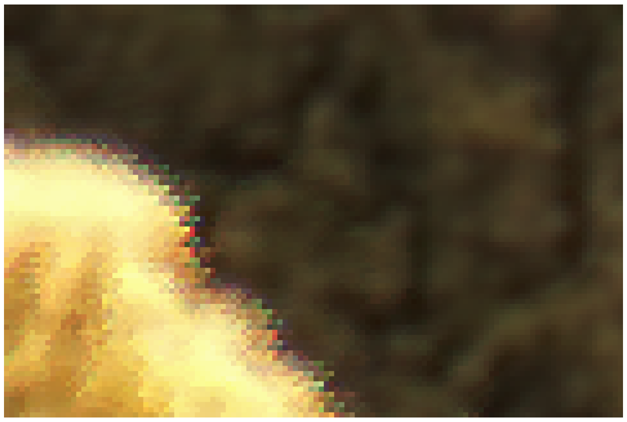
Figure 3. *Fragment of* `parsek` *output corresponding to Figure 2*

in each capture. The most common CFAs have one Red, one Blue, and two Green pixels in a repeating 2×2 pattern (a RGGB pattern was shown in the center of Figure 1). Clearly, to get a Red sample in each pixel location will require four shift captures, and the same holds for Blue. However, such a shift pattern will cause each final pixel to contain two Green samples. If we assume that the two Green filters have identical spectral properties, simply averaging the two Green samples at each pixel location will produce a lower-noise Green value. If the CFA pattern is more complex, for example using a four-color pattern as in the Sony F828[11], the process would be the same except for producing additional color channel data for each pixel location. In any case, it is important to recognize that the result of this process is not a final image rendering, but simply a "raw" image with more complete data at each pixel site: color management and gamma encoding would still need to be done.

Fractional pixel steps: The fractional pixel steps are generally taken to be small integer divisors of the size of a pixel, such as 1/2 pixel steps producing an image with four times the sensor's pixel count. Typical pixel shift patterns will sample each CFA color in each subpixel position, and the simplest processing would only use the data for the color channel centered on each subpixel. The main problem with this is that, as discussed earlier, each sample sees light hitting a larger area than just the subpixel upon which it is centered. The result is that the "raw" image produced will need some type of sharpening to counter this degrading of local contrast. Empirically, relatively aggressive sharpening methods applied with a small radius work well because fine detail is present with low contrast and low noise.

Although pixel shift support is generally in software provided by the camera manufacturer, and some cameras also offer merging of pixel-shift captures in-camera, the only freely available implementation handling raws from a variety of cameras is `PixelShift2DNG`[12]. That software, written by the developers of `LibRaw`[13], uses the metadata recorded by the camera with each capture to recognize pixel-shift images, and automatically combines the captures as described above. The resulting file is a DNG raw that retains the original raw file metadata for color correction, etc, and the quality of the images it produces often is remarkable.

Confidence-Based Pixel Shift Processing

The obvious problem with the processing implemented by `PixelShift2DNG` is that any subject motion causes very distinctive and disturbing artifacts. For example, Figure 2 shows a 120×80 pixel fragment of the 19128×12744 image produced by combining a 16-shot pixel shift sequence shot using a Sony α7RV on a tripod. Care was taken to minimize camera movement, including use of self timer and electronic shutter, but that did not prevent some flowers in the foreground of the scene from swaying with the wind. The shutter speed was fast enough (1/336s) that there is no such artifact in any of the individual captures, so it should be possible to construct a more credible super-resolution image.

When confronted with such a problem, the common reaction is now to apply machine learning (ML) to filter the image. Such methods can be very effective, but ML is inherently biased by the set of training images used. To ensure that artifacts are not systematically biased, the approach in this paper is instead focused on directly using information from the captures themselves. The key concept is to be certain that the value selected for each final pixel is an accurate representation of the scene content, and this **certainty is sought by tracking the confidence with which each potential contribution to a pixel value is known**. Confidence in pixel values is a function of:

1. Variations in the scene content at the target pixel location over the time interval spanned by the individual exposures, i.e., the issue evident in Figure 2. This similarity can be measured in at least two fundamentally different ways, as either the difference from a particular capture treated as a reference or as the difference from the statistically average scene content. Figure 3 shows how effectively the method proposed in the current work resolves this problem.
2. The quality of the alignment of each sensel sample with the target pixel's ideal sample area. This metric also has multiple dimensions in that poor alignment can arise from either a mismatch in shape or size of the active sensor area (i.e., fill factor) or a mechanical positioning error. In conducting the experimental evaluation of the new approach, it was discovered that positioning error is surprisingly significant

under ordinary capture conditions. Where micron-scale positioning is concerned, the phrase "trust, but verify" seems particularly appropriate.

