Published in final edited form as:

Proc SPIE Int Soc Opt Eng. 2007 August 26; 6707: . doi:10.1117/12.740321.

Fast maximum-likelihood estimation methods for scintillation cameras and other optical sensors

L.R. Furenlid^{1,2}, J.Y. Hesterman², and H.H. Barrett^{1,2}

¹Department of Radiology, University of Arizona Tucson, AZ 85724

²College of Optical Sciences, University of Arizona Tucson, AZ 85724

Abstract

Maximum-likelihood estimation methods offer many advantages for processing experimental data to extract information, especially when combined with carefully measured calibration data. There are many tasks relevant to x-ray and gamma-ray detection that can be addressed with a new, fast ML-search algorithm that can be implemented in hardware or software. Example applications include gamma-ray event position, energy, and timing estimation, as well as general applications in optical testing and wave-front sensing.

Keywords

maximum-likelihood; search algorithm; gamma camera

1. INTRODUCTION

Maximum-likelihood estimation (MLE) methods are powerful tools for extracting information from experimental data, especially when combined with carefully measured calibration data.¹ In this manuscript, we survey several estimation tasks relevant to x-ray and gamma-ray detection with a variety of sensors, and describe fast ML-estimation algorithm techniques and how they can be applied to derive position, energy, and timing information.

MLE is generally considered a post-processing step, but new acquisition architectures that pass raw detector signals to in-line dedicated MLE hardware can be contemplated for performing estimates in real time.² This is especially true if algorithms can be implemented on computing hardware that have resources for parallel and/or vector processing that can be efficiently utilized. Depending on the complexity of the required calculations, CPUs, DSPs, FPGAs, and combinations thereof are all possible choices for performing the estimation task.

A fundamental requirement to be able to perform maximum-likelihood estimation is the formulation of a quantitative expression for the event likelihood, the conditional probability of observing the data vector $\mathbf{g}$ given particular values for the set of parameters $\boldsymbol{\theta}$ that are being estimated:

$$\hat{\theta} = \arg \max_{\theta} pr(\mathbf{g} | \theta) \quad (1)$$

The probability, which can be a probability density in the case of a continuous variable or a probability in the case of a discrete variable, can be derived from a physics model of the signal-generation process, calibration measurements, or a combination of the two. In practice, it is always preferable to use a combination of measurements and theory, to derive a functional form for the probability expression from an understanding of the underlying physics and make careful measurements to determine constants such as means, gains, and variances.

If the data points are independent, the probability law may be factored as a product of the individual probabilities of the elements of the data vector,

$$pr(\mathbf{g} | \theta) = \prod_{i=1,m} pr(g_i | \theta). \quad (2)$$

It is possible to simplify the expression by taking the natural logarithm of both sides, a monotonic transformation that leaves the result of the *arg max* function unchanged. That is,

$$\hat{\theta} = \arg \max_{\theta} pr(\mathbf{g} | \theta) = \arg \max_{\theta} \ln[pr(\mathbf{g} | \theta)] = \arg \max_{\theta} \sum_{i=1,m} \ln[pr(g_i | \theta)]. \quad (3)$$

Application of the natural logarithm also has the benefit of expanding the dynamic range of probabilities that can be represented in a digital word of fixed length, albeit at some expense in precision. The probability of measuring a specific, long vector of data values can be very small simply as a result of the existence of a large number of combinations of signals.

The advantages of MLE over other methods, including *ad hoc* curve fits and centroiding methods, are that 1) the statistics of the signal formation process are explicitly considered; 2) the method is asymptotically unbiased, i.e. its mean converges ever closer to the right answer as more independent data is added; 3) the estimate asymptotically achieves the Cramér-Rao lower bound on variance for unbiased estimators; and 4) a figure of merit, the log-likelihood value, is computed that can be used for rejecting observations with unlikely combinations of signals. In a scintillation camera, for example, application of a threshold on event likelihood helps reject events with a partial or multi-centered deposition of energy, background radiation at different photon energies, pulses corrupted by pileup, etc...

