MATLAB User Guide for QIS Simulations

Omar Elgendy and Stanley Chan

Statistical Signal and Image Processing Lab, Purdue University

https://engineering.purdue.edu/ChanGroup/

Version 1.0

This user guide documents the usage of the MATLAB simulations presented in [1]. Acknowledgment of this package should be given to the following reference.

[1] O. A. Elgendy and S. H. Chan, "Optimal threshold design for quanta image sensor," *IEEE Trans. Computational Imaging*, in Press, Oct. 2017. [Available Online]: http://arxiv.org/abs/1704.03886

1 Introduction

Quanta Image Sensor (QIS) is a class of solid-state image sensors designed to solve the miniaturization problems of CMOS sensor and envisioned to be the next generation imaging device after it.

The main idea of QIS design is to allow the pixel size to decrease as much as possible (e.g. 100−200 nm pitch [2]) to form miniature pixels, called *jots*, with intentionally low full-well capacity (FWC) (1−200 photoelectrons [2]). Each jot has sub-electron readout noise (i.e., readout noise with standard deviation less than 0.3 electron [3, 4]) which enables it to have single-photon sensitivity and photon counting capability. The jot counts every incoming photon and produces a binary response "1" if the photon count exceeds a threshold q, and "0" otherwise. By making $q <$ FWC, the resulting signal has high SNR because of its binary nature.

Definitely, the binary quantization of photon counts leads to significant distortion in the output signal. To compensate for this aggressive quantization of light, QIS oversamples the light signal in space and time by having huge spatial resolution (e.g., 10^9 pixels per sensor with 200nm pitch per jot [4]) and huge temporal resolution or frame rate (e.g., 100k fps as reported in [5]), respectively. As a result, each output gray-scale pixel is formed by locally processing a 3d spatial-temporal kernel or a "cubicle" of $K \times K \times T$ binary jots, where K is the spatial kernel size and T is the number of temporal frames. This processing is usually referred to as *binning* and it is frequently used in low-light image processing to mitigate noise. By efficient processing of the cubicle of jots, the output pixel represents the incoming light intensity on these jots. Figure 1 shows

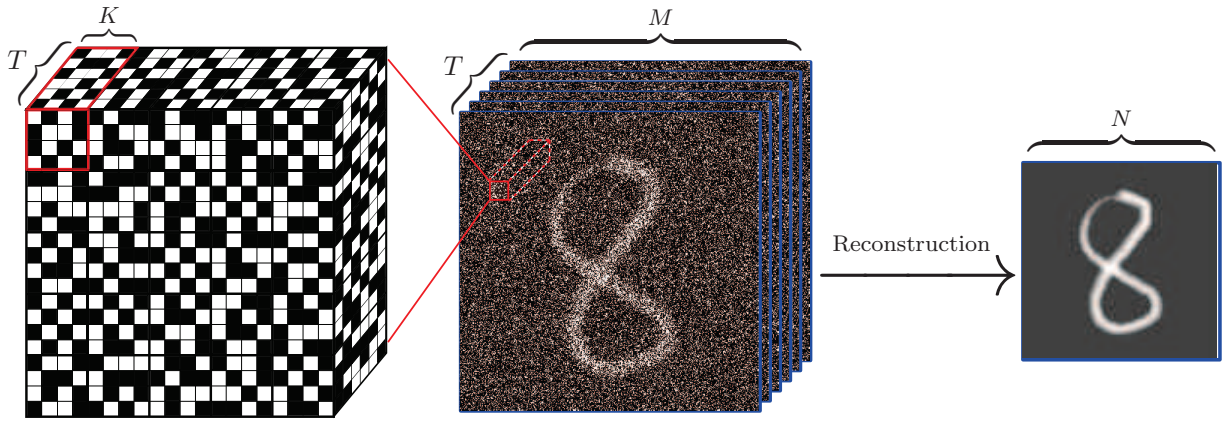

Figure 1: Image reconstruction of QIS data. Given T binary bit planes having high resolution $M \times M$, the reconstruction algorithm processes each $K \times K \times T$ cubicle of jots to form the $N \times N$ gray-scale image shown on the right, where $N = M/K$.

the QIS image formation process. The high spatial-temporal oversampling of QIS increases its dynamic range to levels even higher than CMOS and CCD.

This package presents the MATLAB code used to produce the results in [1]. First, we present the functions the reproduce important figures in the paper. Seconds, we present the functions used for experimental evaluation of QIS based on simulations.

