Image Reconstruction in the Gigavision Camera

Feng Yang [1], Luciano Sbaiz [2], Edoardo Charbon [3,4], Sabine Süsstrunk [1] and Martin Vetterli [1,5*]
[1] LCAV - School of Computer and Communication Sciences, [3] AQUA - School of Engineering
Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
[2] Google, Brandschenkestrasse 110, CH-8002 Zürich, Switzerland
[4] TU Delft, Mekelweg 4, 2628 CD, Delft, The Netherlands
[5] Department of Electrical Engineering and Computer Science
UC Berkeley, Berkeley CA94720, USA

Abstract

Recently we have proposed a new image device called the gigavision camera. The main feature of this camera is that the pixels have a binary response. The response function of a gigavision sensor is non-linear and similar to a logarithmic function, which makes the camera suitable for high dynamic range imaging. Since the sensor can detect a single photon, the camera is very sensitive and can be used for night vision and astronomical imaging.

One important aspect of the gigavision camera is how to estimate the light intensity through binary observations. We model the light intensity field as 2D piecewise constant and use Maximum Penalized Likelihood Estimation (MPLE) to recover it. Dynamic programming is used to solve the optimization problem. Due to the complex computation of dynamic programming, greedy algorithm and pruning quadtrees are proposed. They show acceptable reconstruction performance with low computational complexity. Experimental results with synthesized images and real images taken by a single-photon avalanche diode (SPAD) camera are given.

1. Introduction

In conventional cameras, a lens focuses the incident light onto the image sensor. The optical signal is then converted to an electrical signal, which is amplified and A/D converted. The different gray-levels represent different light intensities.

One drawback of a conventional camera is that the dynamic range is limited by the sensor technology and smaller than that of the human eye [8]. To solve this problem, several approaches have been proposed. One way is to combine several pictures with different exposure times [4, 6, 11, 13]. The disadvantages of this method are the increased overall exposure time, inability to capture motion, and extra computational complexity. Another way is to use sensors with logarithmic response [10]. The main shortcoming of this method is an increased fixed pattern noise (FPN) and potentially lower yield.

In [17], we proposed a new image sensor called *gigavision*. This sensor's pixel value is binary and has high spatial resolution, exceeding that in a conventional camera by orders of magnitude. A conventional gray level image can be obtained by low-pass filtering the binary image and sampling, similar to the oversampling techniques applied in A/D converters [2]. The response function of the gigavision sensor is non-linear and similar to a logarithmic function, and thus capable of capturing high dynamic range scenes.

Another drawback of a conventional camera is that the camera is not sensitive at low illumination level. To overcome this, many photon limited imaging systems [19] have been proposed. These imagers have a lot of applications in three dimensional imaging [18], infrared and thermal imaging [12], single photon emission computed tomography (SPECT) [9], confocal microscopy [15] and astronomical imaging [1]. Many algorithms are proposed to estimate the original image from photon-limited images, for example maximum-likelihood estimation [7], bayesian maximum-probable [5], low-pass filtering with improvement by histogram specification [16], and multiscale modeling and estimation [20].

Since the photon counting system with high performance is very expensive, it prevents a lot of applications. In [14], a new single-photon avalanche diode (SPAD) camera based on CMOS technology is proposed, which achieves high speed with low cost but low spatial resolution. Our proposed gigavision camera, which can be implemented using standard memory chip technology is fast and has low cost. Different from the pixel value in a photon counting system,

*The work presented in this paper was supported by the Swiss National Science Foundation under grant number 200021-116742.

2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops
978-1-4244-4441-0/09/$25.00 ©2009 IEEE

in the gigavision camera, the pixel value is binary and there is a threshold T that determines how many photon-electrons are needed to switch the value of the binary pixel from "0" to "1". The statistics of the pixel value in the gigavision camera is different from that in a photon counting system, so new estimation methods on how to retrieve the 2D light intensity field needed to be developed for this new camera. In this paper, the light intensity is modeled as piecewise constant and MPLE is used for reconstruction. Dynamic programming, greedy algorithm, and pruning binary tree or quadtrees are used to find the optimal solution. Dynamic programming can find the optimal solution for 1D signals, but with high complexity, $O(N^3)$ for 1D signals and $O(N^9)$ for 2D images with resolution $N \times N$. Greedy algorithm and Pruning binary tree or quadtrees can achieve a suboptimal solution, but with low complexity. The complexity of greedy algorithm is $O(N^2)$ for 1D signals and is $O(N^3)$ for 2D images with resolution $N \times N$. Pruning binary tree and quadtrees' complexity is $O(N)$ for 1D signals and $O(N^2)$ for 2D images with resolution $N \times N$.

This paper is organized as follows. Section 2 shows the system architecture of a gigavision camera. Section 3 focuses on estimating light intensity from binary observations when the light intensity is 1D piecewise constant. Section 4 deals with the 2D case. Experimental results with synthesized images and the SPAD camera are in Section 5, and the conclusion is in Section 6.

