Random Coded Sampling for High-Speed HDR Video

Travis Portz Li Zhang Hongrui Jiang
University of Wisconsin–Madison

http://pages.cs.wisc.edu/~lizhang/projects/hs-hdr/

Abstract

We propose a novel method for capturing high-speed, high dynamic range video with a single low-speed camera using a coded sampling technique. Traditional video cameras use a constant full-frame exposure time, which makes temporal super-resolution difficult due to the ill-posed nature of inverting the sampling operation. Our method samples at the same rate as the traditional low-speed camera but uses random per-pixel exposure times and offsets. By exploiting temporal and spatial redundancy in the video, we can reconstruct a high-speed video from the coded input. Furthermore, the different exposure times used in our sampling scheme enable us to obtain a higher dynamic range than a traditional camera or other temporal super-resolution methods. We validate our approach using simulation and provide a detailed discussion on how to make a hardware implementation. In particular, we believe that our approach can maintain a 100% light throughput similarly to existing cameras and can be implemented on a single chip, making it suitable for small form factors.

1. Introduction

Video cameras are a fairly ubiquitous technology in the modern world: automobiles use cameras for extra safety features, manufacturers use vision systems for quality control and automation, surgeons use small cameras for minimally invasive procedures, and mobile phones often have multiple cameras capable of video capture. Our motivation is to squeeze as much performance out of a camera as possible, and conventional video acquisition systems leave plenty of room for improvement.

A conventional video camera has a fixed frame rate and a global shutter (or, in the case of many CMOS cameras, a rolling shutter). Constrained by a constant exposure time, the performance of a conventional camera is limited by the scene it is trying to capture. If we are shooting a scene with significant motion and want to be able to grab sharp frames out of the video, we either have to use a short exposure time that has motion aliasing and captures less light (resulting in a poor signal-to-noise ratio) or must attempt to deblur the frames, which is very difficult and prone to artifacts in the presence of spatially-varying motion blur.

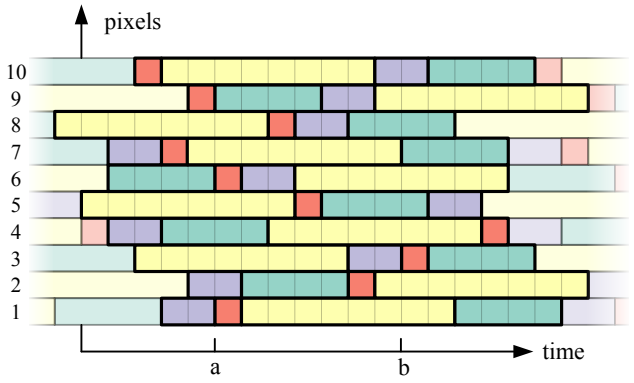

Figure 1. An illustration of our random coded sampling scheme on a 10-pixel camera. Each row represents the sampling intervals of a pixel. The width of each rectangular box represents the exposure time. Each pixel repeatedly uses a sequence of randomly permutated 4 exposure values. Such sampling schemes can be implemented on a single imaging chip, where we read out only a subset of pixels at each frame and skip pixels that are still exposing. For example, at time a, 3 pixels (1, 6, and 9) will be read out; at time b, 2 pixels (3 and 7) will be read out. This sampling scheme allows 100% light throughput. We seek to reconstruct a high-speed HDR video from the samples.

The light conditions of the scene are important as well. A low-light scene might require a longer exposure, resulting in blur, or a scene with a large dynamic range might require a short exposure to avoid saturation in the bright regions at the expense of noise in the dark regions. Being able to capture a wide dynamic range while still handling motion well is a desirable camera trait.

In this paper, we propose a novel video acquisition method that can simultaneously provide better temporal resolution (in the form of a higher frame rate) and a higher dynamic range than an equivalent conventional camera. We accomplish this using a random coded sampling scheme by which different pixels capture different sequences of exposures (as shown in Figure 1) and a novel algorithm that reconstructs the high-speed HDR video by exploiting the spa-

tial and temporal redundancy of natural videos. With our sampling scheme, we show that we do not have to solve the challenging issue of spatially-varying kernels to reduce motion blur; rather, deblurring can be done along the time axis, where integration over the known exposure times acts as "kernels". We validate our approach using simulation and provide a detailed discussion on how to make a hardware implementation. In particular, we believe that our approach is hardware implementable on a single chip and can maintain a 100% light throughput (meaning it has no gaps between or within exposures) similarly to existing cameras.

2. Related Work

Our work is related both to compressive sensing of video and to high dynamic range video. To our knowledge, no previous work provides both features at the same time. Our reconstruction method is also similar to works on temporal super-resolution of video in some respects.

2.1. Compressive Sensing of Video

Substantial research has been done in recent years on using different image acquisition methods to get high frame rate video at lower sampling rates. One approach is to randomly sample a subset of pixels from each frame and attempt to reconstruct the full frame [12]. Gupta *et al*. [7] arrange the samples in a manner that allows voxels to be interpreted as low spatial resolution (SR) and high temporal resolution (TR) in fast moving regions or high SR and low TR in static regions.

