INVESTIGATING AND EXTENDING THE QUANTA TRACKING ALGORITHM

INVESTIGATING AND EXTENDING THE QUANTA TRACKING ALGORITHM

BY

JOSH GILMOUR, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2021)
(electrical & computer engineering)

McMaster University
Hamilton, Ontario, Canada

TITLE: Investigating And Extending The Quanta Tracking Algorithm

AUTHOR: Josh Gilmour
B.Eng. (Electrical & Computer Engineering),
McMaster University, Hamilton, Canada

SUPERVISOR: Dr. T. Kirubarajan

NUMBER OF PAGES: x, 50

Abstract

The traditional tracking approach of forming detections and then associating these detections together is known to perform poorly at low signal-to-noise ratios (SNR). Track-before-detect (TBD) approaches, where the sensor data is used directly as opposed to forming detections, has been shown to perform better than traditional approaches at low SNRs.

One recently introduced TBD algorithm is the Quanta Tracking Algorithm that is formed by applying maximum likelihood estimation to the histogram probabilistic multi-target tracker (HPMHT). Quanta has shown promising performance for low SNR scenarios, but the body of literature is small and the evaluations that have been done so far do not consider several practical aspects of using the algorithm in real applications and are difficult to compare to other algorithms due to the SNR definitions used. This paper seeks to address these deficiencies in the literature. A re-derivation of Quanta that corrects some issues present in the original derivation while also integrating extensions from the HPMHT literature will also be presented. These extensions are shown to make Quanta able to correct for errors in the assumed size targets as well as improve estimating the SNR of fluctuating targets.

To my friends

Acknowledgements

Thanks to Dr. Kiruba for supervising me and introducing me to the field of tracking. He is an approachable, kind and brilliant man that has built a well-deserved legacy in this field.

Thanks to Mehrnoosh Heidarpour for helping to fill in the holes in the derivation of the original Quanta algorithm. She will have a great career in academia.

Thanks to Mike Bradford for his insight and support throughout the development of this paper.

Thanks for Dr. Davidson for being so approachable despite being the chair of the whole department.

Thanks to Cheryl Gies for always being helpful and kind, even when I screw up forms more than once.

Contents

List of Figures

List of Tables

Abbreviations

HPMHT	Histogram Probabilistic Multi-Hypothesis Tracker
MLE	Maximum likelihood estimation
ML-HPMHT	Maximum Likelihood Histogram Probabilistic Multi-Hypothesis Tracker
TBD	Track-before-detect
EM	Expectation-maximization
SNR	Signal-to-noise ratio

Chapter 1

Introduction

1.1 The Tracking Problem and Common Approaches

The field of tracking is concerned with estimating the properties of targets given a set of sensor data. The properties to estimate for a given application are generally some subset of position, velocity, and shape. This information is used to understand a situation so that appropriate action can be taken. For example, in autonomous vehicles, one application of tracking algorithms would be to understand the positions of other vehicles on the road to ensure that a collision does not occur. Tracking algorithms have been applied to the data from numerous sensor types across an incredible variety of applications.

Traditional approaches to tracking operate on point detections that are formed by applying a threshold to sensor data. For example, an infrared camera returns an image where each pixel holds a value based on the temperature of the environment in the direction that corresponds to that pixel. A point detection can be formed at each pixel whose value is higher than a threshold. A tracking algorithm can be applied to these detections over a series of images to form tracks that likely correspond to targets. The tracking problem is non-trivial because of two factors. The first is the presence of random fluctuations in the sensor data referred to as noise. The second is so-called background clutter that is due to non-targets in the environment. For the infrared camera application, an example of background clutter could be the sun, which would generate a high value in every pixel the sun falls on. Noise and background clutter combine to create high values in sensor data that can be mistaken for targets. Tracking algorithms are rigorous approaches to determining the properties (and sometimes the number) of targets in the scene while minimizing error by accounting for noise and background clutter.

There are many traditional tracking approaches that operate on point detections. These approaches first attempt to associate the detections from successive sets of sensor data together. Common data association algorithms include Global Nearest Neighbor (Konstantinova *et al.*, 2003), Probabilistic Data Association (Bar-Shalom *et al.*, 2009), Multiple Hypothesis Tracking (Blackman, 2004), and Probabilistic Multiple Hypothesis Tracking (PMHT). State estimation approaches such as Kalman Filtering (Welch and Bishop, 2006) are then applied to these successive detections to estimate the properties of the target.

Other tracking algorithms take the sensor data itself as input, rather than point detections derived from it. These are referred to as Track-Before-Detect (TBD) algorithms. Examples include the particle filter, the Viterbi algorithm, and the Histogram Probabilistic Multi-Hypothesis Tracker (HPMHT) (Davey *et al.*, 2008).

1.2 The Benefits and Cost of Track-Before-Detect

Traditional tracking approaches that operate on point detections are effective when the signal to noise ratio (SNR) for targets is sufficiently high. TBD algorithms are generally applied when the target SNR is low enough that traditional approaches are not effective. Operating directly on the sensor data stops target information from being lost, which is especially critical in low SNR scenarios (Davey and Rutten, 2007). Unfortunately, this improved tracking performance generally comes with very high computational costs that can preclude TBD algorithms from being applied in real applications (Davey and Rutten, 2007), (Davey *et al.*, 2008).

For applying TBD in practical applications, the Histogram Probabilistic Multi-Hypothesis Tracker (HPMHT) was shown in (Davey and Rutten, 2007) and (Davey *et al.*, 2008) to have significant potential due to its comparable tracking performance with other TBD methods while having a radically lower computational cost. This potential inspired the authors to investigate the algorithm further.

1.3 The State of the Art and Limitations

HPMHT was introduced in (Streit and Lane, 2000) and is generally formulated sequentially. The algorithm uses a model with a known number of Gaussian targets within the sensor plane and a known clutter model. The basic formulation further assumes that the shape of the targets (that is, the covariance matrix that describes their point spread function) is constant and known. Since its introduction, a considerable body of research has built up around HPMHT.

HPMHT was applied to a practical tracking application for radar data in (Davey, 2010). The paper presented a detector-driven track management strategy for HPMHT that yielded a tracker that did not require prior knowledge of the number of targets within the sensor plane. An approach for extending HPMHT to non-Gaussian targets was also discussed.

Modeling the target spread covariance matrices as random matrices (RM) was done in (Wieneke and Davey, 2014) to relieve the assumption of constant and known target shapes. The target covariance matrices are estimated in a similar process to estimating the mean target position. Applying RM to HPMHT was shown to improve tracking peformance in the presence of initialization errors in the target spread covariance matrices.

Part of the measurement model of HPMHT is that the source of each count in the image is a draw from a multinomial distribution. The number of components in the distribution is equal to the number of targets plus the number of components used to model the clutter (which can be as low as one). The normalized mixing proportion for each component in the distribution is one of the parameters estimated by the algorithm. These mixing proportions are assumed to be deterministic and either constant or time varying. This model does not account for targets with fluctuating amplitude. An alternative model was presented in (Gaetjens *et al.*, 2017) where the number of shots produced by each component is modeled by a Poisson distribution. This model allows allows a time-correlated estimate of the target SNR to be extracted that was shown to be much less susceptible to fluctuation than with the multinomial model. The result is a target SNR estimate that is much more useful for track management purposes.