2. The gigavision camera

2.1. Camera architecture

The architecture of a gigavision camera is shown in Figure 1. The incident light is concentrated on the image sensor through a single lens and converted into an electrical signal. Different from a conventional camera, the sensor is binary. A conventional gray level image can be obtained by MPLE or low-pass filtering and sampling.

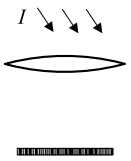


Figure 1. Simplified architecture of a gigavision Camera. The incident light is focused by the lens and then impinges on the image sensor.

2.2. Sensor model

A 1D sensor is considered for simplification. Figure 2 shows the model of a gigavision sensor. The gigavision pixel is similar to a conventional camera pixel except that the quantizers Q' are binary with threshold T. The number of photons impinging on the pixel can be modeled as a Poisson process with intensity $\lambda(x)$, where x is the position parameter. Suppose there are N pixels and neglect the integration for simplicity, then the light intensity for every pixel is $\lambda_i, i = 1 \dots N$, respectively. The response of each binary pixel K_i, $i = 1, \dots, N$ is obtained by comparing the number of arrivals S_i, i.e. the electrons due to detected photons, with the threshold T. The quantities K_i are binary random variables with parameter

$$p_{\lambda_i} = \mathbb{P}[S_i \geq T] = \sum_{k=T}^{\infty} e^{(-\lambda_i)} \frac{(\lambda_i)^k}{k!}, i = 1, \dots, N.$$

Suppose $T = 1$, and $\lambda(x)$ is a constant function, i.e. $\lambda(x) = \lambda$, then if the pixel value $\vec{K} = [K_1, K_2, \cdots, K_{N-1}, K_N]$ and the reconstruction method is maximum likelihood estimation, so $p_\lambda = 1 - e^{-\lambda}$, and

$$\begin{aligned} \hat{\lambda} &= \mathbb{P}(\vec{K}; \lambda) \\ &= \arg\max_{\lambda} (1 - e^{-\lambda})^C (e^{-\lambda})^{N-C} \\ &= \begin{cases} -\ln\left(1 - \frac{C}{N}\right), & C \neq N \\ \lambda_{max}, & C = N \end{cases}, \end{aligned} \quad (1)$$

where C is the number of "1"s in the N pixels, and λ_{max} is equal to $\ln N$, which is used to avoid the estimation value going to ∞.

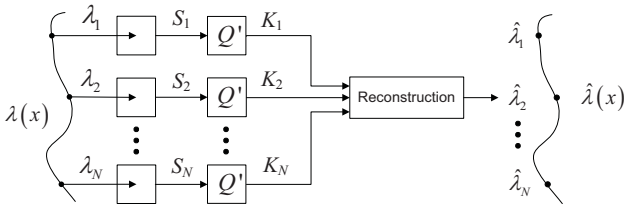


Figure 2. Simplified block diagram of a gigavision sensor. The light intensity function is $\lambda(x)$. There are N pixels in the sensor. The light intensity on each pixel is $\lambda_i, i = 1, \dots, N$. The electrical signal S_i is quantized by a one-bit quantizer with threshold T and an estimation method is implemented to obtain the reconstructed light intensity function $\hat{\lambda}(x)$.

3. Estimating 1D light intensity function through binary observations

We assume that the model of light intensity is 1D piecewise constant, i.e. $\lambda(x)$ is a piecewise constant function. The threshold T is set to "1". Let $\vec{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_{N-1}, \lambda_N]$. The maximum likelihood estima-

tor for this problem is

$$\begin{aligned}\hat{\vec{\lambda}} &= \arg\max_{\vec{\lambda}} \mathbb{P}\left(\vec{K};\vec{\lambda}\right) \\ &= \arg\max_{\vec{\lambda}} \prod_{i=1}^{N}\left((1-k_i)e^{-\lambda_i}+k_i(1-e^{-\lambda_i})\right) \\ &= \arg\max_{\vec{\lambda}} \ln\prod_{i=1}^{N}\left((1-k_i)e^{-\lambda_i}+k_i(1-e^{-\lambda_i})\right) \\ &= \arg\max_{\vec{\lambda}} \sum_{i=1}^{N}\left((1-k_i)e^{-\lambda_i}+k_i(1-e^{-\lambda_i})\right),\end{aligned}$$