Hitomi *et al*. [9] use per-pixel exposure offsets to produce a coded exposure image, which is converted to a high-speed video by doing a sparse reconstruction with a learned over-complete dictionary. Their sampling scheme can be implemented on a single sensor, but it uses a constant exposure time with less than 50% light throughput. The need for training a large dictionary is also a drawback.

Another approach is to use a flutter shutter, which modulates the incoming light during a single exposure to retain more high-frequency information. Holloway *et al*. [10] use a global flutter shutter and video priors to reconstruct a high-speed video. Gupta *et al*. [8] capture videos from multiple cameras with different flutter shutters and combine them by solving a linear system. While this method could be modified to provide HDR video given the different exposures, the multiple camera requirement and the use of a flutter shutter makes it less versatile.

Reddy *et al*. [15] use a per-pixel flutter shutter to modulate low-speed exposures and then reconstruct a high-speed video using wavelet-domain sparsity and optical flow as regularization. Sankaranarayanan *et al*. [16] design the per-pixel flutter shutter in a manner that provides a low-speed preview video without advanced processing. The preview can be used for motion estimation as part of the high-speed recovery. Per-pixel flutter shutters require advanced hardware such as liquid crystal on silicon or digital micromirror devices that would be difficult to fit into smaller cameras.

None of the above works provide 100% light throughput or HDR output. Gu *et al*. [6] propose a coded rolling shutter scheme for CMOS sensors which can either utilize a staggered read-out to produce high-speed video or utilize row-wise exposures to produce low-speed HDR output. The exposure times can be set adaptively for scenes where the dynamic range is mainly spanned vertically, or they can be alternated between rows at the cost of vertical resolution. This method, however, does not produce high-speed and high dynamic range video at the same time.

2.2. HDR Video

Our work is also related to HDR video techniques that do not offer increases in temporal resolution. One approach for HDR video is to have groups of neighboring pixels on a sensor use different exposure times [13]. However, this approach results in a loss of spatial resolution and may require deblurring of the longer exposures to properly match the short exposures. Another approach is to combine videos from multiple cameras with different global exposure times [14]. This approach also requires deblurring of the longer exposures.

HDR video can also be obtained by post-processing a traditional constant exposure video. Bennett and McMillan [2] use spatial and/or temporal filtering based on a pixel's motion to reduce noice and increase the exposure time of dark pixels. Fast moving pixels use less temporal filtering to avoid ghosting and motion blur. Another approach is to denoise a sequence of many frames using optical flow, effectively increasing the video's dynamic range [20]. These approaches are designed for underexposed videos or low-light scenes and do not address saturation.

Instead of using multiple cameras with different exposure times, Tocci *et al*. [18] use a single camera with beam splitters and multiple image sensors. The beam splitters reflect different amounts of light to each sensor, so the same exposure time can be used throughout. The setup for this design still requires extra space, and all of the parts must be carefully aligned.

Narasimhan and Nayar [11] propose a multisampling technique that assigns different pixels different exposures using a spatially-varying neutral density filter (like a Bayer pattern filter but for intensity rather than color). No deblurring is necessary since the pixels use the same exposure time, but the use of a neutral density filter does result in a loss of light throughput.

In general, these HDR video methods can provide much more significant increases in dynamic range than our method by having more separation between the shortest and

longest exposures; however, they do not provide an increase in frame rate.

2.3. Temporal Super-Resolution

The spatial and temporal redundancy of videos that is exploited in our reconstruction algorithm is also exploited by Sharar *et al*. [17] to perform space-time super-resolution on conventional videos. Their method relies on searching for redundancy across scales (different spatial sizes or different speeds of motion) or at subpixel and subframe shifts in order to increase resolution, whereas our method relies on matching samples with different exposure times or offsets from our coded sampling input.

3. Random Coded Sampling

In our coded sampling scheme, each pixel location randomly selects a permutation of exposure times and an offset as illustrated in Figure 1. We use powers of two for the exposure times to get a compression factor of $n : \log_2(n+1)$. Our implementation uses $n = 15$ to provide approximately a 4x increase in frame rate and $\{1, 2, 4, 8\}$ as the set of exposures. The random sequences of exposures are repeated temporally to get longer videos. There are no gaps between exposures, so our sampling scheme maintains 100% light throughput. Furthermore, no gain compensation is performed on the different exposures times, so the eight-frame samples are three stops brighter than the one-frame samples.

At the end of each frame, approximately one fourth of the pixels on the sensor need to be sampled. This can be implemented using a sensor with row and column addressing to activate the electronic shutters of the pixels being sampled followed by a partial read-out of the sensor, skipping pixels that are still exposing. As in a traditional camera, the pixels start exposing immediately after the electronic shutter is activated. The read-out happens in the background and must complete before the electronic shutter is activated again for the next frame's different (but not disjoint) set of pixels [1, 5].

The row and column addressing feature already exists in some image sensors. Current commercial sensors do not provide the level of control necessary to do the partial read-out; however, a custom VLSI chip could implement the control without any new technology (e.g., new types of photoreceptors) in the actual image sensor. The benefit of this sampling scheme with a partial read-out is that, by skipping unsampled pixels in the analog-to-digital conversion and I/O process, we can achieve a higher frame rate than would be possible with a full read-out of the same image sensor, using the same sampling bandwidth.