2 Important Utility functions

To simulate the photon counts using the imaging model in Fig.2 in the paper, the following function can be used.

```
y = generatePhotonCounts(c,K,T,alpha,kernel)
```

This function takes the following inputs

- The ground truth image `c`
- The spatial oversampling factor `K`
- The temporal oversampling factor `T`
- The sensor gain `alpha`
- The point spread function of the interpolation filter `kernel`.

It generate a 3d array of size $mK \times nK \times T$ containing the photon counts, where m and n are the number of rows and columns in the ground truth image, respectively.

In the main paper, two quantization maps (Qmaps) can be generated: 1) Uniform Qmap, and 2) Adapted Qmap. These Qmaps can be generated by the function

```
function [QmapLR,QmapHR] =  generateQmap(imSize,K,type,opts)
```

This function takes the following inputs

- A 2d array containing the number of rows and columns `imSize`
- The Spatial oversampling factor `K`
- The type of the Qmap `type`, which can be one of the strings `'uniform'` or `'bisection'`.
- Other parameters stored in the variable `opts`

It produces the following outputs:

- A high resolution Qmap (`QmapHR`) used for quantizing the photon counts
- a low resolution Qmap (`QmapLR`) used for reconstructing the ML estimate after binning the binary measurements.

The ML estimate is obtained by running the function `imageReconstruct` which is defined as

```
function c = imageReconstruct(b,K,alpha,QmapLR)
```

This function takes the following inputs

- A 3d array of size $mK \times nK \times T$ containing the binary measurements `b`
- The spatial oversampling factor `K`
- The sensor gain `alpha`
- The low resolution Qmap `QmapLR`

and produces the ML estimate `c` as an output.

The bisection adaptation algorithm (Algorithm 1 in the paper) is implemented in the following function:

```
[Q_map,t_bisection,gamma_1f,B,errQ,Q_map_iter,gamma_iter]=BisectionAdapt(y0,qmax,
Qblocksize,oracleQmap,NbBisectionIter,Pixidx,flag)
```

This function takes the following inputs:

- A 3d array containing the photon counts `y0`
- The maximum threshold value `qmax`

- The size of block of pixels sharing the same threshold `Qblocksize`
- The oracle threshold map `oracleQmap` to calculate the MSE with iterations
- The desired number of iteration of the algorithms `NbBisectionIter`
- The indices of pixels at which the output is tracked.
- A flag used to specify whether to calculate the MSE with the oracle map or not

And produces the following outputs

- The adapted high resoultuin Qunatization map `Q_map`
- The elapsed time of the algorithm `t_bisection`
- The final proportions of ones in each kernel of jots `gamma_1f`
- The binary measurements at different iterations `B`
- The MSE with the oracle map at different iterations `Q_map_iter`
- The proportion of ones at different iterations `gamma_iter`

The Markov Chain adaptation algorithm proposed in [6] is implemented in the following function

```
function [Q_map,b,t_Markov,err_Qmap,Q_map_iter]=MCAdapt(y,qmax,oracleQmap,L,Beta,Pixidx)
```

One challenge in this algorithm is that it assumes temporal ovesampling T only and no spatial oversampling; whereas our algorithm assumes both spatial K and temporal oversampling T. To solve this problem, we reorganize the photon counts s to have an enlarged temporal oversampling $T_e = K^2T$ and no spatial oversampling. This function takes the following inputs

- A 3d array containing the photon counts `y`
- The maximum threshold value `qmax`
- The oracle threshold map `oracleQmap` to calculate the MSE with iterations
- The parameter `L` which defines the number of states per threshold in the Markov chain 2^L
- The staying probability `beta` which determines the probability that a state stays unchanged.
- The indices of pixels at which the output is tracked.

It produces the following outputs

- The adapted high resoultuin Qunatization map `Q_map`

- The binary measurements at different iterations `b`
- The elapsed time of the algorithm `t_Markov`
- The MSE with the oracle map at different iterations `Q_map_iter`
- The proportion of ones at different iterations `gamma_iter`

The reconstruction algorithm proposed in [7] is implemented in the following function.

```
function [IM_CR,PSNR_CR]=ConditionalResetAlg(y0n,Kt,Q_sequence,c0)
```

This function takes the following inputs

- A 3d array containing the photon counts `y0n`
- The spatial oversampling factor `Kt`
- The sequence of increasing (or decreasing) thresholds, a uniform threshold for each frame. `Q_sequence`
- The ground truth image `c0`