where k_i is the realization of the random variable K_i, i.e. the pixel value, $i = 1, 2, \cdots, N$. If there is no constraint on λ_i, then the optimal solution is $\lambda_i = 0$ when $k_i = 0$, and $\lambda_i = \lambda_{max}$ when $k_i = 1$. The problem is that the estimation variance is high when exactly one point is used for estimating λ_i. We can gain from using neighboring pixels to estimate λ_i, since natural scenes have some structure. Here, we assume the light intensity function $\lambda(x)$ to be piecewise constant. If the position of each segment of the function is known in advance, then the pixels in this region can be used to estimate the light intensity value. The estimation variance and bias will be smaller in this case. If too many pixels are used for estimating one segment, for example pixels not belonging to the segment are used, then the estimation bias will be larger. If the number of segments of the light intensity function is known, an algorithm that determines the segment position can be designed. Unfortunately, usually we do not know the number of segments. Hence, a tunable penalization term $-\gamma\#P$ is added to the likelihood function, where P is a set containing all the segments from a segmentation of the 1D light intensity function, $\#P$ is the number of segments and γ is a parameter that can be set according to different scenes. Large γ means that the scene has few segments. The MPLE for this problem is

$$\hat{\vec{\lambda}} = \arg\max_{\vec{\lambda}} \sum_{i=1}^{N}\left((1-k_i)e^{-\lambda_i}+k_i(1-e^{-\lambda_i})\right) - \gamma\#P.$$

In the estimation, the pixels are divided into $\#P$ segments, the light intensity function in each segment is assumed to be constant. Let N_j be the number of pixels in jth segment, C_j is the number of "1"s in this segment, $j = 1, 2, \cdots, \#P$. $\hat{\lambda}$ for this segment can be obtained using equation (1). The MPLE can be written in another form

$$\begin{aligned}\hat{P} = \arg\max_{P} &\sum_{j=1}^{\#P}\sum_{i=1}^{N_j}\left((1-k_{j,i})(1-\tfrac{C_j}{N_j})+k_{j,i}\tfrac{C_j}{N_j})\right) \\ &-\gamma\#P \qquad s.t. \quad 1 \le \#P \le N \\ \hat{P} \Rightarrow \hat{\vec{\lambda}},\end{aligned}$$

where $k_{j,i}$ is the value of ith pixel of jth segment.

Dynamic programming, greedy algorithm and pruning binary tree are proposed to solve this optimization problem.

3.1. Dynamic programming

Let $Cost(i,t), 1 \le i \le t \le N$ be the cost when there is only one segment in the region $[i,t]$ and $F(j,i), j \le i, 1 \le i \le N$ be the maximum total cost when there are j segments in $[1,i]$. $Cost(i,t)$ is equal to

$$\sum_{s=1}^{t-i+1}(1-k_{i-1+s})(1-\frac{C_{it}}{t-i+1})+k_{i-1+s}\frac{C_{it}}{t-i+1})-\gamma,$$

where C_{it} is the number of "1"s in the segment $[i,t]$ and k_i is the binary pixel value at position i. Then, $F(j,i), j \le i, 1 \le i \le N$ is computed using the following iteration equations.

$$\begin{cases} F(1,i) = Cost(1,i), & 1 \le i \le N \\ F(j,i) = \max\limits_{2\le s\le i}\{F(j-1,s-1)+Cost(s,i)\}, & \\ \qquad\qquad 2 \le j \le i, 1 \le i \le N & \end{cases}$$

The optimal segmentation

$$P_{optimal} = \arg\max_{P} F(\#P, N), s.t. 1 \le \#P \le N.$$

According to all the segments in $P_{optimal}$, $\hat{\vec{\lambda}}$ is computed using equation (1). The complexity of this algorithm is $O(N^3)$.

3.2. Greedy algorithm

Since the dynamic programming's complexity is $O\left(N^3\right)$, a simple greedy algorithm can be employed to increase the speed. Let $Cost(i,t), 1 \le i \le t \le N$ be the cost value for the segment $[i,t]$. The same cost function $Cost(i,t)$ as in dynamic programming is used here. In the greedy algorithm, the total cost is first set to be $Cost(1,N)$. Then, we decide whether to divide the segment $[1,N]$ into two segments $[1,s-1]$ and $[s,N]$, where

$$s = \arg\max_{2\le i\le N}\{Cost(1,i-1)+Cost(i,N)\}.$$

If $Cost(1,N) < Cost(1,s-1)+Cost(s,N)$, we make the division, otherwise not. If the segment is cut into two segments, then we consider if the two segments can still be cut to gain in the total cost. This is done iteratively until no segment can be cut. The procedure gives sub-optimal solution. $\hat{\vec{\lambda}}$ can be computed as the method in dynamic programming. The complexity for this algorithm is $O(N^2)$, which is faster than dynamic programming. The pseudocode for this algorithm is shown in Table 1.

Initialize: Cut = $\{[1, N]\}$, Not_Cut = *null*,
Cut is the set which contains the segments to be divided,
Not_Cut is the segments which can not be divided.
Loop:
while #Cut $\neq 0$, #Cut is the number of elements in Cut
  for i = 1 to #Cut
    suppose Cut{i} = [m,n]
    $s = \arg\max_{m+1 \leq t \leq n} Cost(m, t-1) + Cost(t, n)$
    if $Cost(m, s-1) + Cost(s, n) > Cost(m, n)$
      put [m,s-1] and [s,n] into a set Cut_temp
      delete [m,n] from set Cut
    else
      put [m,n] into the set Not_cut
    end if
  end for i
  Cut = Cut_temp
end while
Estimate:
$\hat{\vec{\lambda}}$ can be computed similar to that in dynamic programming according to the segments in Not_cut.