3. The probabilistic error bounds on each digitized sensel value. This summarizes the combined effect of photon shot noise, sensel characteristics, and basic analog and digital processing of sensel values.

These three aspects of pixel value quality are each treated somewhat differently, yet all are internally represented as probability-like confidence values between 0 and 1 for each individual sample. Confidences across these dimensions either can be combined numerically to produce a unified confidence metric or can be used as thresholds to invalidate low-quality potential contributions to a final pixel value.

Parsek

The proposed new model for combining pixel shift captures was prototyped by creating an open source C++ program called `parsek` (probabilistic alignment raw stitcher experiment from Kentucky)[14]. The name is a nod to the concept of a parsec, which is a unit of distance corresponding to a parallax of one second – i.e., the distance at which 1 AU subtends an angle of one arc-second, or approximately 3.26 light years. Both are about making big images by looking at small angle offsets.

At this writing, `parsek` has gone through over 40 revisions. The current version is approximately 1500 lines of C++ source code that takes advantage of the `OpenCV`[15] library and uses either `dcraw`[16] or `LibRaw unprocessed_raw`[13] as a helper program to convert various proprietary raw formats into unprocessed linear sensel data in the `PGM` format[17]. The `PGM` file format with 16 bits per sensel value is used, but most cameras have fewer active bits per raw sample, so `parsek` recognizes how many bits are active, bpp, and promotes the values to 16-bit integers using the formula $((v << (16-bpp))|(v >> (bpp-(16-bpp)))$. All values within `parsek` are represented either as 16-bit integers or 32-bit floating point values.

Raw Sensel Data?

In most computational merging of multiple images, such as image stitching, the image data is usually cooked before merging. For example, `Adobe Photoshop` can be used to automatically image align layers for merging, but reading a raw image into the software demosaics it, filling-in any missing pixel color channels by interpolation. In contrast, traditional pixel shift processing is performed entirely on raw sensel data.

While `parsek` can operate on raw data in a wide range of formats using the raw decoding process described above, it also can operate on cooked image data, including JPEGs. This was done partially to make the software more flexible, but also to allow direct comparisons between use of raw data, use of cooked data, and even use of approximate raw data synthesized from cooked images. Input full color images with either 24 or 48 bits per pixel can be used in their cooked form with all color channels for each pixel contributing to the final image. Most image formats encoding 8 bits per color channel assume a gamma between 1.8 and 2.2, so correction with a default gamma of 2.0 is applied to create linear 16-bit values. The `parsek` software allows explicit specification of a 2×2 CFA pattern; for example, the pattern seen in Figure 1 would be described as `RGGB`. If a CFA pattern is specified, even if the input image was a cooked format, only the pixel color channels that are not in interpolated positions are used – effectively reverse-engineering approximate raw data.

Although the differences were typically small, using real or reverse-engineered raw data tended to render details a little sharper, but with slightly more noise, than using using interpolated full-color data.