The random matrix covariance model from (Wieneke and Davey, 2014) and the Poisson measurement model from (Gaetjens *et al.*, 2017) were combined in (Davey *et al.*, 2014). Some clutter components were assigned Gaussian models and had their parameters estimated by HPMHT alongside the targets. Combining these extensions expands HPMHT to be applicable to scenarios where a clutter model with fluctuating, moving, and shape-variant Gaussian components and known uniform components is suitable. In (Wieneke and Davey, 2014), RM-Poisson-HPMHT was shown to greatly reduce the number of false tracks created for an airborne imagery application compared to HPMHT with a uniform clutter model.

In (Davey, 2015), Davey presented an approach for approximating HPMHT that decoupled the targets in the estimation process while simultaneously significantly reducing the computational cost of estimation for each target. The result was a radically lowered computation and memory cost that was independent of the size of the sensor plane. The decoupling also resulted in an algorithm that was much more amenable to parallel computing.

Maximum-likelihood estimation (MLE) was combined with HPMHT to form ML-HPMHT in (Willett *et al.*, 2013). ML-HPMHT is a batch estimator that assumes targets have a straight trajectory across the batch (though any parameterizable trajectory could be used with a re-derivation). The estimation process ends up being similar to core HPMHT. The presented formulation lacks the many extensions that were integrated into HPMHT to deal with its limitations. Thus, Quanta assumes that the spread of the targets is a known constant and uses the same multinomial mixing model that was previously discussed. Nonetheless, the results in (Willett *et al.*, 2013) and (Dunham *et al.*, 2019) demonstrate that the tracking performance of Quanta for very low SNR targets is promising. Unfortunately, the body of literature for Quanta is small and the evaluations that have been done so far do not consider several practical aspects of using the algorithm in real applications and are difficult to compare to other algorithms due to the SNR definitions used. Furthermore, the presented ML-HPMHT derivation has significant missing information and what is there has errors. These deficiencies in the literature are what we seek to expand upon with this paper.

Note that ML-HPMHT was later rebranded by the authors as the Quanta tracking algorithm. For the sake of consistency, we will thus refer to ML-HPMHT as Quanta for most of the remainder of this paper.

A more complete summary of the HPMHT literature can be found in (Davey and Gaetjens, 2018).

1.4 Objectives

This paper will provide two main contributions. The first contribution is to add to the existing analysis of the basic Quanta algorithm by evaluating the algorithms estimation performance when common track initialization approaches are used while also analyzing the iterations required for Quanta to reach convergence. Both of these aspects will be analyzed in a way that is as generally applicable as possible in an attempt to aid those evaluating Quanta for use in real applications. In line with that goal, this analysis will use a more common SNR definition than that applied in the Quanta literature to help with comparing the algorithm to other methods. The second contribution is integrating two of the extensions developed for HPMHT into Quanta. The extensions chosen were the deterministic Gaussian appearance estimation first presented in (Streit and Lane, 2000) and the Poisson measurement model presented in (Gaetjens *et al.*, 2017). Integrating these extensions provides the opportunity to re-derive the algorithm more verbosely than what was presented in (Willett *et al.*, 2013) and (hopefully) without error. A final minor contribution of this paper is to discuss implementation considerations for applying Quanta to real applications.

1.4.1 Contributions

1. An evaluation of the convergence performance of Quanta using common initialization strategies for multiple target SNRs and initial errors

2. Extend Quanta by integrating a Poisson measurement model and deterministic Gaussian appearance estimation. In doing so, a complete and error-free derivation of Quanta will be provided that will help to fill in the gaps left in the original derivation of Quanta.

    - This contribution will be published as the following paper:
      Gilmour J., Heidarpour M., Kirubarajan T. (2021). *Extending the Quanta Tracking Algorithm*

3. A discussion of implementation considerations for applying Quanta to real applications

Chapter 2

Background

2.1 Expectation Maximization (EM)

The proceeding discussion of Expectation Maximization is based on the explanations from (Borman, 2004) and (Blume, 2002).

The probability density that relates how likely receiving a measurement vector w is given a model with parameters Θ is written as

$$p(w|\Theta) \tag{2.1.1}$$

For a given measurement vector w, the parameters Θ that maximize 2.1.1 is the maximum likelihood estimate of Θ. That is, it is the estimate of Θ that is most likely to have resulted in the given measurement vector. 2.1.1 where w is fixed and Θ is to be maximized with respect to is often referred to as the likelihood function for the problem and is written as

$$L(\Theta) = p(w|\Theta) \tag{2.1.2}$$

There may be additional variables in the model required to maximize with respect to Θ that we do not observe. These so-called hidden variables may be introduced because they make estimating Θ tractable. We will denote the hidden variables as Z and one realization of them as z.

Expectation Maximization is an iterative algorithm for maximizing with respect to Θ while taking into account the hidden variables Z. The algorithm is summarized by the equation

$$\Theta_{n+1} = \arg\max_{\Theta} E_{Z|X,\Theta_n}\left(\ln p(w, z|\Theta)\right) \tag{2.1.3}$$

where Θ_n represents the current estimate for parameters Θ and Θ_{n+1} is the new estimate formed by maximizing the so-called auxiliary function $E_{z|w,\Theta_n}\left(\ln p(w, z|\Theta)\right)$ with respect to Θ. Note how maximizing $p(w, z|\Theta)$ with respect to Θ requires knowledge of both the observation w *and* the hidden variables z. Thus, $p(w, z|\Theta)$ is referred to as the complete data likelihood. Furthermore, we see that the expectation is with respect to the probability density function of the hidden variables, $p(z|w, \Theta_n)$, which (for a fixed w) is equivalent to the likelihood function for Θ_n with respect to the hidden variables. We term this the missing data likelihood. Thus, 2.1.3 is the expectation of the logarithm of the complete data likelihood with respect to the missing data likelihood.

The EM algorithm is thus summarized as first setting initial values for the parameters Θ_0 and then performing the following iteration until convergence:

1. *Expectation step:* Form the auxiliary function by taking the conditional expectation of the log-likelihood of the complete data likelihood $p(w, z|\Theta)$ with respect to the missing data likelihood $p(z|w, \Theta_n)$

2. *Maximization step:* Maximize the auxiliary function with respect to Θ

The key reason why the EM algorithm works (the mathematical machinery for which is beyond the scope of this paper) is that the likelihood for the parameters Θ is non-decreasing with each iteration. Non-rigorously, when the auxiliary function converges via application of the EM algorithm, the Θ likelihood is at a local maximum.

2.2 Histogram Probabilistic Multi-Hypothesis Tracker (HPMHT)

The proceeding discussion of HPMHT is largely based on the explanations from (Davey and Gaetjens, 2018).

Imagine a sensor that produces two-dimensional images as data. The images are divided into a uniform grid of pixels that have integer values. Each component of the scene (be it clutter or a target) produce energy that is incident on the sensor. This energy is produces point measurements on the continuous space of the sensor plane, similar to photons hitting the lens of a camera. The sensor counts the point measurements from all scene components that fall into the two-dimensional space that corresponds to each pixel. This count is assigned to the pixel in the output image. In this sense, the output image can be viewed as a two-dimensional histogram that counts the number of point measurements that fell into the corresponding area of the sensor plane. It is important to understand that the sensor did not retain the locations of the point measurements within the pixels, nor any information about which scene component produced which point measurements.

The previous description is the model for the image that HPMHT uses. HPMHT is an EM algorithm and so it is clear from the description that two pieces of the missing information are the components from which each point measurement originated K_t and the exact location of those point measurements within the pixel Y_t.

We require a model to describe how the measurements produced by each component are distributed throughout the sensor plane. In the basic formulation of HPMHT, clutter measurements are uniformly distributed and target measurements are Gaussian distributed with a constant covariance. That is, the shape and size of the targets is constant.