Table 1. Pseudocode for 1D greedy algorithm

Initialize:
Calculate $Cost\_value(S_{j,i})$
Loop:
for j = $\log(N)$-1 to 0
  for i = 1:2^j
    if child nodes of $S_{j,i}$ has no child node that is unpruned
      if $Cost\_value(S_{j,i}) > Cost\_value(S_{j+1,2i-1}) + Cost\_value(S_{j+1,2i})$
        prune the two child nodes
      end if
    end if
  end for i
end for j
Estimate:
$\hat{\vec{\lambda}}$ can be computed similar to that in dynamic programming according to the segmentation denoted by the leaf nodes.

Table 2. Pseudocode for pruning binary tree algorithm

3.3. Pruning binary tree

To further reduce complexity, a binary tree is first constructed. The tree is denoted as $S_{j,i}, 0 \leq j \leq \log(N), i = 1, \cdots, 2^j$. Each node $S_{j,i}$ means that the light intensity $\lambda_s, s \in [(i-1)2^{(\log(N)-j)} + 1, i \times 2^{(\log(N)-j)}]$ is constant. We remark that this choice restricts the number of allowed partitions with respect to the previous models. The cost function $Cost(i, t)$ is the same as in the greedy algorithm. Each node has a cost value $Cost\_value(S_{j,i}) = Cost\left((i-1)2^{(\log(N)-j)} + 1, i \times 2^{(\log(N)-j)}\right)$. We prune this tree from the leaf nodes. If the cost value of the parent node is larger than the summation of two child nodes, then we prune two child nodes. If the parent node has a child node that has unpruned child nodes, then the parent will not be considered for pruning. The pruning process is implemented iteratively until no node can be pruned. All the leaf nodes in the pruned tree denote the segmentation of the N pixels. $\hat{\vec{\lambda}}$ can be computed as in dynamic programming, according to this segmentation. The time complexity of this algorithm is $O(N)$. The pseudocode for this algorithm is shown in Table 2.

4. Estimating 2D light intensity through binary observations

We assume that the model of the light intensity function $\lambda(x, y)$ is 2D piecewise constant, i.e. the definition region of $\lambda(x, y)$ can be divided into a lot of blocks with different size, in which $\lambda(x, y)$ is constant. The threshold T of the gigavision camera is set to "1". Let $K = \{k_{ij}\}_{N \times N}$ be the pixel value captured by the gigavision camera and

$$M_\lambda = \begin{pmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1N} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{N1} & \lambda_{N2} & \cdots & \lambda_{NN} \end{pmatrix}$$

be the light intensity on each pixel. Then the maximum likelihood estimator for this problem is

$$\begin{aligned} M_{\hat{\lambda}} &= \arg\max_{M_\lambda} \mathbb{P}\left(K; M_\lambda\right) \\ &= \arg\max_{M_\lambda} \prod_{i=1}^{N} \prod_{j=1}^{N} \left((1-k_{ij})e^{-\lambda_{ij}} + k_{ij}(1-e^{-\lambda_{ij}})\right) \\ &= \arg\max_{M_\lambda} \ln \prod_{i=1}^{N} \prod_{j=1}^{N} \left((1-k_{ij})e^{-\lambda_{ij}} + k_{ij}(1-e^{-\lambda_{ij}})\right) \\ &= \arg\max_{M_\lambda} \sum_{i=1}^{N} \sum_{j=1}^{N} \left((1-k_{ij})e^{-\lambda_{ij}} + k_{ij}(1-e^{-\lambda_{ij}})\right), \end{aligned}$$

If there is no constraint on λ_{ij}, then the optimal solution is $\lambda_{ij} = 0$ when $k_{ij} = 0$, and $\lambda_{ij} = \lambda_{max}$ when $k_{ij} = 1$. As with the 1D case, the problem is that the estimation variance is high when only one point is used for estimating λ_{ij}. We can gain from using neighboring pixels to estimate λ_{ij}, since natural scenes have some structures. Here, we assume the light intensity function $\lambda(x, y)$ is 2D piecewise constant. Similar to the 1D case, the maximum penalized

likelihood estimator for this problem is

$$M_{\hat{\lambda}} = \arg\max_{M_\lambda} \sum_{i=1}^{N}\sum_{j=1}^{N}\left((1-k_{ij})e^{-\lambda_{ij}} + k_{ij}(1-e^{-\lambda_{ij}})\right) \\ -\gamma \#P,$$