It produces the following outputs

- The reconstructed image `IM_CR`
- The reconstruction PSNR `PSNR_CR`

3 Demonstration File

The simulation file `demo.m` shows a demostration of how to simulate QIS binary measurements, and how to reocnstruct a grayscale image from them. First, the QIS simulation parameters are specified as follows

```
K           =  4;               % Spatial  Overasampling Factor
T_adapt     =  8;               % Frames for Threshold Adaptation
T_recon     =  22;              % Frames for Image Recosntruction
T           =  T_adapt+T_recon; % Temporal Overasampling Factor
Qsize       =  K;               % Kernel size of pixels that share the same threshold
qmax        =  16;              % Maximum Threshold
g           =  ones(K)/K^2;     % Box-car Interpolation kernel
```

After that, the ground truth image is read, downscaled, and transfomed to a grayscale image. The oracle threshold map is estimated Proposition 5 in the paper. The photon counts are generated using the `generatePhotonCounts` function as follows

```
IM = imresize(im2double(imread('./ImageData/SSIPDataset/IMG_0138.JPG')),0.1);
if size(IM,3)~=1;
    IM          =  rgb2gray(IM);
end
[rows,cols]     =  size(IM);
alpha           =  K^2*(qmax-1);    % Sensor gain
oracleQmap      =  floor(alpha*IM/K^2)+1;
y               =  generatePhotonCounts(IM,K,T,alpha,g);
```

The next step is to generate the adapted Qmap using the first T_{adapt} frames. This Qmap is used to generate the subsequent T_{recon} binary frames used for reconstruction.

```
opts.y          =  y;
opts.qmax       =  qmax;
opts.oracleQmap =  oracleQmap;
opts.T_adapt    =  T_adapt;
opts.Qblock     =  Qsize;
[QmapLR,QmapHR] =  generateQmap([rows,cols],K,'bisection',opts);
reconFrames     =  T_adapt + (1:T_recon);
b               =  1*(y(:,:,reconFrames)>=repmat(QmapHR,1,1,T_recon));
```

The final step is to reconstruct the grayscale image using the `imageReconstruct` function.

```
IM_qs           =  imageReconstruct(b,K,alpha,QmapLR);
PSNR            =  psnr(IM_qs,IM);
```

4 Reproduce Figures 1, 4, 5, 6 and 8

4.1 Motivation Figure

The motivation figure (Fig. 1 in the paper) can be reproduced by running the following simulation

`DrawMotivationFig.m`

In this simulation, QIS binary measruements are generated using a) low uniform threshold value ($q = 3$), b) high uniform threshold value ($q = 12$), and c) our proposed bisection adapted threshold map. For the uniform thresholds, the whole T frames are used for reconstruction; whereas for our adapted threshold map,

T_{adapt} and T_{recon} frames are used for adapting the threshold map and reconstruction the image, respectively, where $T = T_{\text{adapt}} + T_{\text{recon}}$.

4.2 Signal-to-Noise Ratio

An important performance metric of QIS is the signal-to-noise (SNR) ratio of its maximum likelihood (ML) estimate. Two variants of SNR were derived in the paper: 1) the output-referred SNR and 2) the exposure-referred SNR. The variation of these metrics with the light exposure can be drawn by running the simulation:

```
SNRcalculations.m
```

This simulation file also generates the SNR curves for different thresholds and different integration periods for HDR imaging. In this simulation, 4 integration times (normalized to the readout scan time) are used to generate an HDR image: 1, 0.2, 0.04, and 0.08. The SNR versus the light exposure values is drawn for low threshold value ($q = 1$) and high threshold value ($q = 16$). These sub-optimal methods are compared to our method which use the optimal threshold for each light exposure value.

4.3 Phase Transition Behavior

One of the main contributions in our paper is the phase transition behavior of the ML estimate for QIS images. The ML solution is asymptotically unbiased when the threshold lies in a certain interval of thresholds. Outside this interval, the ML solution deteriorates quickly. The relevant figures for this phenomenon are generated by running the simulation:

```
PhaseTransitionSimulations.m
```

In this simulation, Monte Carlo simulation is performed using 5000 realizations of the Poisson counts to get an estimate for the average ML solution. These realizations are generated using different spatial and temporal oversampling factors, K and T. For each combination of K and T, the ML solution is obtained using different values of quantization thresholds q.