The measurement model of basic HPMHT is that each point measurement has a probability of originating from each component m that is equal to that components mixing proportion $\pi_{m,t}$. Thus, we have $\sum_{m=1}^{M} \pi_{m,t} = 1$. Intuitively, $\pi_{m,t}$ describes the portion of the total image counts (or more generally, the total image energy) that component m is expected to produce.

HPMHT estimates the mixing proportions for all components and the target positions X_t for all targets given initial values for these parameters and an image. Defining Π_t as the set of mixing proportions for time t and N_t as the set of pixel counts in frame t (the image, essentially), we can then state using the notation from the previous EM section that:

- $w = N_t$
- $\Theta = X_t, \Sigma_t, N_t$
- $z = K_t, Y_t$

Denoting the current estimate for parameters with a circumflex (Ex: $\hat{a}$), the resulting EM auxiliary function is

$$Q(X_t, \Pi_t | \hat{X}_t, \hat{\Pi}) = \sum_{K_t} \int \log \left[p_{comp}(X_t, K_t, Y_t, N_t | \Sigma_t, \Pi_t) \right] p_{miss}(K_t, Y_t | N_t, \hat{X}_t, \hat{\Sigma}_t, \hat{\Pi}) dY_t \tag{2.2.1}$$

The complete likelihood is conditional on Π_t and Σ_t because the former is treated as an unknown constant and the later is a known constant. They are not modeled as random variables.

The approach to maximizing this auxiliary function with respect to Θ will be well illustrated in the derivation to come and so we will not dwell on the details here. The one key point that we will note is that the value of the Gaussian model for the target is that the maximization for the Gaussian parameters can be implemented as a Kalman filter update and thus is quite simple and intuitive.

Note that this description is for the sequential version of HPMHT. For the version based on a batch of frames, the Gaussian parameter update can be implemented as a Kalman smoother.

As an aside, attentive readers may have noticed in the introduction that PMHT was classified as a traditional tracking algorithm while HPMHT was classified as a TBD algorithm.

Given the information in this section, we can now clarify that HPMHT is simply the data association approach defined by PMHT applied directly to image data (Davey and Gaetjens, 2018). Because the input for HPMHT is the image data instead of point detections (as with PMHT), HPMHT is classified as a TBD algorithm.

2.3 Quanta

The reference used to write this section was the chapter on Quanta from (Davey and Gaetjens, 2018).

HPMHT treats the evolution of the target positions as a random process driven by noise. The kinematic evolution can alternatively be modeled as deterministic where a parameterized curve is estimated using an initial curve and the data. Such a model within the HPMHT framework defines a batch algorithm that amounts to a curve fitting problem on synthetic point measurements. The simplest parameterized curve is a straight line, which can be fully specified for a track by its beginning and end points in the batch.

What has been described here is the Quanta (ML-HPMHT) algorithm introduced by Willet in (Willett *et al.*, 2013). Quanta combines a linear deterministic kinematic evolution model with HPMHT image association. As opposed to the sequential HPMHT discussed in section 2.2, Quanta operates on a batch of frames and thus would normally be implemented as a sliding window.

Define X, Σ, Π, K, and N as the corresponding variables from section 2.2 but for the entire batch of frames. The auxiliary function for Quanta then takes the form

$$Q(X, \Pi | \hat{X}, \hat{\Sigma}, \hat{\Pi}) = \sum_{K} \int \log\{p_{comp}(Y, N, K | X, \Sigma, \Pi)\} p_{miss}(Y, K | N, \hat{X}, \hat{\Sigma}, \hat{\Pi}) dY \tag{2.3.1}$$

Note how the complete data likelihood is conditioned on the set of target positions X. This difference compared to equation 2.2.1 is due to the deterministic kinematic evolution model that Quanta uses.

Chapter 3

Estimation Performance of the Basic Quanta Tracking Algorithm

3.1 The Need for Further Analysis

There has been limited publications on Quanta since the algorithms introduction in (Willett *et al.*, 2013). These publications answered some initial questions but have left holes that make it difficult to determine whether Quanta is suitable for a particular application or not. We will begin by describing some holes in the existing literature and then proceed with experiments that attempt to fill them in.

The work thus far uses valid but HPMHT-specific SNR measures that are not understood by those not already familiar with HPMHT theory. Using unique and algorithm specific measures of SNR makes it difficult to compare Quanta against other algorithms for a use case based on looking at the results from the literature. In aim of addressing this, the proceeding analysis uses peak SNR to define the target intensity.

Furthermore, we find that the existing analysis (while thorough from a tracking perspective) does not consider the practical aspects of initializing tracks on sensor frames of large sizes such as high-resolution cameras. In line with wanting to make a suitability analysis for a use case easier, we attempt to analyze the algorithms estimation performance under reasonable initialization schemes that are based on traditional one and two-point detector driven approaches, similar to what was applied to HPMHT in (Davey, 2010).

The final aspect that the analysis will cover is the iterations required to reach convergence, which is important given the practical limitations of computation for many tracking applications, especially given that Quanta is a batch algorithm and thus carries a higher computational cost to each iteration. We will also touch on some failure cases for Quanta.

3.2 One-point Initialization

With one-point initialization, the track position at every frame in the batch would be assigned the position given by one detection from a detector. For a perfectly accurate detector and a target that is moving, only the batch position from the frame that the detection originated from will be correct. Following the linear motion assumption of Quanta, each subsequent or previous frame from the detection frame will have the target further away from the detection position and thus the given initialization for those frames. Recall that under Quanta, the track parameters are driven to the truth via an algorithmic force produced by the overlap of the true target Gaussian and the track Gaussian. Given the effect of noise, there will clearly be a level of overlap between the Gaussians for the non-detection frames that stops the algorithm from being able to force the track positions for those non-detection frames to the truth. Furthermore, in those cases where the algorithm can converge to the truth, the algorithm will take more iterations to do so as the relative distance apart of the Gaussians is increased. We seek to perform a convergence analysis of Quanta when using one-point initialization under various SNRs. In doing so, we will note cases where the algorithm stopped being able to consistently converge. That is, the limits of the algorithms convergence.

To generalize the analysis to various target sizes, we normalize the distances between the true subsequent target positions in the batch by the target standard deviation. We refer to this quantity as the "position error to standard deviation ratio" or "position error ratio".

The scenario used in the analysis is as follows. The image plane is 200 by 200 pixels. The batch size is 11 frames, with the true target position for frame 6 (the middle frame) in the center of the image plane. Given a position error to standard deviation ratio for a particular test, this is multiplied by the target spread standard deviation to get how much the true target positions should be offset from each other in subsequent frames. The resulting trajectory is linear and is longer end-to-end the larger the given ratio is.

The initial track positions for all frames were set equal to the center of the image plane, simulating a perfectly accurate single frame detector. The spread standard deviation of the target for both the X and Y dimension was set to 3.

Convergence was consistently defined as when the endpoints of the track were within a small threshold to the true endpoints.

Figure 3.1 and Figure 3.2 illustrate the results of the experiments. Note that the plots illustrate essentially the same thing, they just focus on either the SNR or position error to standard deviation ratio respectively by displaying it on the x axis. Note that all data points represent averages of 10 runs. Figure 3.1 shows that the number of iterations required to reach convergence decreases exponentially with the peak SNR of the target. A key observation is that even for targets with high SNR (for a TBD context), the algorithm requires a large number of iterations to reach convergence. For example, we see that for a position error ratio of 1 between frames and an SNR of 15, Quanta required nearly 300 iterations to reach convergence. For the same error ratio at a peak SNR of 6, Quanta required around 800 iterations to reach convergence. Given the significant computational cost of each iteration with Quantas batch-driven strategy, such a large number of iterations could make Quanta impractical for use-cases with limited computational resources. This is especially true if the scenarios have multiple targets because:

- The computational cost of a single iteration grows considerably with the number of targets

- The method for estimating the number of tracks that was presented in (Willett *et al.*, 2013) requires essentially running Quanta multiple times

The computational cost may still be acceptable if the position error ratio is low. This is equivalent to the ratio of the target velocity and size being low for the application. Alternatively, as we will see, using two-point initialization instead of one-point initialization may considerably lower the number of iterations required.