There are two principal obstacles that must be overcome in order to implement a maximum-likelihood estimation approach. As introduced above, an expression for the log likelihood must be determined, which may involve calibration measurement. And a fast method of searching the parameter space must be devised. This must often begin with a definition of the physical reasonable parameter space that needs to be considered and how finely it will be meshed.

2. FAST MAXIMUM-LIKELIHOOD ESTIMATION ALGORITHMS

2.1 Look-up Tables

One approach that has been employed for real-time MLE is look-up tables, in which the data vector is concatenated to form a single digital word that is used as an address that points to a pre-computed set of parameter estimates. This is easily the fastest method possible to implement in hardware; once a look-up table has been loaded into memory, processing a new event requires only a few memory access cycles. Processing speeds in excess of 10^8 events per second can be achieved. However, the total length of the data vector cannot be much larger than 32 bits to keep the dimensions of the addressed space from being impractical.

If the look-up table is sparse, such as might occur if many or most possible signal combinations do not correspond to physically possible event data, a hash algorithm can be used to reduce the size of the look-up table. A simple hash function essentially reorders the bits of the data vector in order to collect the regions of the look-up table that contain valid estimates into as contiguous a space as possible. More complicated hash functions apply linear or non-linear operators that may require enough processing time to affect throughput. With either approach, look-up-table sizes can easily be reduced by one or two orders of magnitude, but a fundamental restriction on total data-vector length still applies.

Look-up tables can play an important role as fast components of other MLE algorithms, for example carrying out trigonometric, logarithmic, or factorial functions. They are used to advantage in the fast search algorithm described next.

2.2 A Fast Search Algorithm

Many algorithms have been developed that can be used for MLE.³ Most require implementation in conventional CPUs and offer little opportunity for speedup with vector or parallel processing. We have recently found a convenient algorithm for carrying out fast maximum-likelihood estimation that is suitable for implementation in a variety of computing hardware, including CPUs, clusters of CPUs, GPUs, DSPs, and FPGAs. The new method can be parallelized and pipelined to achieve very high throughput. The steps can be summarized as follows:

1. Pad the region around the physically reasonable parameter space with an area (volume or hypervolume depending on number of parameters and thereby dimensionality of search) that yields vanishingly small likelihood regardless of data vector values. This allows test locations in step 3a to probe outside the allowable domain of estimates without possibility of being returned as the highest likelihood *without the use of conditional tests* ("if statements" in software).
2. Pre-compute all terms in the likelihood expression that depend only on calibration data and scale appropriately with powers of two to permit use of integer multiplications and sums
3. For $i = 1$ to the number of iterations required to resolve the estimates to the desired precision:

- a. Compute the log likelihood at test locations defined as the centers of squares (cubes or hypercubes) that cover the parameter region under test
- b. Select the location with the highest log likelihood as the center of the next iteration's region of test
- c. Decrease the size of the region of test by a contraction factor in each dimension.

As will be seen in the example application below, the value of the likelihood at test points can be computed in parallel and each iteration can be implemented as one step in a pipeline. We illustrate application of the algorithm with a simple x and y position-estimation problem for a modular gamma camera.

2.3 Application in a Modular Scintillation Camera

A modular scintillation camera (ModCAM) combines a monolithic scintillator crystal, a fused-quartz light guide, and an array of photomultiplier tubes (PMTs) to form a compact gamma-ray detector.⁴ The output signal at each PMT anode is proportional to number of primary photoelectrons produced at the photocathode. The number of primary photoelectrons generated obeys Poisson statistics for a model including three-dimensional interaction location and energy, and the signals in different PMTs are independent. In a camera with a relatively thin scintillator, a simplification to a two-dimensional position estimation problem with fixed energy allows use of a scaled-Poisson model. Then the probability of a given data vector resulting from a gamma-ray event at a location $\mathbf{r}$ on the camera face is characterized by the mean signals on the PMTs arising from events at $\mathbf{r}$. This is a fundamental property of the Poisson distribution, and the probability law on a data vector of length m is

$$pr(\mathbf{g}|\mathbf{r}) = pr(\mathbf{g}|\bar{\mathbf{g}}(\mathbf{r})) = \prod_{i=1}^m \frac{e^{-\bar{g}_i(\mathbf{r})} \bar{g}_i(\mathbf{r})^{g_i}}{g_i!}. \quad (4)$$