4.4 Bisection Algorithm for Threshold Adaptation

We propose a bisection algorithm to adapt the quantization threshold q to the light exposure value. Assuming a temporal oversampling factor T, this algorithms uses T_{Adapt} frames to learn the best quantization threshold for each pixel (or a kernel of pixels) without reconstructing the image. After obtaining a quantization map of adapted thresholds, the image is reconstructed using the remaining frames, which are denoted by $T_{\text{Recon}} \stackrel{\text{def}}{=} T - T_{\text{Adapt}}$. This process is depicted in Figure 2. The simulation file

```
DrawBisectionFig.m
```

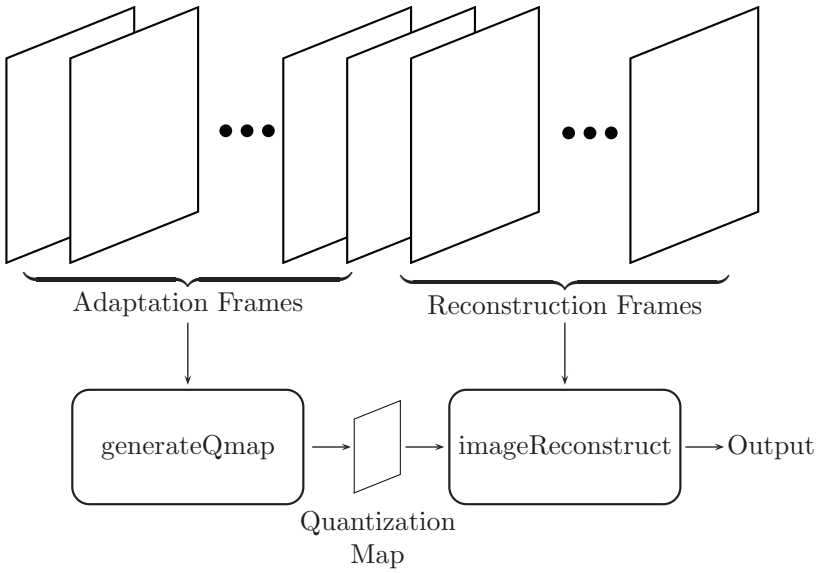

Figure 2: Our proposed image reconstruction technique: T_{Adapt} frames are used for adapting the threshold map. Then, this map is used for reconstructing the image using T_{Recon} frames. $T = T_{\text{Adapt}} + T_{\text{Recon}}$

draws a figure showing the operation of the bisection algorithm for an arbitrary scenario.

5 Reproduce Experimental Results

5.1 Experiment 1: Convergence of Threshold Adaptation Algorithm

In this experiment, we compare convergence behavior of our proposed bisection method to the method presented in [6]. We use the two methods to generate an adapted threshold map for 77 images and calculate the mean square error (MSE) with respect to the oracle threshold map. To eliminate the randomness in the photon counts, we repeat this experiment 50 times and take the average MSE, then we take the average over the 77 images. The output figures can be generated by running the simulation

```
Experiment1ThresholdMapConvergence.m
```

5.2 Experiment 2: Image Reconstruction Quality

In this experiment, we evaluate the reconstruction quality using different threshold maps. The results are generated by the simulation

```
Experiment2ThresholdMapPerformance.m
```

We use uniform threshold maps with different that have the same quantization value for all pixels. We try different quantization values, which are stored in the following variable

```
QmapU         =  [1 5 10 16];      % Uniform Threshold
```

We also use a sequence of decreasing or increasing threshold as proposed in [7]. Because this method does not tolerate spatial oversampling, we assumed a temporal oversampling factor that is equivalent to the combined temporal-spatial oversampling factor in our simulation. This is shown in the following part of code

```
TCR                          =  K^2*T;         %Temporal Overasampling Factor
sampling_interval_Dur        =  logspace(0,-log10(TCR),TCR);
y0CR=zeros(rows,cols,TCR);
for t=1:TCR
    y0CR(:,:,t)=poissrnd(alpha*c*sampling_interval_Dur(t)/K^2); % Photon Count
end
Q_sequence                   =  fliplr(1:qmax)';
[IM_CR_descQ_descDur,PSNR_CR_desQ_descDur(i,r)]=ConditionalResetAlg(y0CR,TCR,Q_sequence,c);
Q_sequence                   =  (1:qmax)';
[IM_CR_ascQ_descDur,PSNR_CR_ascQ_descDur(i,r)]=ConditionalResetAlg(y0CR,TCR,Q_sequence,c);
```