The final note we will make about this experiment is some failure cases that were found. Observe in Figure 3.2 that the final datapoint for the peak SNR 6, 10, and 15 curves are at a ratio of 1.8, 2.0, and 2.2, respectively. These were just below the largest ratios for the given SNRs for which we found Quanta was able to consistently converge to the solution. That is, above these ratios, the algorithm was often not converging.

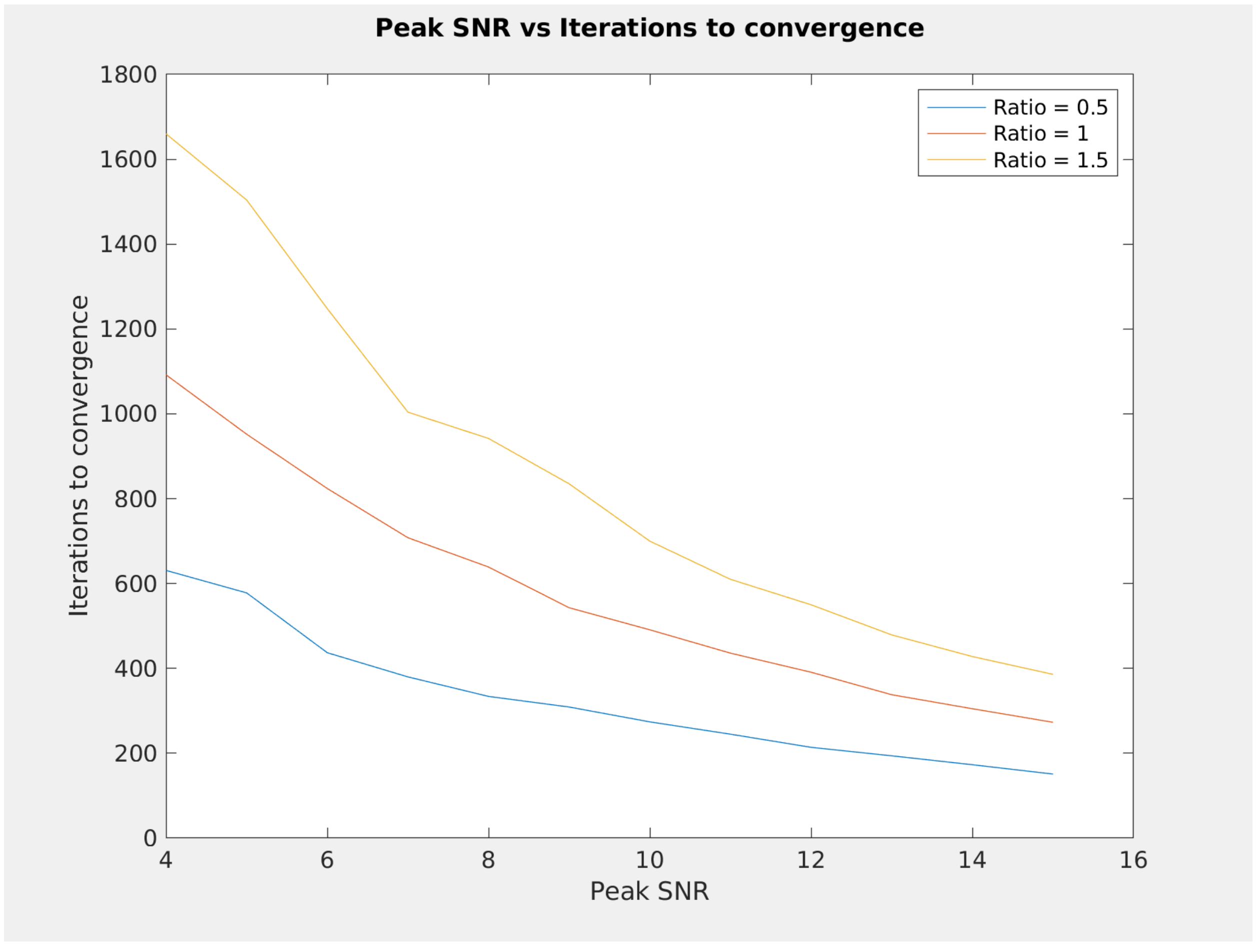

Figure 3.1: Relationship between target peak SNR and iterations to convergence for one-point initialization