3.1. Simulation

Since no commercial image sensors support our coded sampling scheme yet, we simulate such a sensor by capturing high-speed ground truth videos and summing pixel values temporally to match the sampling pattern. We captured the ground truth videos for our examples at 200 frames per second using a 14-bit Point Grey Grasshopper. We used a 0dB gain setting to avoid noise in the ground truth data as much as possible.

Our simulations model read noise, shot noise, and the bit depth of the A/D converter. Suppose $x \in [0, 8]$ is the ground truth sample value (assuming the longest samples are eight frames). Let the random variable $Z \sim \text{Poisson}(rx)$ be the number of photons captured, where the parameter r is the number of photons that correspond to a ground truth value of 1.0. Let $V \sim \mathcal{N}(0, \sigma^2)$ be the read noise in photons. The simulated noisy sample is then

$$Y = \frac{1}{2^d - 1} \left\lfloor (2^d - 1) \text{ clamp} \left(\frac{\min(Z, w) + V}{s} \right) \right\rfloor, \quad (1)$$

where d is the bit depth, w is the full well capacity (photons), s is the number of photons mapped to the maximum output value of 1.0 (based on the gain setting of the camera), and $\text{clamp}(x) = \max(0, \min(1, x))$. The output value Y is in the range $[0, 1]$. Saturation can occur either from the full well capacity being exceeded or from the A/D converter. In all of our examples, saturation is caused by the A/D converter since we use $s < w$.

4. Reconstruction

We reconstruct high-speed video from the low-speed coded sampling video by exploiting spatial and temporal redundancy. This requires block matching to find similar video patches within frames and across frames. Three-dimensional space-time patches are used as blocks. Once we have groups of similar patches, we use their different sampling patterns to effectively deblur any longer exposure samples and fill in saturated pixels. No ill-posed spatially-varying blur kernel estimation is needed since the processing is done temporally where integration over the known exposure time acts as the "kernel".

The reconstruction is done in two stages. The first stage must perform block matching on the sampled input data and do the optimization using these less accurate matches. In the second stage, we perform matching on the intermediate output of the first stage to get better matches and, thus, sharper results. Both stages use the same optimization method once the matches have been obtained.

4.1. Block Matching

The block matching in the first stage performs an exhaustive search for the K-nearest space-time patches within a three-dimensional search window around the reference patch. To compute a patch distance between two differently sampled patches, samples captured with shorter exposures

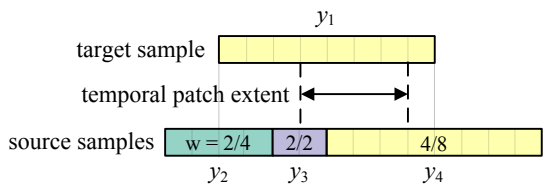

Figure 2. Example of the resampling/blurring done to compute patch distances for the sampled input data. In this example, the blurred source value that we compare with y_1 will be $z = y_2/2 + y_3 + y_4/2$ and its variance will be $\sigma^2 = 1/4 + 1 + 1/4$. In this example, we assume the temporal patch extent is 4 frames, marked by the black dashed lines. Since y_1 covers the full 4 frame extent of the patch, the residual $(y_1 - z)^2$ will have a weight of $4/(1 + \sigma^2)$ in the total patch distance. See the supplementary material for detailed formulae in general cases.

(we call source samples) are blurred to match a sample captured with a longer exposure (we call the target sample).

Figure 2 shows how samples are blurred and compared. The source samples are weighted based on their coverage of the target sample, and a variance for the blurred value is computed assuming each individual sample has the same variance. The squared difference between the target sample and the blurred source value contributes to the patch distance with a weight proportional to the target sample's coverage of the current patch divided by the variance of the residual. See the supplementary material for the code of this patch distance algorithm.

In the second stage, we use a combination of optical flow and search-based block matching on the estimated video from the first stage. Optical flow is computed over the entire video using a GPU implementation of the TV-L1 optical flow algorithm [19]. We concatenate and refine flows to get trajectories of length up to some temporal radius (we used a radius of 5 for all of our results). Forward and backward consistency checks are used to terminate trajectories early if necessary. Search-based block matching is used to reach a target number of matches (20 used in our examples) in addition to the matches from flow. Block matching in this stage uses a simple SSD patch distance because the result of the first stage is a densely-sampled high-speed video.

We will use the following notation for matching in the remainder of the paper:

1. Let i be the location of a reference patch.
2. G_i is its set of matches with $j \in G_i$ being the location of a match.
3. The patch distance for a given match is d_{ij} (which is a normalized, mean squared error value).