where P is a set containing all the blocks from a segmentation of the 2D light intensity function, $\#P$ is the number of the blocks and γ is a parameter that can be set according to different scenes. In the estimation, the pixels are divided into $\#P$ blocks and the light intensity function in each block is assumed to be constant. Let $N_{jm} \times N_{jn}$ is the size of jth block, C_j is the number of "1"s in this block, $j = 1, 2, \cdots, \#P$. $\hat{\lambda}$ for this block can be obtained using equation (1). The MPLE can be written in another form

$$\hat{P} = \arg\max_{P} \\ \sum_{j=1}^{\#P}\sum_{m=1}^{N_{jm}}\sum_{n=1}^{N_{jn}}\left((1-k_{j,mn})(1-\tfrac{C_j}{N_{jm}N_{jn}}) + k_{j,mn}\tfrac{C_j}{N_{jm}N_{jn}}\right) \\ -\gamma \#P \qquad s.t. \quad 1 \le \#P \le N \\ \hat{P} \Rightarrow M_{\hat{\lambda}},$$

where $k_{j,mn}$ is the pixel value at position (m, n) of jth block.

Dynamic programming, greedy algorithm and pruning quadtrees are proposed to solve this optimization problem.

4.1. Dynamic programming

Let $Cost(s,t,m,n), 1 \le s \le m \le N, 1 \le t \le n \le N$ be the cost when there is only one block in the region $[s,m] \times [t,n]$ and $F(j,s,t,m,n), 1 \le s \le m \le N, 1 \le t \le n \le N$ is the maximum total cost when there are j blocks in the $[s,m] \times [t,n]$. $Cost(s,t,m,n)$ is equal to

$$\sum_{a=s}^{m}\sum_{b=t}^{n}(1-k_{ab})(1-\tfrac{C}{(m-s+1)(n-t+1)}) \\ +k_{ab}\tfrac{C}{(m-s+1)(n-t+1)}) - \gamma,$$

where C is the number of "1"s in the block, k_{ab} is the binary pixel value at position (a, b). Then, $F(j,s,t,m,n), 1 \le s \le m \le N, 1 \le t \le n \le N$ is computed using the following iteration equations.

$$\begin{cases} F(1,s,t,m,n) = Cost(s,t,m,n), \\ \quad 1 \le s \le m \le N, 1 \le t \le n \le N \\ F(j,s,t,m,n) = \max\{h\_cut, v\_cut\}, \\ \quad 2 \le j \le (m-s+1)\times(n-t+1), \\ \quad 1 \le s \le m \le N, 1 \le t \le n \le N \\ h\_cut = \max\limits_{s+1\le w\le m}\max\limits_{1\le a\le j-1}\{F(a,s,t,w-1,n) \\ \quad +F(j-a,w,t,m,n)\}, \\ v\_cut = \max\limits_{t+1\le v\le n}\max\limits_{1\le a\le j-1}\{F(a,s,t,m,v-1) \\ \quad +F(j-a,s,v,m,n)\}, \end{cases}$$

The optimal segmentation is

$$P_{optimal} = \arg\max_{P} F(\#P, 1, 1, N, N), s.t. 1 \le \#P \le N.$$

According to all the blocks in $P_{optimal}$, $M_{\hat{\lambda}}$ is computed using equation (1). The time complexity of this algorithm is $O(N^9)$.

4.2. Greedy algorithm

The same cost function $Cost(s,t,m,n)$ as in dynamic programming is used here. In the greedy algorithm, the total cost is first set to $Cost(1,1,N,N)$. After that, we decide whether to divide the block $[1,N] \times [1,N]$ into two blocks $[1,N] \times [1,s-1]$ and $[1,N] \times [s,N]$ or $[1,t-1] \times [1,N]$ and $[t,N] \times [1,N]$,

$$s = \arg\max_{2\le i\le N}\{Cost(1,1,N,i-1) + Cost(1,i,N,N)\}. \\ t = \arg\max_{2\le i\le N}\{Cost(1,1,i-1,N) + Cost(i,1,N,N)\}.$$

If $Cost(1,1,N,N) < \max\{Cost(1,1,N,s-1) + Cost(1,s,N,N), Cost(1,1,t-1,N) + Cost(t,1,N,N)\}$, we make the division, otherwise not. If the block is cut into two blocks, then we consider whether the two blocks can still be cut to gain in the total cost. This is done iteratively, until a sub-optimal solution is reached. $M_{\hat{\lambda}}$ can be computed similarly as in dynamic programming. The complexity of this algorithm is $O(N^3)$.