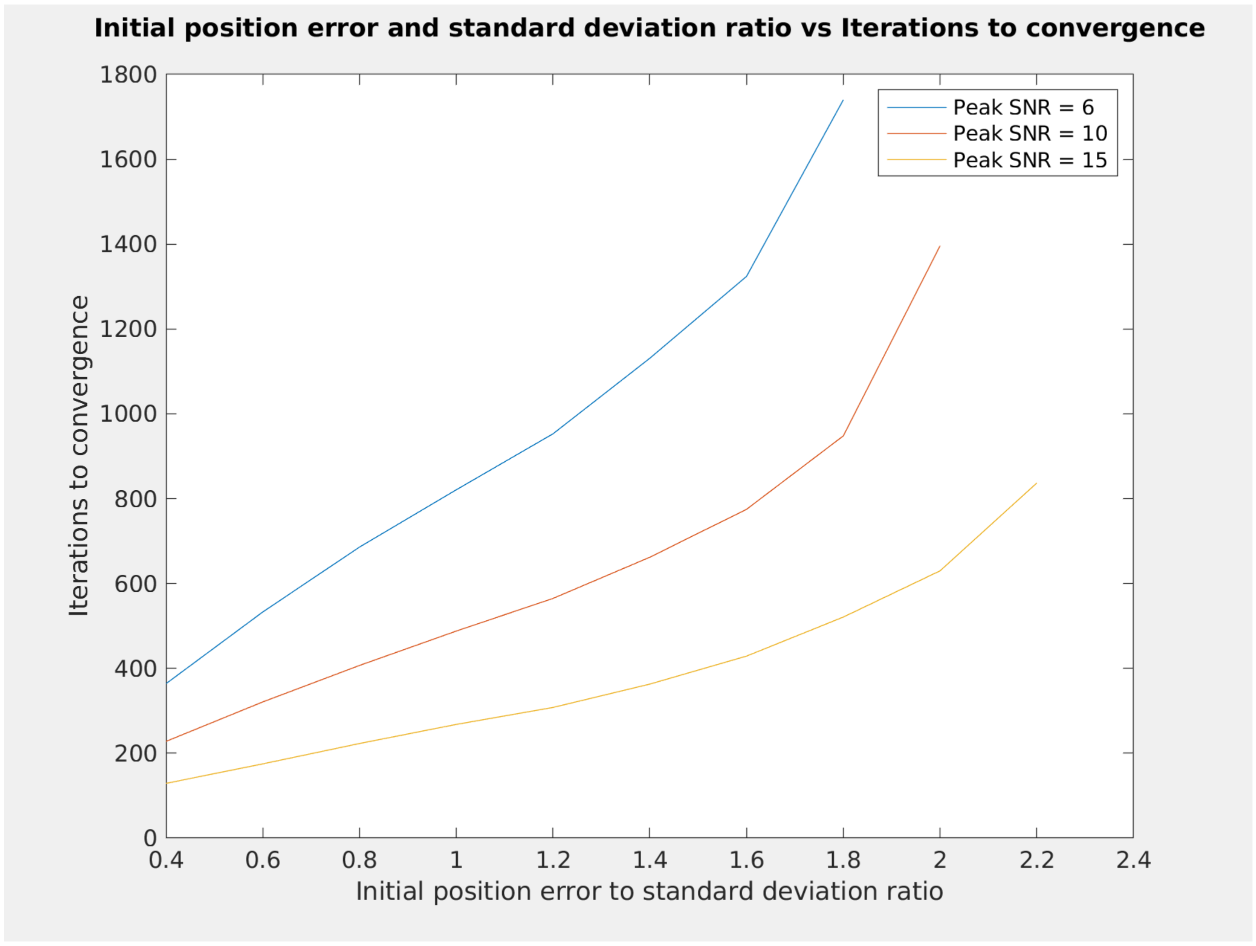

Figure 3.2: Relationship between the initial position error ratio and iterations to convergence for one-point initialization

3.3 Two-point Initialization

The two-point initialization case is more complicated to test than the one-point initialization case. We would like to still use one error ratio to define the test because that allows for a more direct comparison against the one-point initialization tests. However, two-point initialization is equivalent to defining the two endpoints of the batch. An approach must be decided for applying the single error ratio to specifying these two endpoints which still captures the various ways that errors in a two-point initialization could present themselves. The approach that was decided on was to use the position error ratio to define how far the initialization batch endpoints would be from the true batch endpoints. This defines a circle around each true endpoint that the initialization endpoint could lie on. The value assigned for the iterations to convergence for that position error ratio (and peak SNR) is then the average iterations to convergence when the initialization points along the circles were independently randomly selected. This is what is presented in Figures 3.3 and 3.4. The image plane was 200 by 200 pixels and the true target positions across the 11 frame batch went from position $(60.5, 60.5)$ to $(140.5, 140.5)$

As expected, we see that the iterations to convergence still decreases exponentially with increasing peak SNR. However, we see from Figure 3.4 that the iterations to convergence increases logarithmically as the position error ratio was increased for a given peak SNR. We can see a similar seemingly logarithmic increase in iterations for the one-point case at low position error ratios before the iterations begin to rise exponentially as the limits of the algorithms ability to converge to the truth is reached. It is expected that we would see a similar trend as the limits of convergence were approached if higher initial position errors were tested for the two-point case.

In general, the iterations required for convergence in the two-point initialization case are radically lower than in the one-point case. We see that the iterations to convergence at the highest tested position error ratio of 2.2 being lower for two-point initialization than the iterations to convergence for the lowest tested ratio of 0.4 for one-point initialization. The required iterations under two-point initialization appear significantly more practical and may allow the algorithm to be applied even to applications with limited computational resources, especially if the target velocity is low enough to result in lower position error ratios than those tested here.

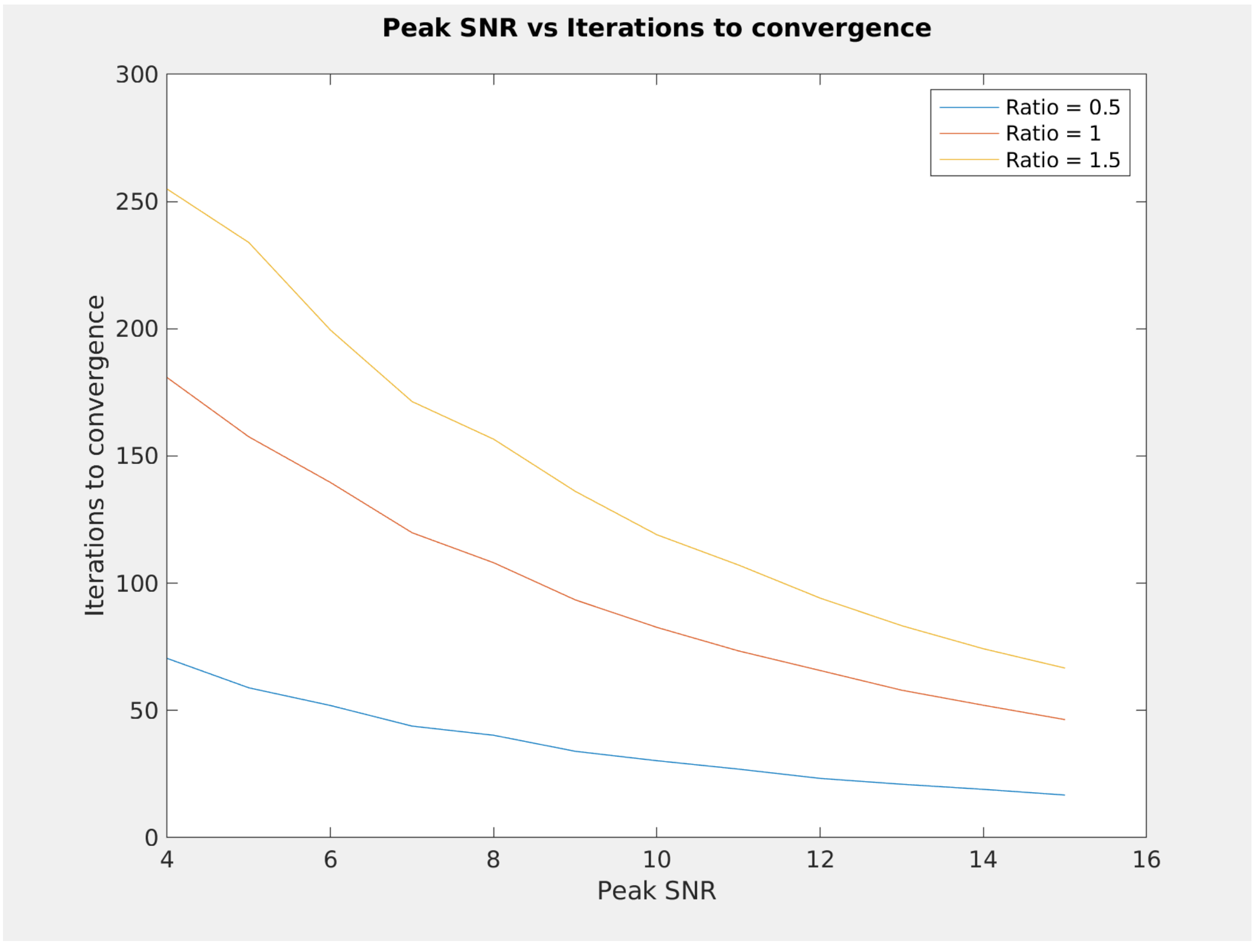

Figure 3.3: Relationship between target peak SNR and iterations to convergence for two-point initialization

This radical improvement in convergence speed is reasonable given how Quanta works. Even in the higher position error ratio cases for two-point initialization, the initial position for each frame is much closer to the true position than in the one-point case. There is simply less error in the initialization that must be corrected by iterating. This is especially true for the cases where the initialization ends up crossing the true trajectory, usually resulting in multiple frames initially being very close to the truth.