4.2. Objective Function

We propose an objective function using the matching results for both a data term and a regularization term. To make it robust to bad matches, we use the L1 norm for both terms. If the input value is saturated in the data term, we use a one-sided version of the L1 norm (shown in Figure 3 and referred to as $\|.\|_{1^*}$) so that brighter values are not penalized. The objective function is then given by:

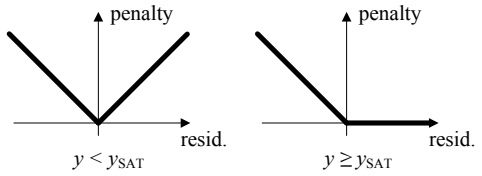

Figure 3. The one-sided version of the L1 norm used to handle saturation in the input. Only residuals where the estimate is less than the saturated input are penalized.

$$\sum_i \sum_{j \in G_i} w_{ij} \left(\frac{\tau_{\max}}{\tau_j} \|S_{i|j}x - y_j\|_{1^*} + \gamma \|x_i - x_j\|_1 \right), \tag{2}$$

where $S_{i|j}$ samples the high-speed data x at location i using the sampling pattern at location j; we use y_j, the input at location j, to constrain $S_{i|j}x$. The weights w_{ij} are given by

$$w_{ij} = \exp\left(-\frac{d_{ij}}{2\sigma^2}\right). \tag{3}$$

The $\tau_{\max}/\tau_j$ term gives more weight to constraints with shorter exposures to provide sharper results and better reconstruction of saturated regions. Note that τ_j actually varies on a per-sample basis rather than a per-patch basis, but we use the notation above for simplicity.

The second stage of the reconstruction also assigns a smaller weight to constraints from the search-based block matching. This helps to improve the sharpness provided by the more reliable optical flow matching. Specifically, we reduce the weights by a factor of about 0.5 for search-based block matching constraints.

We can rewrite the objective function as

$$\|Ax - b\|_{1^*} + \gamma \|Fx\|_1, \tag{4}$$

where A is a sparse matrix with the weighted $S_{i|j}$ terms as it rows, b is a column vector of the weighted y_j inputs, and F is a sparse matrix with $w_{ij}(\delta_i - \delta_j)$ as its rows (δ_i is a row vector with a 1 at location i).

4.3. Optimization

We minimize the objective function in Eq. (4) using the ADMM algorithm [3] by formulating it as a constrained convex optimization:

$$\begin{aligned} &\text{minimize } \|z_1\|_{1^*} + \|z_2\|_1 \\ &\text{subject to } \begin{bmatrix} A \\ \gamma F \end{bmatrix} x - \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}. \end{aligned} \tag{5}$$

The primal-dual algorithm for solving this optimization problem is shown in Algorithm 1.

Algorithm 1 ADMM optimization to minimize Eq. (5).

$M \coloneqq A^{\mathrm{T}}A + \gamma^2 F^{\mathrm{T}}F$
$z \coloneqq u \coloneqq 0$
for $k = 1$ to iters **do**
  $d \coloneqq A^{\mathrm{T}}(b + z_1 - u_1) + \gamma F^{\mathrm{T}}(z_2 - u_2)$
  $x \coloneqq M^{-1}d$
  $r_1 \coloneqq Ax - b$
  $r_2 \coloneqq \gamma Fx$
  $z_1 \coloneqq \text{SoftThreshold}_{1^*}(r_1 + u_1, \rho)$
  $z_2 \coloneqq \text{SoftThreshold}_{1}(r_2 + u_2, \rho)$
  $u \coloneqq r + u - z$
end for

The $\text{SoftThreshold}_{1^*}$ function is a soft thresholding operation modified for our one-sided penalty function:

$$z = \text{SoftThreshold}_{1^*}(v, \rho) = \begin{cases} \min\left(v + \frac{1}{\rho}, 0\right), & v \leq 0 \\ \max\left(v - \frac{1}{\rho}, 0\right), & v > 0 \text{ and not saturated} \\ v, & \text{otherwise} \end{cases} \quad (6)$$

where "not saturated" means the corresponding input value from y is less than y_{SAT}. The SoftThreshold_1 function is the standard soft thresholding operation for the L1 norm. The ρ parameter affects the convergence of the algorithm: larger values result in smaller steps between iterations. We use $\rho = 2$ in the first stage and $\rho = 10$ in the second stage.

We use a sparse GPU implementation of the conjugate gradient method to solve the $x \coloneqq M^{-1}d$ step. The A, A^{T}, F, and F^{T} operations in the loop are performed without explicitly forming the matrices for memory purposes. The main bottleneck in the algorithm is forming the sparse matrix M, which we accomplish using the Eigen library's RandomSetter class [4].

5. Results

The ground truth videos used in our examples (referred to as the *fruit*, *circles*, *pencils*, *games*, and *cat* videos) contain a variety of motion and lighting conditions. Using Eq. (1), we simulate an 8-bit camera with a read noise level of $\sigma = 10$ photons and a full well capacity of $w = 20,000$ photons. For the *fruit*, *pencils*, *games*, and *cat* videos, we use $r = 1000$ photons and $s = 2000$ photons. Thus, the 1-frame and 2-frame exposures are never saturated, the 4-frame exposures are saturated if the average of ground truth values in the exposure interval is greater than 0.5, and the 8-frame exposures are saturated if the average of ground truth values is greater than 0.25. In the *circles* video, which has a lower dynamic range, we use $r = 500$ photos so that only the 8-frame exposures can be saturated.

To compare our sampling scheme with conventional constant exposure settings, we also simulate constant exposure time videos at a frame rate of 1/4 the high-speed frame rate; the factor 1/4 is used because our random sampling scheme has a compression ratio of 4:1, as discussed in Section 3. In particular, we simulate two constant exposure settings: a long exposure time that corresponds to 4 high-speed frames (referred to as 4x exposure) and a short exposure time that corresponds to 1 high-speed frame (referred to as 1x exposure). These videos use the same camera parameters as our coded sampling simulations.