For a 9-PMT modular camera, the expression for log likelihood simplifies to a two term expression: a 9-element sum of the products of observed PMT signals with the natural logarithm of their corresponding mean signals and a 9-element sum of mean signals:

$$\hat{\mathbf{r}} = \arg \max_{\mathbf{r}} \ln \text{prob}(\mathbf{g}|\mathbf{r}) = \arg \max_{\mathbf{r}} \sum_{i=1}^9 [g_i \ln \bar{g}_i(\mathbf{r}) - \bar{g}_i(\mathbf{r}) - \ln g_i!] = \arg \max_{\mathbf{r}} \left[\sum_{i=1}^9 g_i \ln \bar{g}_i(\mathbf{r}) - \sum_{i=1}^9 \bar{g}_i(\mathbf{r}) \right]. \quad (5)$$

The arg-max function eliminates the factorial terms, which do not depend on the search variable $\mathbf{r}$. However, if a final log-likelihood value is needed for thresholding, a sum-over-log-factorials term must be subtracted off after the search is completed.

The mean PMT signals as a function of event location are easily measured with a calibration procedure that scans a collimated gamma-ray source across the detector face as shown in Figure 1a below.⁵ The resulting data are shown in the 9 intensity plots in Figure 1b. Each intensity plot represents the mean signal for a PMT, in A/D units, as a function of event position on the camera face.

The progress of the search algorithm, described in section 2.1, as it locates the maximum-likelihood-estimated position for a single event is shown in Figure 2. To illustrate the path towards convergence, the log likelihood for the event data vector at every possible position on the camera face is plotted as an intensity embedded in the zero-likelihood region specified in step 1. The number of iterations required is determined by the number of effective pixels that are defined to exist on the detector face. The example shown is for an 80×80 pixel camera; each four-fold increase in the number of pixels adds 1 iteration step.

The most significant features of the search algorithm are: 1) a known number of iterations are required for convergence regardless of data vector values; 2) the number of iterations grows slowly with an increasing number of pixels or precision of estimate; 3) the processing time scales linearly with the number of PMTs or other sensors; 4) no conditional statements are required for implementation, only a sort at the end of each iteration to identify the test location with the highest log likelihood. From a computational point of view, the initial grid in the first iteration is always the same, and all subsequent grid locations can be quickly generated with pointer arithmetic about the grid center location.

2.4 Implementation in Hardware

These features make it possible to implement the algorithm in, for example, a field-programmable gate array (FPGA).⁶ Each iteration makes a natural stage for a processing pipeline, an architecture in which a computation is broken down into a set of steps governed by a clock. On each clock cycle, a new piece of data is fed in to the beginning of the pipeline, every partially processed piece of data advances by one step, and one result emerges from the last pipeline stage. This process is illustrated in Figure 3. The computations required for each iteration are diagrammed in Figure 4. Each stage of the ML-estimation pipeline conveniently has exactly the same structure; only the offsets for the pointer arithmetic need to be different. The event log likelihoods for the ensemble of test locations may all be computed in parallel, and as shown in Figure 5, those computations themselves can be made parallel. A quick analysis suggests that more than 10^6 events per second could be processed on a dedicated ML-estimation board.

3. OTHER APPLICATIONS OF MLE IN OPTICAL SENSORS

Maximum-likelihood methods are applicable to a very broad spectrum of data-processing tasks in optical sensors, and often are less complicated to implement than might be assumed. For example, MLE can be used to estimate not only x and y coordinates of gamma-ray events in scintillation cameras, but also photon energy, depth of interaction (the z coordinate), and event time. ML estimation can be used to optimally deduce event location,⁷ depth of interaction,⁸ event timing and energy deposition also in solid-state detectors. Other applications include optical testing, adaptive optics, resolution recovery and tomographic image reconstruction.