4.3. Pruning quadtrees

A quadtrees is constructed first. The tree is denoted as $S_{j,m,n}, 0 \le j \le \log_4 N, m = 1, \cdots, 4^j, n = 1, \cdots, 4^j$. Each node $S_{j,m,n}$ means that the light intensity $\lambda_{ab}, (a,b) \in [(m-1)4^{(\log_4 N - j)} + 1, m \times 4^{(\log_4 N)-j}] \times [(n-1)4^{(\log_4 N - j)} + 1, n \times 4^{(\log_4 N)-j}]$ is constant. The cost function $Cost(s,t,m,n)$ is the same as in greedy algorithm. Each node has a cost value

$$Cost\_value(S_{j,m,n}) = \\ Cost((m-1)4^{(\log_4 N-j)} + 1, (n-1)4^{(\log_4 N-j)} + 1, \\ m \times 4^{(\log_4 N-j)}, n \times 4^{(\log_4 N-j)}).$$

We prune this tree from the leaf nodes. If the cost value of the parent node is larger than the summation of four child nodes, we prune all the four child nodes. If the parent node has a child node that has at least an unpruned child node, the parent node will not be considered for pruning. The pruning process is implemented iteratively until no node can be pruned. All the leaf nodes in the pruned tree denote the segmentation of the $N \times N$ pixels. $M_{\hat{\lambda}}$ is computed according to equation (1). The complexity of this algorithm is $O(N^2)$.

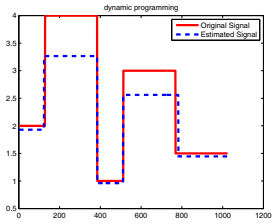


Figure 3. Estimation result using dynamic programming for the optimization. The threshold of the gigavision camera is $T = 1$, the number of pixels is $N = 1024$ and parameter $\gamma = 5$.

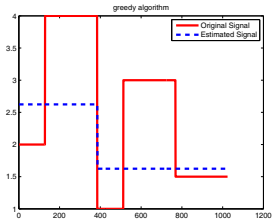


Figure 4. Estimation result using greedy algorithm for the optimization. The threshold of the gigavision camera is $T = 1$, the number of pixels is $N = 1024$ and parameter $\gamma = 5$.

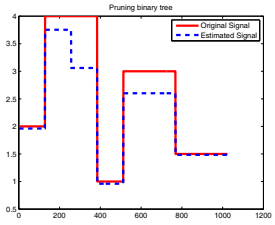


Figure 5. Estimation result using the pruning binary tree for the optimization. The threshold of the gigavision camera is $T = 1$, the number of pixels is $N = 1024$ and parameter $\gamma = 5$.

5. Experiments

5.1. 1D synthesized signal

A 1D piecewise constant signal is generated. A gigavision camera with threshold $T = 1$ is simulated to take images of this signal and the number of pixels N is 1024. We chose the parameter γ for MPLE to be 5 based on experiments. The results are shown in Figure 3, Figure 4, and Figure 5. The mean square error (MSE) for dynamic programming, greedy algorithm, and pruning binary tree are $0.21, 1.0504$, and 0.1583, respectively. The MSE of dynamic programming is larger than the pruning binary tree algorithm. The reason is that although the penalization term in the objective function for the optimization has some influence on the reconstruction error, there is no one-to-one mapping. Even if dynamic programming gets the optimal solution for the objective function, the MSE may be bigger than that of pruning binary tree algorithm. The performance of the two algorithms also depends on γ. We need to choose this parameter based on our knowledge of the original signal. If the signal has many segments, γ is small, otherwise large.

5.2. 2D synthesized image

In this section, we simulate the acquisition of an image using the gigavision sensor. The diagram of this experiment is shown in Figure 6. The input image is 'building.bmp' with resolution 512×512. Each image pixel has a gray level in the range $[0, 255]$. In the gigavision camera, the sampling frequency is higher than the bandwidth of the scene. To simulate this, the original image is low-pass filtered by a Gaussian filter with size 20×20 and $\sigma = 20$. The bandwidth of this filter is about $\frac{\pi}{8}$. The gray level is then scaled to $[0, 5]$. We assume that the gray level corresponds to the light intensity, i.e. setting λ equal to the scaled gray level. For each pixel, we generate the random number of detected photons according to the Poisson distribution of parameter λ and we simulate the behavior of the gigavision sensor. An approximation of the intensity M_λ is estimated using the algorithm above with $\gamma = 2$. The final image is obtained through low-pass filtering the estimated intensity, scaling to $[0, 255]$ and downsampling. The low-pass filter here is the same as the previous low-pass filter. The downsampling factor is chosen according to the bandwidth of the low-pass filter. If the low-pass filter is an ideal low-pass filter with bandwidth B, the downsampling factor should be $\frac{2\pi}{B}$. But as the Gaussian filter used here is not ideal, we use a smaller downsampling factor 8 instead of 16 to reduce aliasing. The resolution of the reconstructed image is 64×64. Since dynamic programming for the 2D case is too complex, only greedy algorithm and pruning quadtrees are used.