In the first stage, we use 6x6x4 blocks; reference blocks have a spatial step size of 3 and no overlap temporally. Each reference patch has at most 12 block matches with no error threshold. In the second stage, we use 6x6x3 blocks for the search-based block matching. The *fruit* and *cat* videos use 4x4x1 blocks for the optical flow based matching; the *pencils*, *games*, and *circles* videos use 2x2x1 blocks.

For every video, we do 10 ADMM iterations in the first stage and 50 iterations in the second stage. More iterations are required in the second stage because ρ is larger, resulting in smaller steps and slower convergence. See the supplementary material for a full listing of parameter values.

When displaying results, we show the raw images with a constant gain factor so that the brightness levels match the ground truth. We also show the images using a global curve adjustment that provides more contrast in the dark regions. The coded sampling videos are displayed with the samples divided by their exposure time to match brightness levels. Saturated values from long exposures appear darker than their unsaturated, short exposure counterparts when displayed in this manner.

Figure 4 shows results for the *fruit* video. This video has large camera motion, which is spatially varying due to depth in the scene and slight rotation. The 4x exposure video contains significant blur and has saturated highlights, whereas the 1x exposure video is sharp but has poor signal-to-noise ratio in the dark regions. Our result is nearly as sharp as the 1x exposure with low noise and good reproduction of the highlights. In addition, it has a 4x higher frame rate than the constant exposure time videos.

The other videos have similar results in general. Surprisingly, our results have lower noise than the "ground truth" data in most cases. Our method struggles somewhat with more severe and complex motions, such as the top of the rotation in Figure 5 and the dice in Figure 7. Also, 50 iterations of ADMM was not enough to get complete convergence in the saturated region of the hand in Figure 7. This slow convergence is due to the one-sided penalty function being in effect for most of the constraints.

The *pencils* video in Figure 6 shows our method performing well on motion of finer objects as long as the size of the motion is not too large. Our method can also handle large regions containing saturation, as seen with the paper in the background. The *cat* video in Figure 8 has little motion,

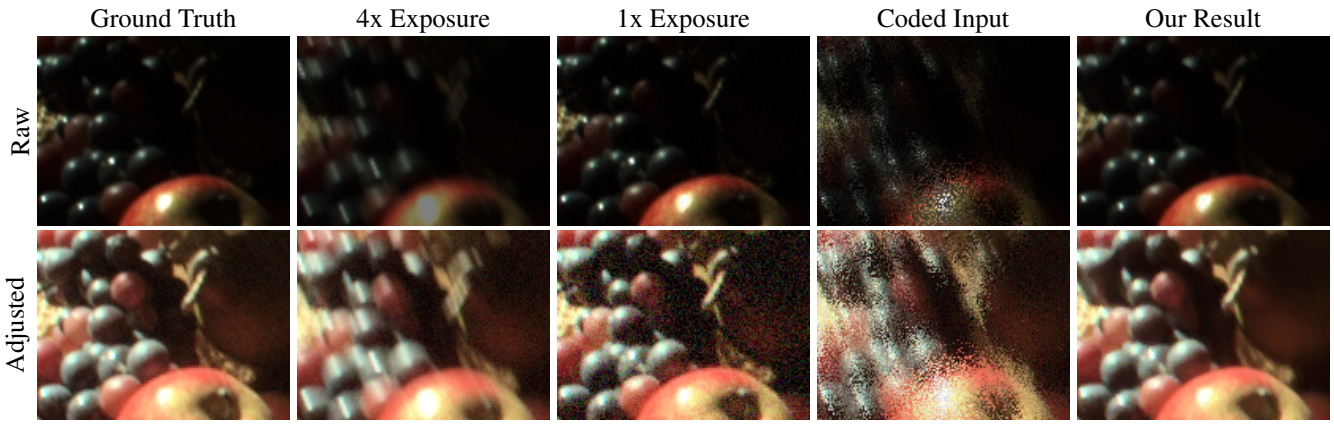

Figure 4. Results on the *fruit* video. This video contains large camera motion, and as a result, the 4x exposure video is significantly blurred and has saturated highlights; the highlights in the raw image of the 4x exposure appear darker because its image intensity has been scaled to match the ground truth brightness. Our method is able to produce a sharp, high-speed video from the coded sampling input data.