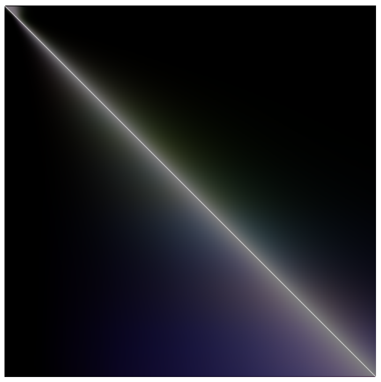

Figure 4. *A pixel value error model created by* `parsek`

Pixel Value Error Model

The third attribute listed above as a key contributor to pixel confidence is the probabilistic error bounds on sensel values: the contribution of noise. In earlier work[18][19], we have shown that a useful, yet concise, model of probabilistic errors in sensel values sampled can be constructed as a 2D histogram for each color channel. If two sensel values, v_1 and v_2, are found to be likely recording the same scene content, then the histogram box with the corners (v_1, v_2) and (v_2, v_1) is incremented. After all such pairs have been processed, the counts are normalized to values between 0 and 1 by scaling each row v's entries by the count in entry (v, v). The result is essentially a lookup table in which the value recorded at (v_1, v_2) can be treated as the confidence that sensed value v_1 is really representing the ideal value v_2. Such an error model can be visualized as a square color image like that shown in Figure 4.

Typical methods used to determine when two different samples might really represent the same true pixel value included using the exact same pixel location sampled from consecutive video frames or searching areas within a single image looking for the closest matches. In `parsek`, the analysis is done by performing feature-based alignment of the images and marking all the boxes determined by all possible pairings of the best-aligned pixel values. For example, if there are three images and a particular aligned pixel location in the first image has the value v_1, the second has v_2, and the third has v_3, then three histogram boxes would be incremented: (v_1, v_2) to (v_2, v_1), (v_2, v_3) to (v_3, v_2), and (v_1, v_3) to (v_3, v_1). The histogram is then normalized to confidence values between 0 and 1 and used as a lookup table for determining how likely it is that two different samples are testing the same scene content and should thus be averaged to obtain a more precise estimate of the true pixel value.

Alignment

One of the key points of pixel shift is that the precision movement of the sensor makes it unnecessary to perform alignment computations. However, we tested that assumption and the results were surprising in more than one way.

The first surprise was how simple it was to compute sufficiently accurate sub-pixel alignments using `OpenCV`[15]. The `findTransformECC()` function is intended to find the geometric transformation (warp) between two single-channel images by maximizing the Enhanced Correlation Coefficient[20]. Although uninterpolated raw sensor data is represented as a single-channel image, the data interleaves color channels, and it was not clear that this algorithm would be effective. Test cases synthesized by simulating precise unit pixel raw shifts yielded subpixel accuracies much finer than the 1/2-pixel shifts which are the finest movements intended in the pixel shift cameras available to us. A known weakness of this alignment algorithm is that it requires an initial estimate of the alignment transformation in order to be effective with large displacements and/or rotations, and `parsek` incorporates a filter to ignore input images requiring stronger transformations, but the identity transform proved to be a sufficient estimate for the scale of movements seen in pixel shifting.

The second surprise was that the offsets in “real world” pixel shift images were not the precise multiples of 1/2-pixel shifts expected. For 16-shot pixel-shift sequences captured using a Sony α7RV, Figure 5 shows the theoretical offsets, offsets measured under near-ideal conditions indoors, and offsets measured from capturing the the real-world scene from which Figures 2 and 3 were cropped. The shift measurements used the same camera and tripod, with the camera set to electronic shutter and triggered by self-timer to minimize vibration in both cases, but the offsets differed dramatically. Ideal conditions led to offsets close to theory, but the real-world offsets are shockingly different. Is it reasonable to expect a camera on a tripod to limit accidental movement of the image on the sensor to far less than a $3.7\mu m$ pixel over the course of a 16-shot capture sequence that took approximately one minute? In the particular case described here, we photographed Yosemite’s El Capitan from a popular viewing spot with the tripod resting on solid concrete, but surrounded by a crowd of people with cars driving by and enough breeze to wildly move flowers in the foreground (as seen in Figure 2).

The fact that pixel-shift offsets are not precise under normal shooting conditions implies that higher resolution could be achieved by using computed alignments than by using the expected shift values. In practice, there was very little if any additional detail visible in $2\times$ super-resolution, but the tonal quality was notably better using measured offsets to compute confidences. Super-resolution processing to $3\times$, which is not possible using traditional pixel-shift processing that assumes 1/2-pixel offsets, revealed slightly more detail. In all cases, traditional pixel-shift processing resulted in slightly higher contrast, but that was likely due to a higher noise level. Confidence-weighted averaging mean that a typical pixel color channel value has contributions from 16 samples rather than just 1.