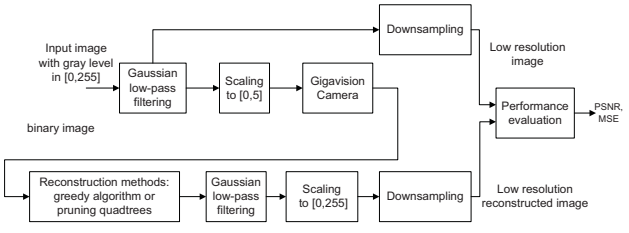
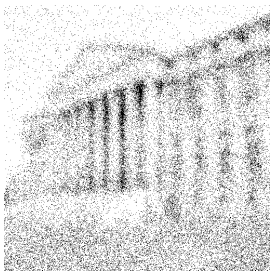


Figure 6. The experimental diagram

Figure 7(a) and (b) show the original image and the binary image captured by the gigavision camera. Figure 7(c) shows the downsampled image of the original low-pass filtered image with resolution 64×64. Figure 7(d) shows the reconstructed image using the greedy algorithm. Figure 7(e) shows the reconstructed image using pruning quadtrees. The PSNR for the greedy algorithm is 28.79dB and 28.14dB for pruning quadtrees.

We then apply a Gaussian low-pass filter with size 40×40, $\sigma = 20$ and bandwidth $\frac{\pi}{16}$ is used. The downsampling factor is 16. Figure 8(a) and (b) show the low-pass filtered

(a) Original image



(b) Binary image

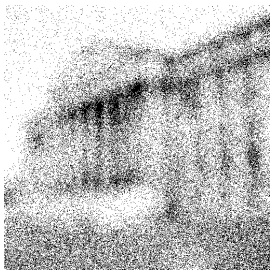
(c) Original downsampled

(d) Greedy

(e) Quadtrees

Figure 7. Simulated images for the gigavision sensor. The image 'building.bmp' with resolution 512×512 is used to simulate the number of photons at each pixel and MPLE is used for reconstruction. The parameter $\gamma = 2$. The Gaussian low-pass filtering's size is 20×20 and $\sigma = 20$. The downsampling factor is 8. The resolution of the reconstructed image is 64×64.

(a) Low-pass image



(b) Binary image

(c) Original downsampled

(d) Greedy

(e) Quadtrees

Figure 8. Simulated images for the gigavision sensor. The image 'building.bmp' with resolution 512×512 is used to simulate the number of photons at each pixel and MPLE is used for reconstruction. The parameter $\gamma = 2$. The Gaussian low-pass filtering's size is 40×40 and $\sigma = 20$. The downsampling factor is 16. The resolution of the reconstructed image is 32×32.

original image and the binary image captured by the gigavision camera. Figure 8(c) shows the downsampled image of the original low-pass filtered image with resolution 32×32. Figure 8(d) shows the reconstructed image using the greedy algorithm. Figure 8(e) shows the reconstructed image using pruning quadtrees. The PSNR for the greedy algorithm is 32.58dB and 32.50dB for pruning quadtrees. Note that with a higher downsampling factor, i.e., oversampling factor, better performance can be achieved.

5.3. Images taken by SPAD camera

We additionally do some experiments with SPAD camera [3], shown in Figure 9. The resolution of the SPAD camera is 32×32 pixels. The pixel value of the image taken by the SPAD camera is "1"(at least one photon hitting the pixel) or "0"(no photon hitting the pixel). The pixel value is the same as in the gigavision camera except that the spatial resolution is much smaller.

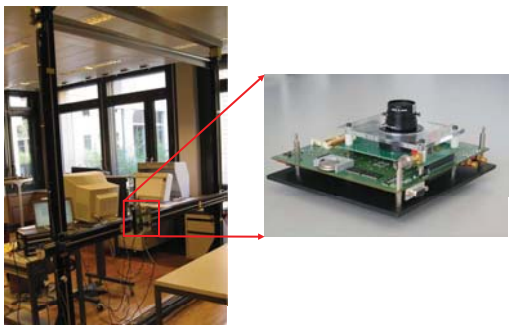


Figure 9. A SPAD camera with resolution 32×32 is fixed on a 2D positioning system.

(a) Binary image

(b) Estimated image

Figure 10. a) is one of the binary images taken by the SPAD camera. b) is the estimated images with 256 binary images.

In the first experiment, the position of the object and the SPAD camera are fixed and a set of pictures are taken. The time for capturing each binary image is about $4\mu s$. Since the position is fixed, the light intensity value is constant and we can estimate the light intensity using equation (1). We use 256 binary images. One of them and the estimated light intensity are shown in Figure 10(a) and (b). Although the object is not recognizable in the binary image, it becomes visible in the reconstructed image.

To get a larger resolution binary image with the SPAD camera, we do a second experiment with a 2D positioning system (Figure 9). The SPAD camera is fixed to the 2D positioning system and can be moved in the 2D plane. The camera is moved to 32×32 positions and 1024 images are taken. By stitching 1024 binary images, we get a binary image with resolution 1024×1024. The same reconstruction algorithm as for 2D synthesized images is implemented for the binary image. The Gaussian low-pass filter's size is 20×20 and $\sigma = 20$. The downsampling factor is 8. Thus, the resolution of the reconstructed image is 128×128. The large resolution image generated by stitching 1024 binary images taken by the SPAD camera is shown in Figure 11.