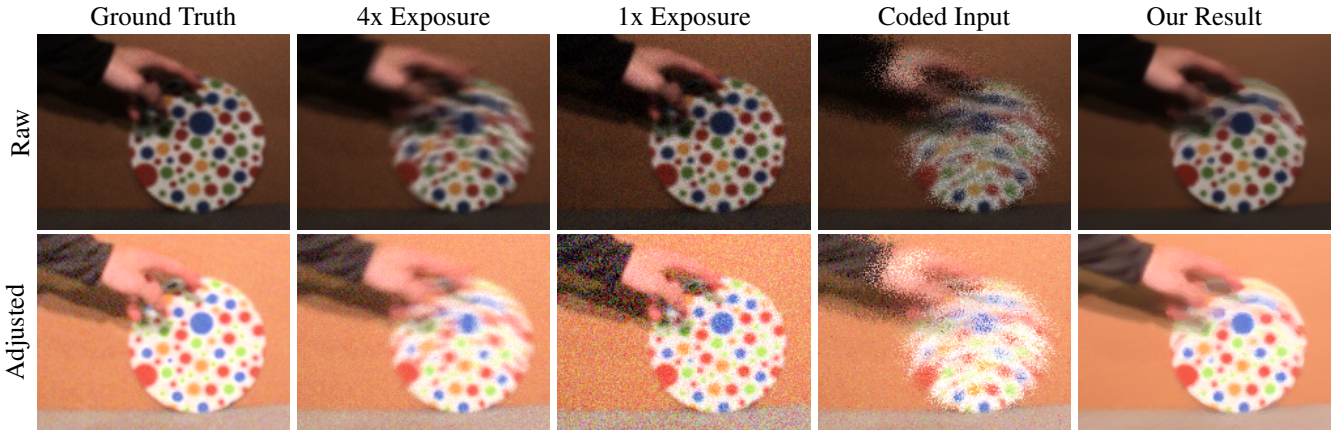

Figure 5. Results on the *circles* video. The object in the video is rolling along the surface, so the motion is larger at the top. The coded input has some saturation in the white regions of the object, but this video has a low dynamic range overall. Our result is sharper than the 4x exposure video in the middle and bottom parts of the object, but still contains some blur at the top of the object. Our method provides very low noise output with a major improvement to the dark sleeve compared to the 1x exposure video.

so the 4x exposure video performs well apart from saturation in bright regions of the fur; our method recovers the bright regions. Our result also has near perfect reproduction of the darker background regions, which are very noisy in the 1x exposure video.

More results, including reconstructed high-speed videos, are available in the supplementary material.

6. Conclusions

We have proposed a method for capturing high-speed HDR video using random coded sampling. Our main contributions are i) the design of a sampling scheme that can be implemented on a single chip and supports HDR imaging while maintaining 100% light throughput, and ii) the development of an algorithm for reconstructing the high-speed HDR video from the sampled data. Our method is capable of producing sharp videos with low noise and a higher dynamic range than possible using a constant exposure time.

6.1. Future Work

Our current reconstruction algorithm requires a global sparse system to be solved due to the type of regularization we use. Building this sparse system is a major bottleneck in the algorithm (the constraints must be accumulated without knowing the exact structure of the matrix in advance, taking several minutes for our test videos) and could possibly be avoided by using a different regularization method that permits local updates with aggregation. Future work can explore different types of regularization that may also help to preserve detail in regions with fast, complex motion.

In addition, our current simulation works with a full RGB input, which would require hardware implementation on vertically stacked photodiodes that are used in Foveon X3 sensors. A more popular approach would likely use a Bayer pattern filter. Conventional demosaicing as a preprocessing step would not be possible since neighboring pixels have different exposure times and offsets. Future work can

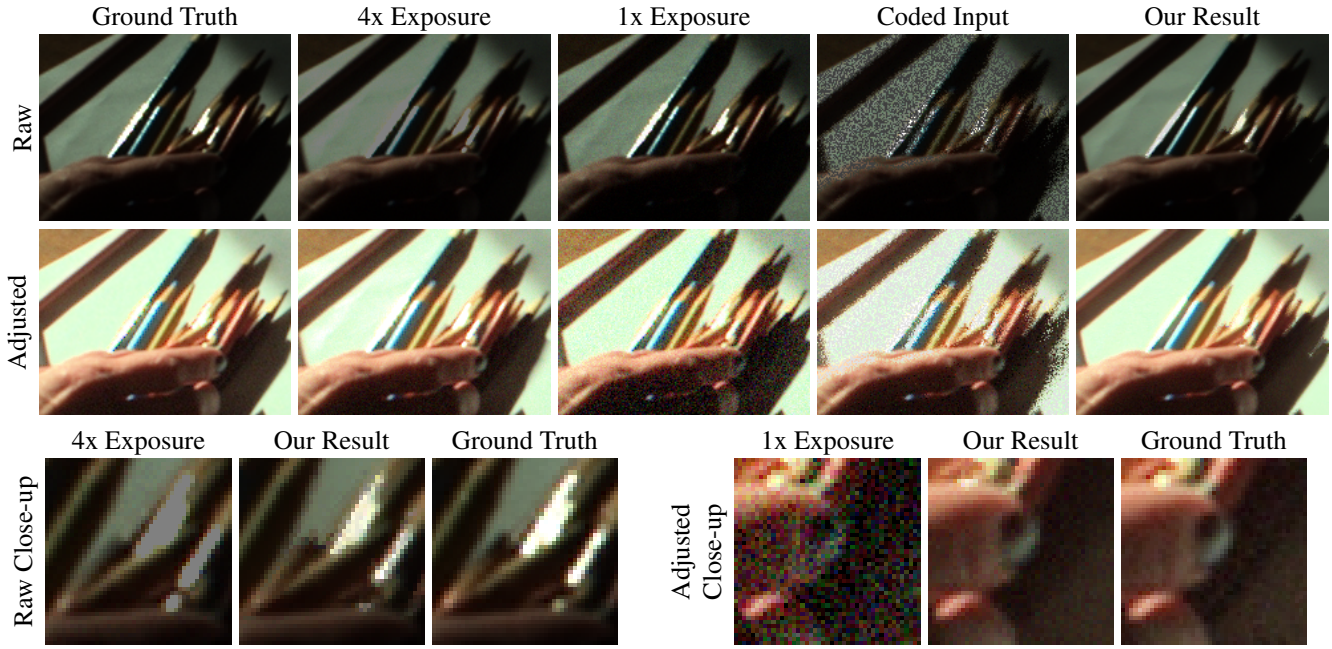

Figure 6. Results on the *pencils* video, which contains multiple motions of finer objects. Our method performs well despite large amounts of saturation in the coded input. As shown in the close-ups in the third row, our result has better contrast in the highlights than the 4x exposure video and better signal-to-noise ratio in the shadows than the 1x exposure video. The frame rate improvement can be seen in the videos in the supplementary material.