5.3 Experiment 3: Influence of QIS Threshold on HDR Imaging

In this experiment, we compare the performance of our adapted threshold map to uniform thresholds on HDR imaging. We use the HDR-Eye dataset by Nemoto et al. [8,9]. This dataset contains 9 images acquired at different exposure settings (-2.7,-2, -1.3, -0.7, 0, 0.7, 1.3, 2, and 2.7 EV) to reconstruct one HDR image. For image fusion and tone mapping, we use the HDR toolbox [10] to reconstruct an HDR image. The results can be obtained by running the simulation

```
Experiment3HDRImaging.m
```

6 Installation

The MATLAB package includes all necessary files for QIS simulations. After unzipping the package, the following files will be present:

```
./ImageData/
./Utilities/
demo.m
DrawBisectionFig.m
```

`DrawMotivationFig.m`

`SNRcalculations.m`

`PhaseTransitionSimulations.m`

`Experiment1ThresholdMapConvergence.m`

`Experiment2ThresholdMapPerformance.m`

`Experiment3HDRImaging`

ImageData

This folder includes two datasets

- Our own dataset, which contains 77 images captured by Canon EOS Rebel T6i camera.
- HDREye dataset [8, 9], which must be downloaded from `https://mmspg.epfl.ch/hdr-eye`

Utilities

The utilities folder contains necessary functions to support the main routines. The following 3rd party packages must be installed:

- HDRToolbox: A package for performing image fusion and tone mapping for HDR imaging. It can be downloaded from: `https://www.mathworks.com/matlabcentral/linkexchange/links/2792-the-hdr-toolbox`
- blockfun: A package for executing arbitrary functions on a sliding block over an image. It can be downloaded from `https://www.mathworks.com/matlabcentral/fileexchange/17000-blockfun-applies-function-on-sub-blocks-of-an-array`
- columnlegend: A package for drawing a legend containing many entries efficiently. It can be downloaded from `https://www.mathworks.com/matlabcentral/fileexchange/27389-simonhenin-columnlegend`

7 Copyright

uninterrupted or error-free. The end-user understands that the program was developed for research purposes and is advised not to rely exclusively on the program for any reason.

References

[1] O. A. Elgendy and S. H. Chan, "Optimal threshold design for quanta image sensor," Available online at: http://arxiv.org/abs/1704.03886, Oct. 2017.

[2] E. R. Fossum, "Modeling the performance of single-bit and multi-bit quanta image sensors," *IEEE J. Electron Devices Soc.*, vol. 1, no. 9, pp. 166–174, Sep. 2013.

[3] N. Teranishi, "Required conditions for photon-counting image sensors," *IEEE Trans. Electron Devices*, vol. 59, no. 8, pp. 2199–2205, Aug. 2012.

[4] E. R. Fossum, J. Ma, and S. Masoodian, "Quanta image sensor: concepts and progress," in *Proc. SPIE 9858, Advanced Photon Counting Techniques X*, 2016, vol. 9858, pp. 985804–985804–14.

[5] I. M. Antolovic, S. Burri, C. Bruschini, R. Hoebe, and E. Charbon, "Nonuniformity analysis of a 65k pixel CMOS SPAD imager," *IEEE Trans. Electron Devices*, vol. 63, no. 1, pp. 57–64, Jan. 2016.

[6] C. Hu and Y. M. Lu, "Adaptive time-sequential binary sensing for high dynamic range imaging," in *Proc. SPIE Conf. Adv. Photon Counting*, 2012, vol. 8375, pp. 83750A–1.

[7] T. Vogelsang, D. G. Stork, and M. Guidash, "Hardware validated unified model of multibit temporally and spatially oversampled image sensors with conditional reset," *J. Electronic Imaging*, vol. 23, no. 1, pp. 013021, 2014.

[8] "HDR-eye dataset," `http://mmspg.epfl.ch/hdr-eye`, Accessed: 2016-11-30.

[9] H. Nemoto, P. Korshunov, P. Hanhart, and T. Ebrahimi, "Visual attention in LDR and HDR images," in *9th Int. Workshop on Video Process. and Quality Metrics for Consumer Electronics (VPQM)*, 2015.

[10] T. Mertens, J. Kautz, and F. V. Reeth, "Exposure fusion," in *Proc. 15th Pacific Conf. Comp. Graphics and App., 2007. PG '07.*, Oct 2007, pp. 382–390.