Perhaps more significantly, it was found that without taking exceptional measures, simply mounting a camera on a tripod and making a series of captures caused random movements of comparable magnitude to those implemented by pixel shift. This was true for a Sony NEX-5 (mechanical shutter mirrorless), Canon EOS 5D Mark IV (mechanical shutter DSLR), Panasonic Lumix DC-GX850 (electronic shutter mirrorless), etc. Whereas movement range over the Yosemite sequence was around $8\mu m$, manually firing the electronic shutter of a mirrorless body tended toward about twice as much movement and mechanical shutters caused $2\times-4\times$ that much movement. The tripod-mounted movements also had a distinct profile: there was almost no rotation and X movement tended to be significantly more than Y movement. Of course, it is also possible for parts of the camera itself to cause shifts; it is common in compact cameras that vibration induced by lens aperture or shutter movement can shift a lens within the mechanical tolerance limits of its focus and zoom mechanisms. Using `parsek` to process these random-movement sequences produced results strikingly similar to those using pixel shift.

Pushing this idea to the extreme, the amount of motion across shots when hand-holding a camera with IBIS also was measured. Within each shot, IBIS is effective in keeping overall motion negligible. However, IBIS systems may reset between shots to avoid hitting motion limits and the user’s grip may also change slightly. Using a Sony α7II handheld with IBIS and mechanical shutter, worst-case movement was limited to about $0.2mm$ – an order of magnitude more than using a tripod. However, the IBIS correction of Roll was very effective. The result was that the `findTransformECC()` routine was able to find subpixel-accurate alignments and handheld super-resolution processing using `parsek` still provided notable improvements even when alignment of some shots failed causing their data to be ignored.

Subject Motion

The first motivation listed above for finding a better pixel-shift processing technique was to better deal with subject motion. Although `PixelShift2DNG` apparently does not try to reduce such artifacting, there are at least three methods that have been used to avoid the artifacting seen in Figure 2.

One approach is to use a slow shutter speed for the captures. If the individual captures are blurry due to subject motion, merging them tends to produce a natural-looking blur rather than the type of artifacts seen in Figure 2. The main problem with this is that we now know the natural movement of a tripod-mounted camera in poorly-controlled environments easily could cause pixel-level blur for portions of the scene that aren’t moving. This is also a method that can only be applied at capture time, not a decision made during post-processing.

A second alternative is to pick a reference capture and substitute interpolated values from it when the intended sample is significantly different. A minor variation on this scheme measures sharpness of all captures and prefers data from the image that is sharpest in image regions where the difference is significant. A very simple early version of this is Nikon’s BSS (best shot selector), which captures multiple shots and then in-camera

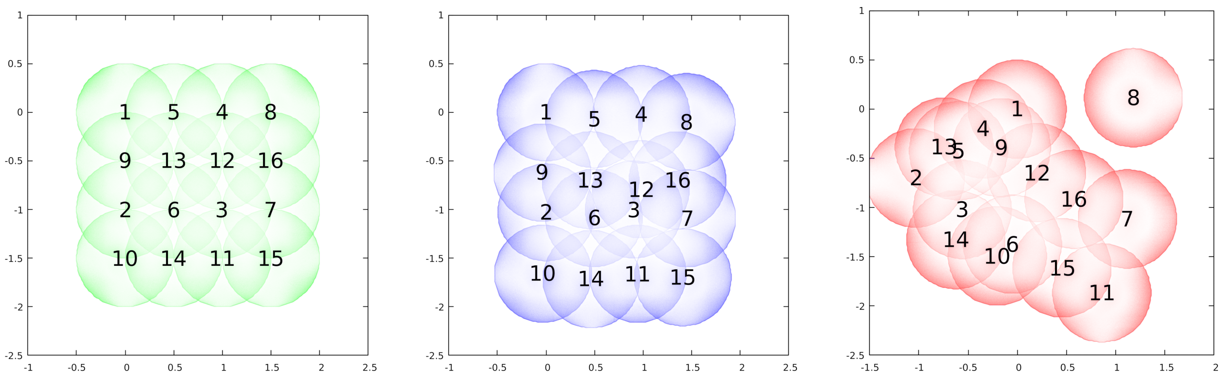

Figure 5. *Plot of Sony α7RV 16-shot pixel shift offsets: theoretical, measured ideal indoor conditions, measured real-world outdoor scene*

discards all but the sharpest. Of course, this solution implies that where there is motion, there will be no increase in spatial or tonal resolution.