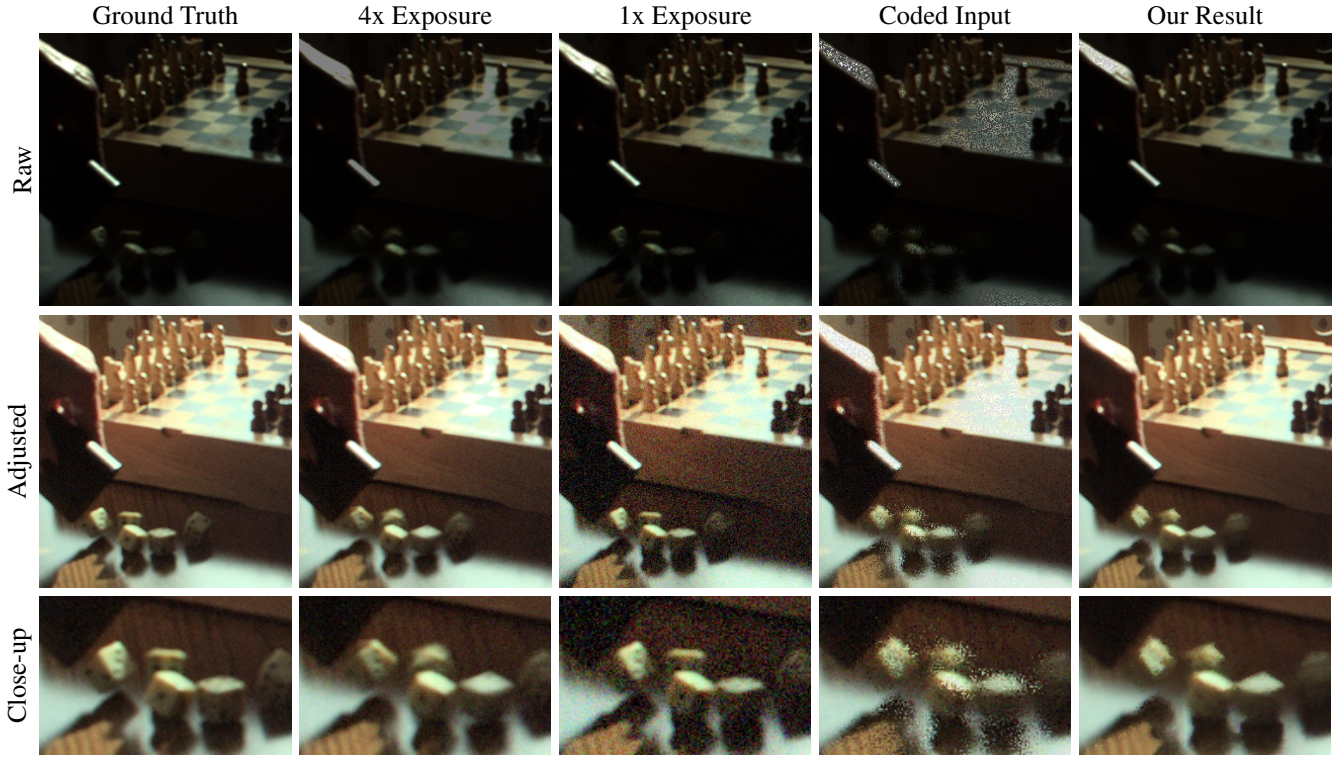

Figure 7. Results on the *games* video, which contains fast and complex motion of rolling dice. While our method is able to correct the sampling artifacts of the coded input, it does not recover as much detail as the constant exposure time videos. However, the improvements in dynamic range remain such as on the surface of the chess board in the raw images (compared to the 4x exposure) and in the shadow of the chess board in the adjusted images (compared to the 1x exposure).