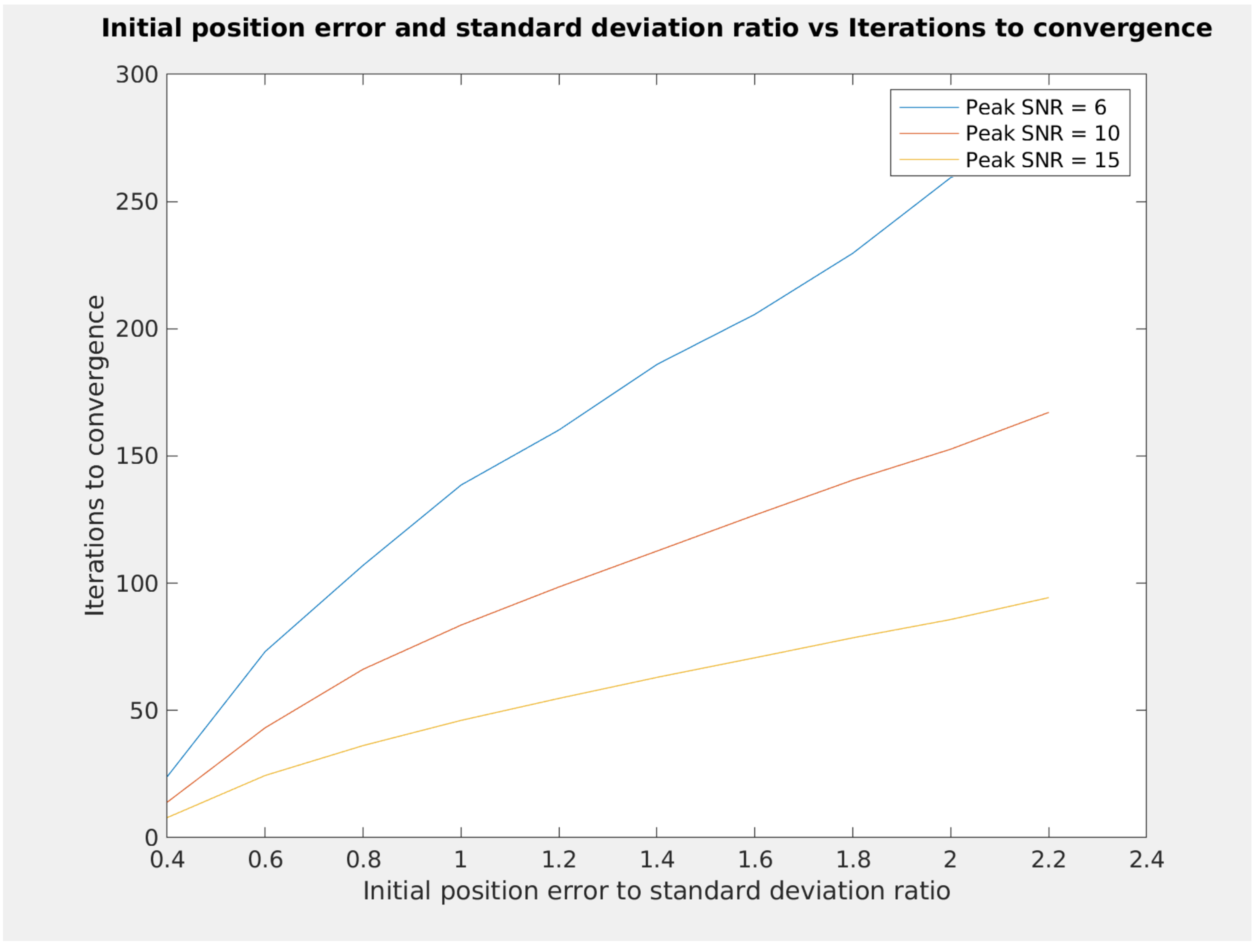

Figure 3.4: Relationship between the initial position error ratio and iterations to convergence for two-point initialization

3.4 Implementation Considerations

Applying Quanta to real applications may require minimizing the computational cost of the algorithm. The results for one-point initialization presented here and the possibility for two-point initializations of lower quality than what was tested here encourages us to consider how to minimize the computational cost of each iteration and applying the algorithm as a whole. Here we will briefly discuss strategies based on the literature and our own observations, but leave exploring the improvements for future work.

3.4.1 Single Chip Processing

In (Davey, 2015), Davey presented an approach for approximating HPMHT that decoupled the targets in the estimation process while simultaneously significantly reducing the computational cost of estimation for each target. The result was a radically lowered computation and memory cost that was independent of the size of the sensor plane. The decoupling also resulted in an algorithm that was much more amenable to parallel computing. This is especially attractive given the increasing availability of chips with parallel processing capabilities that are applicable to a wide range of systems and applications. It is expected that applying these approximations to Quanta would result in similar benefits.

The approximations presented in (Davey, 2015) for HPMHT are, briefly, as follows. Computation for each target is done only over a sub-image around the target, rather than the full frame. This makes computation independent of the size of the sensor plane and the computational benefits of not having to perform integrals over the entire sensor plane are obvious. For estimating the parameters for a given target, the initial parameters for all other targets are used. This decouples the estimation for each target and makes the algorithm amenable to parallel computation.

The decoupling can be applied to Quanta exactly as with HPMHT. However, estimation over a sub-image is more complicated given the batch nature of Quanta. The simplest approach would be to extract a sub-image for each frame for each target. The targets to consider in the overall estimation process would be any that effect those sub-images based on the parameters for those targets from the previous batch. Setting the sub-image size would be an application specific (and potentially dynamically changing) parameter that would be important for this single-chip Quanta to perform well.

3.4.2 Breaking the Tracking Problem Down

As previously referred to, the method for estimating the number of tracks that was presented in (Willett *et al.*, 2013) requires running Quanta multiple times. The approach is based around the minimum description length (MDL) that, to put it simply, assigns a score to a model based on a balance of how simple the model is and how well it describes the data (Grünwald, 2004). Estimating the number of tracks is described as follows:

1. Compute the MDL for a model with 0 targets, MDL_0
2. Compute the MDL using the Quanta-estimated parameters, assuming 1 target, MDL_1
3. If $\text{MDL}_1 > \text{MDL}_0$, compute the MDL using the Quanta-estimated parameters, assuming 2 targets
4. If $\text{MDL}_2 > \text{MDL}_1$, compute the MDL using the Quanta-estimated parameters, assuming 3 targets
5. ...and so on, until $\text{MDL}_n < \text{MDL}_{n-1}$. Accept that there is $n-1$ targets with the parameters estimated by Quanta

In a real application with a large sensor plane, this strategy would work if Quanta, when told to assume a target exists, was guaranteed to find that target (if it exists) regardless of how far off the initialization is. Unfortunately Quanta has reasonable limits to the distance over which it can converge, as we have touched on in this chapter. Consider a case where a detector has been used on a batch of frames where there is two targets present and we are at step 3 in the process above. We select one of the detections to initialize the second track at, but we unfortunately do not select the one that corresponds to the other target. If we run Quanta and get some erroneous final parameters for the track that was initialized on noise and then compute the MDL for these parameters, we should get $\text{MDL}_2 < \text{MDL}_1$, even though this is not correct. To reach the truth, after reaching step 3, you would instead have to run Quanta over and over again with the second track getting initialized at the different detections until you either get a parameter set that yields $\text{MDL}_2 > \text{MDL}_1$ (and thus you have hopefully found the second targets parameters) or you find for every detection that $\text{MDL}_2 < \text{MDL}_1$. The result is a ridiculous combinatorial problem that is entirely impractical.

An alternative is to break the problem down. Similarly to with Single Chip Processing, we can associate detections and tracks together to form sub-groups that are as small as possible while respecting the characteristics of targets for the application. A large set of sub-problems would be formed with much smaller combinatorial complexity than the overall target number estimation problem. These sub-problems could be dealt with independently, which when combined with the single chip processing approach described previously would help transform Quanta into an algorithm that is well suited to parallelization.