The third option would be using the average value from nearby pixels across all captures to define the expected value of the pixel and discard pixel values distant from that. This is essentially the idea used in the Drizzle algorithm[21], and the principle is extremely effective for astronomical imaging because most "motion" is essentially noise from short-lived phenomena that are not the intended target. Unfortunately, motion artifacts caused by natural scene element movement, such as plants swaying in the breeze, often survive this averaging process.

The current work investigated a wide range of different weighting methods for reducing motion artifacting. The key to most methods is being able to distinguish between samples that are or are not seeing the same scene element. In `parsek` this is not done by ML recognition of scene content, but by having the histogram-based model of noise described above. Not only does table lookup provide a confidence that two values are approximations to the same ideal value, but as a side-effect it also filters-out excessively noisy or "stuck" pixels. In fact, this type of noise model is highly effective for noise reduction in images[19].

Using this similarity confidence metric, `parsek` can filter potential pixel contributions by either similarity to pixels in a given reference image or by similarity to the average value – the second and third options discussed above, but with important twists.

The confidence metric can implement either a hard or soft cutoff. Command line options for `parsek` allow setting a confidence threshold for discarding a value, but confidences also can be directly used for merging not only disparate data values, but also their component confidences. There is even an adjustable level of softness implemented by specifying an exponent for the histogram-based confidences; sharpen the cutoff by specifying an exponent >1, smooth it with an exponent <1.

The "average" used is not the mean, but a metric closer to the median. The interesting concept here is how to deal with having three color channels. Rather than computing median based on value, we compute it as the sample that has the highest total confidence of being paired with all the other samples potentially relevant to this pixel. In case of a tie, the contending values are averaged to define the reference value.

The best choice for filtering of motion artifacts depended somewhat on the scene content. For example, natural landscapes with moving foliage were handled best by soft filtering based on similarity to a reference image while noisy relatively static scenes are handled better using a hard filter on the average. In cases like raw landscape captures that have uncorrected bad pixels (e.g., "Christmas Tree light" artifacts from long exposures without dark frame subtractions), a combination filter is most effective.

Conclusions

The current work has explored an alternative model for processing of pixel-shift and other types of captures for multi-shot super-resolution rendering as full-color raw images. Rather than assuming that the intended pixel-shift motions are perfectly implemented and using ML techniques to filter-out larger-scale motion artifacts, the proposed approach simply pays closer attention to the statistical properties of the raw captures – or of reverse-engineered approximations to raws generated from cooked input images. Expected shifts are used as no more than initial estimates for directly measuring the shifts from the captures themselves. All potential sample contributions to the the computation of each final raw pixel value are weighted by confidence values taking noise, alignment, and consistency across component captures into account. These confidence values also allow considerable tuning by adjusting a small number of easily understood model variables.

Working with pixel-shift captures as input, the typical result is a modest improvement in quality of the super-resolution raw result. Spatial resolution is slightly better than conventional pixel-shift processing would produce. Contrast is lower using the proposed approach, but noise is significantly reduced, so the resulting super-resolution raw can be aggressively sharpened.

Unless the capture environment is very carefully controlled, random movements, primarily in the X axis for tripod-mounted cameras, occur across captures with magnitude similar to or greater than pixel shifts. Although these movements make pixel shift offsets differ from the expected motion, by measuring the actual shifts in a sequence of shots with a tripod-mounted camera or even a handheld camera with IBIS, our `parsek` software was able to produce high-quality super-resolution images – even when pixel shift was not used. The C++ source code of `parsek` is freely available[14].