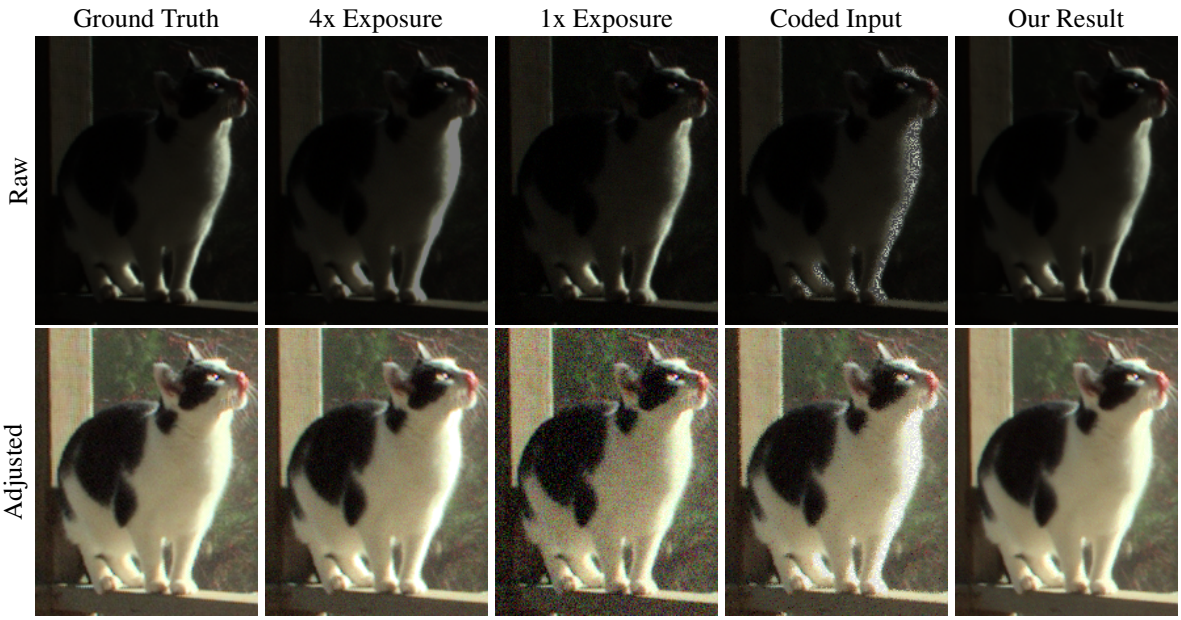

Figure 8. Results on the *cat* video, which has very small motion. Our result is not saturated in the bright chest and leg regions like the 4x exposure video, as seen in the raw images, and has much higher quality compared to the 1x exposure video, as seen in the adjusted images.

adapt the matching in the first stage and the data term in both stages to handle a Bayer pattern input. Actual hardware implementation of the sampling scheme may also require a tiled sampling pattern rather than a completely random pattern in order for the partial read-out to be feasible.

References

[1] Andor Technology. Rolling and Global Shutter. Technical report. http://www.andor.com/pdfs/literature/Tech_Note_Rolling_Global_Shutter.pdf.

[2] E. Bennett and L. McMillan. Video enhancement using per-pixel virtual exposures. *ACM Trans. Graph.*, 24(3):845–852, 2004.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[4] Eigen. *RandomSetter Class Template Reference*, December 2012.

[5] T. Fellers and M. Davidson. Concepts in Digital Imaging Technology: Digital Camera Readout and Frame Rates. http://micro.magnet.fsu.edu/primer/digitalimaging/concepts/readoutandframerates.html.

[6] J. Gu, Y. Hitomi, T. Mitsunaga, and S. Nayar. Coded rolling shutter photography: Flexible space-time sampling. In *ICCP*, 2010.

[7] M. Gupta, A. Agrawal, A. Veeraraghavan, and S. Narasimhan. Flexible Voxels for Motion-Aware Videography. In *ECCV*, 2010.

[8] M. Gupta, A. Veeraraghavan, and S. Narasimhan. Optimal coded sampling for temporal super-resolution. In *CVPR*, 2010.

[9] Y. Hitomi, J. Gu, M. Gupta, T. Mitsunaga, and S. Nayar. Video from a Single Coded Exposure Photograph using a Learned Over-Complete Dictionary. In *ICCV*, 2011.

[10] J. Holloway, A. Sankaranarayanan, A. Verraraghavan, and S. Tambe. Flutter Shutter Video Camera for Compressive Sensing of Videos. In *ICCP*, 2012.

[11] S. Narasimhan and S. Nayar. Enhancing Resolution Along Multiple Imaging Dimensions Using Assorted Pixels. *TPAMI*, 27(4):518–530, 2005.

[12] J. Park and M. Wakin. A multiscale framework for Compressive Sensing of video. In *PCS*, 2009.

[13] T. Poonnen, L. Liu, K. Karia, M. Joyner, and J. Zarnowski. A CMOS video sensor for High Dynamic Range (HDR) imaging. In *Asilomar SSC*, 2008.

[14] V. Ramachandra, M. Zwicker, and T. Nguyen. HDR Imaging From Differently Exposed Multiview Videos. In *3DTV*, 2008.

[15] D. Reddy, A. Verraraghavan, and R. Chellappa. P2C2: Programmable Pixel Compressive Camera for High Speed Imaging. In *CVPR*, 2011.

[16] A. Sankaranarayanan, C. Studer, and R. Baraniuk. CS-MUVI: Video Compressive Sensing for Spatial-Multiplexing Cameras. In *ICCP*, 2012.

[17] O. Shahar, A. Faktor, and M. Irani. Space-time super-resolution from a single video. In *CVPR*, 2011.

[18] M. Tocci, C. Kiser, N. Tocci, and P. Sen. A Versatile HDR Video Production System. In *SIGGRAPH*, 2011.

[19] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow. In *DAGM*, 2007.

[20] L. Zhang, A. Deshpande, and X. Chen. Denoising versus Deblurring: HDR Techniques Using Moving Cameras. In *CVPR*, 2010.