Chapter 4

Extending the Quanta Tracking Algorithm

We seek to re-derive the Quanta Algorithm in detail and (hopefully) without error while integrating the deterministic Gaussian appearance estimation first presented in (Streit and Lane, 2000) and the Poisson measurement model presented in (Gaetjens *et al.*, 2017). The presented derivation does not make the assumption of a single uniform clutter distribution. It also integrates modeling of shots from "unobserved pixels", which stops the target state estimate from being biased when the target is near the edge of the sensor plane. This bias was thoroughly discussed in (Davey and Gaetjens, 2018). The original development of the unobserved pixels approach was given in (Streit and Lane, 2000).

4.1 Extensions

4.1.1 Poisson measurement model

The estimated SNR of a target can be used as a part of track maintenance decisions. However, the mixing proportions estimated by HPMHT are only suitable for estimating the SNR of targets with a constant SNR or an SNR that is independent frame-to-frame (depending on the corresponding mixing proportion model that is used). In (Gaetjens *et al.*, 2017), the mixing proportion measurement model of HPMHT was replaced with a Poisson measurement model that essentially models the target energy as the total number of image shots associated with the target rather than as a proportion of the total image energy. This decouples the energy estimates for the tracks and allows for dynamics to be imposed on the individual track energy estimates.

Imposing dynamics on the track energy estimates was shown to make the resulting SNR estimate much more robust to fluctuation in the target energy. Thus far, Quanta has only been presented with the mixing proportion model from HPMHT. We will re-derive Quanta using a Poisson measurement model to realize the robustness to fluctuation in the target energy demonstrated by the change in (Gaetjens *et al.*, 2017).

4.1.2 Constant spread estimation

In the original HPMHT paper (Streit and Lane, 2000), Streit developed estimation of the spread (Gaussian covariance) of the target in the sensor plane. Estimating the spread allows the algorithm to inherently handle applications where the targets change in size (such as with a camera when the target comes closer or farther) and be robust to poor assumptions of the target spread. (Wieneke and Davey, 2014) demonstrated that HPMHT tracking performance can be significantly improved by applying spread estimation in the presence of spread initialization error. Unfortunately, the paper that introduced Quanta (Willett *et al.*, 2013) assumed that the target had a constant known spread. To realize the previously described benefits, Quanta will be re-derived under the assumption of an unknown but constant spread estimate.

Theoretically, targets could vary in size fast enough over the course of the batch that a single spread estimate still cannot accurately describe the evolution of the target state. To deal with these cases, Quanta could be re-derived under a Random-Matrix model for the spread that allows evolution over time. This was developed for HPMHT in (Wieneke and

Davey, 2014). We do not present this derivation here, but it should not be difficult given the proceeding derivation and the content of (Wieneke and Davey, 2014).

4.2 Derivation

The deterministic kinematic evolution model of Quanta is given by

$$x_t^m = (1-\phi)x_{begin}^m + \phi x_{end}^m \tag{4.2.1}$$

where ϕ falls in the range [0 1]. x_{begin}^m is the position of component m at the beginning of the batch. x_{end}^m is the position of component m at the end of the batch.

4.2.1 Expectation step

As with the basic HPMHT developed in (Streit and Lane, 2000), Quanta is an expectation-maximization (EM) algorithm. As with any EM algorithm, the auxiliary function must be formulated. The variables that define the complete and missing data are

Variable	**Definition**
X	Track states
Σ	Track spread matrices
Y	Exact point measurement locations
$\tilde{\Lambda}$	Quantized Poisson mixing rates
N	Number of shots in each pixel m
N^k	Number of shots in each cell attributed to each component m
K	Measurement sources

Due to the deterministic state and spread model employed by Quanta, neither X nor Σ are random variables. The form of the complete data likelihood is thus $p_{comp}(Y, \tilde{\Lambda}, N, K|X, \Sigma)$. Following Gaetjens et al. in (Gaetjens *et al.*, 2017), we can replace N with N^k in the complete data likelihood to write $p_{comp}(Y, \tilde{\Lambda}, N^k, K|X, \Sigma)$.

The missing data for this problem is the exact measurement locations Y and measurement sources K. The missing data likelihood is thus obtained as $p_{miss}(Y, K|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N)$, where $\hat{X}$, $\hat{\Sigma}$, $\hat{\tilde{\Lambda}}$ represents estimates from the previous iteration. The EM auxiliary function can then be written as

$$\begin{aligned} Q(X, \Sigma, \tilde{\Lambda}|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}) &= E_{Y,K}[\log\{p_{comp}(Y, \tilde{\Lambda}, N^k, K|X, \Sigma)\}] \\ &= \sum_K \int_Y \log\{p_{comp}(Y, \tilde{\Lambda}, N^k, K|X, \Sigma)\} p_{miss}(Y, K|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N) dY \end{aligned} \tag{4.2.2}$$

The complete data likelihood can be simplified as

$$p_{comp}(Y, \tilde{\Lambda}, N^k, K|X, \Sigma) = p(\tilde{\Lambda}) p(N^k, K|X, \Sigma, \tilde{\Lambda}) p(Y|X, \Sigma, K) \tag{4.2.3}$$

Due to the dynamics imposed on the Poisson mixing terms, a first-order Markov model is applied to the quantized $\tilde{\Lambda}$ (Gaetjens *et al.*, 2017).

$$p(\tilde{\Lambda}) = \prod_{m=0}^{M} \left[p(\tilde{\lambda}_0^m) \prod_{t=1}^{T} p(\tilde{\lambda}_t^m|\tilde{\lambda}_{t-1}^m) \right] \tag{4.2.4}$$

The deterministic model for X and Σ means that they have no associated prior.

Per results from (Gaetjens *et al.*, 2017), the shot and measurement source density can be written as

$$p(N^k, K|X, \Sigma, \tilde{\Lambda}) = \prod_{t=1}^{T} \frac{1}{exp\{\tilde{\lambda}_t\}} \prod_{i=1}^{I} \left[\frac{1}{n_t^i!} \prod_{r=1}^{n_t^i} \tilde{\lambda}_t^{K_t^{ir}} h^i(x_t^{K_t^{ir}}, \Sigma_t^{K_t^{ir}}) \right] \tag{4.2.5}$$

where $\tilde{\lambda}_t$ is the quantized Poisson rate parameter of the total number of shots $||N_t||$. $N_t = \{n_t^i\}_{i=1}^I$ is a quantized vector where each element represents the number of shots from the corresponding pixel. K_t^{ir} specifies the component index of the rth shot in the ith cell at time t. Thus, $\tilde{\lambda}_t^{K_t^{ir}}$, $x_t^{K_t^{ir}}$, $\Sigma_t^{K_t^{ir}}$ refer to the quantized Poisson intensity rate parameter, state, and spread for the component with index K_t^{ir}. If $h(\tau|x_t^m, \Sigma_t^m)$ is the point spread function (PSF) for component m at time t, then the probability of a shot due to the mth component falling into the ith cell is given by $h^i(x_t^m, \Sigma_t^m) = \int_{B_i} h(\tau|x_t^m, \Sigma_t^m) d_\tau$.