References

[1] D. Glasner, S. Bagon and M. Irani, "Super-resolution from a single image," 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 2009, pp. 349-356, doi: 10.1109/ICCV.2009.5459271

[2] W. Yang, X. Zhang, Y. Tian, W. Wang, J-H. Xue and Q. Liao, "Deep Learning for Single Image Super-Resolution: A Brief Review," in IEEE Transactions on Multimedia, vol. 21, no. 12, pp. 3106-3121, Dec. 2019, doi: 10.1109/TMM.2019.2919431

[3] Honggang Chen, Xiaohai He, Linbo Qing, Yuanyuan Wu, Chao Ren, Ray E. Sheriff, and Ce Zhu, "Real-world single image super-resolution: A brief review," Information Fusion 79 (2022): 124-145.

[4] Sung Cheol Park, Min Kyu Park and Moon Gi Kang, "Super-resolution image reconstruction: a technical overview," in IEEE Signal Processing Magazine, vol. 20, no. 3, pp. 21-36, May 2003, doi: 10.1109/MSP.2003.1203207

[5] S. Farsiu, M. D. Robinson, M. Elad and P. Milanfar, "Fast and robust multiframe super resolution," in IEEE Transactions on Image Processing, vol. 13, no. 10, pp. 1327-1344, Oct. 2004, doi: 10.1109/TIP.2004.834669

[6] Helmut Dersch, "Panorama tools: open source software for immersive imaging," The international VR photography conference proceedings, vol. 1, 2007

[7] "BetterLight," https://www.betterlight.com/index.html (accessed 2/3/2021)

[8] Henry Dietz and Paul Eberhart, "An Ultra-Low-Cost Large-Format Wireless IoT Camera," Electronic Imaging, Imaging Sensors and Systems, pp. 70-1 - 70-7(7), 2021 doi: 10.2352/ISSN.2470-1173.2021.7.ISS-070

[9] "Professional Digital Cameras and Jenoptik - a Long Tradition," in G. I. T. Imaging and Microscopy, vol. 6, pp. 26-27, July 2004.

[10] Ricoh Imaging Company, Ltd., "The advanced Pixel Shift Resolution System II for super-high-resolution images," https://www.ricoh-imaging.co.jp/english/products/k-1-2/feature/02.html (accessed 2/15/2024)

[11] DPReview, "Sony announce new RGBE CCD," https://www.dpreview.com/articles/1471104084/sonyrgbeccd (accessed 2/15/2024)

[12] LibRaw LLC, "PixelShift2DNG: Convert Sony and Pentax Pixel Shift Files to DNG," https://www.fastrawviewer.com/PixelShift2DNG (accessed 2/15/2024)

[13] LibRaw LLC, "LibRaw raw image decoder," https://www.libraw.org/ (accessed 2/15/2024)

[14] Henry Dietz, "The Aggregate: PARSEK," http://aggregate.org/DIT/PARSEK (accessed 2/15/2024)

[15] OpenCV, https://opencv.org/ (accessed 2/15/2024)

[16] Dave Coffin, "Decoding raw digital photos in Linux," https://www.dechifro.org/dcraw/ (accessed 2/15/2024)

[17] Jef Poskanzer, "NETPBM: Extended portable bitmap toolkit," 1993, https://netpbm.sourceforge.net/ (accessed 2/15/2024)

[18] Henry Dietz, Paul Eberhart, John Fike, Katie Long, Clark Demaree, Jong Wu, "TIK: a time domain continuous imaging testbed using conventional still images and video," in Proc. IS&T Intl. Symp. on Electronic Imaging: Digital Photography and Mobile Imaging XIII, 2017, pp 58 - 65, https://doi.org/10.2352/ISSN.2470-1173.2017.15.DPMI-081

[19] Henry Dietz, "An improved raw image enhancement algorithm using a statistical model for pixel value error," in Proc. IS&T Intl. Symp. on Electronic Imaging: Computational Imaging, 2022, pp 151-1 - 151-6, https://doi.org/10.2352/EI.2022.34.14.COIMG-151

[20] Georgios D. Evangelidis, and Emmanouil Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," IEEE transactions on pattern analysis and machine intelligence 30, no. 10, pp. 1858-1865, 2008

[21] A. S. Fruchter and R. N. Hook, "Drizzle: A method for the linear reconstruction of undersampled images," Publications of the Astronomical Society of the Pacific 114, no. 792 (2002): 144