The greedy and the quadtrees algorithm are used to reconstruct the image. Since in the reconstructed images most of the pixels have values lower than 150, the image appears dark. We change the pixel values by rescaling the values below 150 in the range $[0, 255]$ and saturating pixels with value larger than 150. The reconstructed images are shown in Figure 12(a) and (b), respectively.

Figure 11. The binary image generated by stitching 1024 binary images taken by the SPAD camera.

(a) Greedy algorithm (b) Quadtrees

Figure 12. Reconstructed images.

6. Conclusion

In this paper, we model the light intensity as piecewise constant and propose MPLE for reconstructing original light intensity from the binary images taken by the gigavision camera. Dynamic programming is used to solve the optimization problem. To increase the speed of reconstruction, two other methods, greedy algorithm and pruning binary tree or quadtrees are also proposed. Experiments on synthesized images show the good performance of our method. We additionally perform experiments with a SPAD camera, which can detect single photons, but with low spatial resolution. Future work will focus on using more complex models like piecewise linear or piecewise smooth models to enhance the estimation performance and design a real gigavision sensor.

References

[1] P. B. Boyce. Low light level detectors for astronomy. *Science*, 198(4313):145–148, October 1977.

[2] J. C. Candy and G. C. Temes. *Oversampling Delta-Sigma Data Converters — Theory, Design and Simulation*. IEEE Press, New York, 1992.

[3] L. Carrara, C. Niclass, N. Scheidegger, H. Shea, and E. Charbon. A gamma, x-ray and high energy proton radiation-tolerant cmos image sensor for space applications. In *IEEE International Solid-State Circuits Conference*, pages 40–41, 2009.

[4] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proc. of SIGGRAPH*, pages 369–378, 1997.

[5] B. R. Frieden. Maximum-probable restoration of photon-limited images. *Applied Optics*, 26(9):1755–1764, May 1987.

[6] M. D. Grossberg and S. K. Nayar. High dynamic range from multiple images: Which exposures to combine? In *ICCV Workshop on Color and Photometric Methods in Computer Vision*, 2003.

[7] M. Guillaume, P. Melon, and P. Réfrégier. Maximum-likelihood estimation of an astronomical image from a sequence at low photon levels. *J. Opt. Soc. Am. A*, 14(11):2841–2848, November 1998.

[8] B. Hoefflinger. *High-Dynamic-Range Vision — Microelectronics, Image Processing, Computer Graphics*. Springer, Berlin Heidelberg, 1998.

[9] R. J. Jaszczak, R. E. Coleman, and C. B. Lim. Spect: single photon emission computed tomography. *IEEE Transactions on Nuclear Science*, 27:1137–1153, June 1980.

[10] S. Kavadias, B. Dierickx, D. Scheffer, A. Alaerts, D. Uwaerts, and J. Bogaerts. A logarithmic response CMOS image sensor with on-chip calibration. *IEEE Journal of Solid-State Circuits*, 35(8):1146–1152, August 2000.

[11] S. Kavusi, K. Ghosh, and A. E. Gamal. Architectures for high dynamic range, high speed image sensor readout circuits. In *International Federation for Information Processing, VLSI-SoC*, 2006.

[12] S. Komiyama, O. Astafiev, V. Antonov, T. Kutsuwa, and H. Hirai. A single-photon detector in the far-infrared range. *Nature*, 403(27):405–407, January 2000.

[13] L. Meylan and S. Süsstrunk. High dynamic range image rendering using a retinex-based adaptive filter. *IEEE Transactions on Image Processing*, 15(9):2820–2830, September 2006.

[14] C. Niclass, A. Rochas, P. Besse, and E. Charbon. Design and characterization of a cmos 3-d image sensor based on single photon avalanche diodes. *IEEE Journal of Solid-State Circuits*, 47(9):1847–1854, Sep. 2005.

[15] J. B. Pawley. *Handbook of biological confocal microscopy*. Springer, New York, 2006.

[16] P. Prieto and M. P. Cagigal. Low-pass filtering applied to photon-limited images: improvement by histogram specification. *Pure Appl. Opt.*, 4:79–87, 1995.

[17] L. Sbaiz, F. Yang, E. Charbon, S. Süsstrunk, and M. Vetterli. The gigavision camera. In *IEEE Conference on Acoustics, Speech and Signal Processing*, 2009.

[18] B. Tavakoli, B. Javidi, and E. Watson. Three dimensional visualization by photon counting computational integral imaging. *Optics Express*, 16(7):4426–4436, March 2008.

[19] Y. Tsuchiya, E. inuzuka, T. kurono, and M. Hosoda. Photon-counting imaging and its application. *Advances in electronics and electron physics*, 64, 1985.

[20] R. M. Willet and R. D. Nowak. Platelets: a multiscale approach for recovering edges and surfaces in photon-limited medical imaging. *IEEE Transactions on Medical Imaging*, 22(3):332–350, March 2003.