4. DISCUSSION AND CONCLUSIONS

There are currently multiple platforms on which it's possible to implement an ML algorithm such as the grid search described above, each with some advantages and drawbacks. For

example, a networked cluster of commodity processors offers the attractive features of being general purpose, parallel, easily programmed in multiple languages, and yielding codes that are portable as computing resources are upgraded. However, pipelining is difficult and systems with many detectors streaming data at once may be expensive to accommodate.

Significant processing power is now being incorporated into graphical processing units (GPUs).⁹ These allow pipelining and have efficient vector operations, but also have limited instruction sets, are expensive to make massively parallel, and require programming specific to a processor architecture. As discussed above, FPGA implementations have the advantage of providing opportunities for pipelining and parallelization, but gate-array programming is challenging and processing boards may have to be custom designed. Modern hardware description languages at least make algorithms portable between FPGA vendors and chip generations.

In the position-estimation problem discussed in section 2.3, it was found that ML-estimation with Poisson statistics can be carried out with integer-only math by appropriate scaling. Other probability models may require floating-point operations, including divisions, that make them unfeasible to implement in anything but a networked cluster of CPUs. High dimensional spaces, as when there are a large number of parameters to estimate, are also challenging to navigate with a hardware-based search algorithm.

ACKNOWLEDGEMENT

This work was supported by NIH/NIBIB grant 5P41-EB002035.

REFERENCES

1. Barrett, HH.; Myers, KJ. Foundations of Image Science. New York: John Wiley and Sons; 2004.
2. Furenlid, LR.; Hesterman, JY.; Barrett, HH. Real-time data acquisition and maximum-likelihood estimation for gamma cameras; 14th IEEE-NPSS Real Time Conf. Proc.; 2005. p. 498-501.
3. Barrett, HH. Detectors for Small-Animal SPECT II: Statistical Limitations and Estimation Methods. In: Kupinski, MA.; Barrett, HH., editors. Chap. 3 in Small Animal SPECT. New York: Springer; 2005.
4. Milster TD, Aarsvold JN, Barrett HH, Landesman AL, Mar LS, Patton DD, Roney TJ, Rowe RK, Seacat RH III. A Full-Field Modular Gamma Camera. J. Nucl. Med. 1990; 31(5):632–639. [PubMed: 2341900]
5. Chen, Y.; Furenlid, LR.; Wilson, DW.; Barrett, HH. Calibration of Scintillation Cameras and Pinhole SPECT Imaging Systems. In: Kupinski, MA.; Barrett, HH., editors. Chapter 12 in Small Animal SPECT. New York: Springer; 2005.
6. Hesterman, JY. PhD Dissertation. University of Arizona; 2007. The Multi-Module Multi-Resolution SPECT System.
7. Marks DG, Barber HB, Barrett HH, Tueller J, Woolfenden JM. Improving performance of a CdZnTe imaging array by mapping the detector with gamma rays. Nucl. Meth. Phys. Res., A. 1999; 428:102–112.
8. Marks, DG. PhD Dissertation. University of Arizona; 2001. Estimation methods for semiconductor gamma-ray detectors.
9. Macedonia M. The GPU enters computing's mainstream. Computer. 2003; 36(10):106–108.

a)

b)

Fig. 1.

- a) A collimated source being scanned in regular two-dimensional grid across camera face during calibration.
- b) Mean signals in A/D units resulting from acquisition of ~5000 events per position on camera face. Each intensity plot corresponds to complete camera-face data for one PMT.

Fig. 2.

The six iterations of the fast-ML-search algorithm required to converge on a position estimate for a modular gamma camera are shown for a sample event. The intensity plot represents the log-likelihood function that is being sampled at the locations indicated.

Fig. 3.
The pipeline structure of the fast-search algorithm.

Fig. 4.

The computations in a single stage of the pipeline shown in Figure 3. The test-location event log likelihoods may all be computed in parallel.

Fig. 5.

The computations needed to compute a single event log likelihood in Figure 4. If log-mean values are stored with scaling by appropriate powers of two in order to bring an adequate number of significant digits to the left of the decimal point, integer multiplies may be used. The result can be rescaled with fast and convenient logical shifts.