Per (Davey and Gaetjens, 2018), the measurement likelihood can be written as

$$p(Y|X, \Sigma, K) = \prod_{t=1}^{T} \prod_{i=1}^{I} \prod_{r=1}^{n_t^i} \frac{h^{K_t^{ir}}(y_t^{i,r}|x_t^{K_t^{ir}}, \Sigma_t^{K_t^{ir}})}{h^i(x_t^{K_t^{ir}}, \Sigma_t^{K_t^{ir}})} \tag{4.2.6}$$

We can then write the complete data likelihood as

$$p_{comp}(Y, \tilde{\Lambda}, N^k, K|X, \Sigma) = p(\tilde{\Lambda}) \prod_{t=1}^{T} \frac{1}{exp\{\tilde{\lambda}_t\}} \prod_{i=1}^{I} \left[\frac{1}{n_t^i!} \prod_{r=1}^{n_t^i} \tilde{\lambda}_t^{K_t^{ir}} h^{K_t^{ir}} \left(y_t^{i,r}|x_t^{K_t^{ir}}, \Sigma_t^{K_t^{ir}} \right) \right] \tag{4.2.7}$$

Applying the logarithm,

$$\begin{aligned} \log\{p_{comp}(Y, \tilde{\Lambda}, N^k, K|X, \Sigma)\} = & \log\{p(\tilde{\Lambda})\} - \sum_{t=1}^{T} \tilde{\lambda}_t + \sum_{t=1}^{T} \sum_{i=1}^{I} \log\left\{\frac{1}{n_t^i!}\right\} + \\ & \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{r=1}^{n_t^i} \log\left\{\tilde{\lambda}_t^{K_t^{ir}}\right\} + \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{r=1}^{n_t^i} \log\left\{h^{K_t^{ir}} \left(y_t^{i,r}|x_t^{K_t^{ir}}, \Sigma_t^{K_t^{ir}} \right)\right\} \end{aligned} \tag{4.2.8}$$

The first three terms in (4.2.8) do not involve the missing data and thus are their own expectations. Defining $\hat{\tilde{\lambda}}_t^m$ and $\hat{x}_t^m$ as the estimated quantized Poisson mixing term and state from the previous iteration, it was shown in (Gaetjens *et al.*, 2017) that the expectation of the fourth term is

$$\sum_K \int_Y \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{r=1}^{n_t^i} \log\left\{\tilde{\lambda}_t^{K_t^{ir}}\right\} p_{miss}(Y, K|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N) dY = \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{m=0}^{M} n_t^i \mu_t^{im} \log\{\tilde{\lambda}_t^m\} \tag{4.2.9}$$

where

$$\mu_t^{im} = \frac{\hat{\tilde{\lambda}}_t^m h^i(\hat{x}_t^m, \hat{\Sigma}_t^m)}{f_t^i} \tag{4.2.10}$$

and

$$f_t^i = \sum_{m=1}^{M} \hat{\tilde{\lambda}}_t^m h^i(\hat{x}_t^m, \hat{\Sigma}_t^m) \tag{4.2.11}$$

We now collect the Poisson assignment prior terms together, which later can be maximized individually.

$$\mathcal{Q}_{\tilde{\Lambda}} = \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{m=0}^{M} n_t^i \mu_t^{im} \log\{\tilde{\lambda}_t^m\} - \sum_{t=1}^{T} \tilde{\lambda}_t + \log\{p(\tilde{\Lambda})\} \tag{4.2.12}$$

The expectation for the last term in (4.2.8) was shown in (Davey and Gaetjens, 2018) to be equal to

$$\begin{aligned} &\sum_K \int_Y \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{r=1}^{n_t^i} \log\left\{h^{K_t^{ir}}\left(y_t^{i,r}|x_t^{K_t^{ir}}, \Sigma_t^{K_t^{ir}}\right)\right\} p_{miss}(Y, K|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N)dY \\ &= \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{m=0}^{M} \frac{n_t^i \mu_t^{im}}{h^i(\hat{x}_t^m, \hat{\Sigma}_t^m)} \int_{W^i} \log\{h(y|x_t^m, \Sigma_t^m)\} h(y|\hat{x}_t^m, \hat{\Sigma}_t^m)dY \end{aligned} \tag{4.2.13}$$

where W^i notates the region of the ith pixel. Denoting $\xi_t^{im} = n_t^i \mu_t^{im}$ and $||\xi_t^m|| = \sum_{i=1}^{I} \xi_t^{im}$, (4.2.13) was shown to simplify for a single component to (Davey and Gaetjens, 2018)

$$-\frac{1}{2}\sum_{t=1}^{T} ||\xi_t^m|| \log\{|\Sigma_t^m|\} - \frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{I} \frac{\xi_t^{im}}{h^i(\hat{x}_t^m, \hat{\Sigma}_t^m)} \int_{W^i} (x_t^m - y)^T (\Sigma_t^m)^{-1} (x_t^m - y) h(y|\hat{x}_t^m, \hat{\Sigma}_t^m)dY \tag{4.2.14}$$

Define the equivalent measurement for the mth component to be

$$\tilde{y}_t^m = \frac{1}{||\xi_t^m||} \sum_{i=1}^{I} \frac{\xi_t^{im}}{h^i(\hat{x}_t^m, \hat{\Sigma}_t^m)} \tilde{y}_t^{mi} \tag{4.2.15}$$

where

$$\tilde{y}_t^{mi} = \int_{W^i} y h(y|\hat{x}_t^m, \hat{\Sigma}_t^m)dY \tag{4.2.16}$$

As shown in (Davey and Gaetjens, 2018), we can then expand and manipulate the second term in (4.2.14) such that (4.2.14) can be written as

$$-\frac{1}{2}\sum_{t=1}^{T}\left\{||\xi_t^m||\log\{|\Sigma_t^m|\} + ||\xi_t^m||(x_t^m - \tilde{y}_t^m)^T(\Sigma_t^m)^{-1}(x_t^m - \tilde{y}_t^m) + \text{trace}\left[\tilde{Z}_t^m(\Sigma_t^m)^{-1}\right]\right\} \tag{4.2.17}$$

where

$$\tilde{Z}_t^m = \sum_{i=1}^{I}\xi_t^{im}\tilde{Z}_t^{mi} = \sum_{i=1}^{I}\frac{\xi_t^{im}}{h^i(\hat{x}_t^m,\hat{\Sigma}_t^m)}\int_{W^i}(y-\tilde{y}_t^m)^T(\Sigma_t^m)^{-1}(y-\tilde{y}_t^m)h(y|\hat{x}_t^m,\hat{\Sigma}_t^m)dY \tag{4.2.18}$$

From (Davey and Gaetjens, 2018) and (Wieneke and Davey, 2014), we can now write the full state and spread component of the auxiliary function for the mth component as

$$\mathcal{Q}_{X,\Sigma}^m = -\frac{1}{2}\sum_{t=1}^{T}\left\{||\xi_t^m||\log\{|\Sigma_t^m|\} + ||\xi_t^m||(x_t^m - \tilde{y}_t^m)^T(\Sigma_t^m)^{-1}(x_t^m - \tilde{y}_t^m) + \text{trace}\left[\tilde{Z}_t^m(\Sigma_t^m)^{-1}\right]\right\} \tag{4.2.19}$$

Unobserved pixels

We will now develop modeling of unobserved pixels into Quanta. Note that the statement of the EM iteration of the basic Quanta algorithm in (Davey and Gaetjens, 2018) accounts for unobserved pixels, but a derivation is only given in the context of sequential HPMHT. We take the authors approach to provide a derivation for Quanta.

Denote the number of unobserved pixels as I^U and the set of shot counts from unobserved pixels as N^U. The set of shot counts due to each component in unobserved pixels is then N^{kU}. The statement of the auxiliary function is now

$$\begin{aligned} Q(X,\Sigma,\tilde{\Lambda}|\hat{X},\hat{\Sigma},\hat{\tilde{\Lambda}}) &= E_{Y,K,N^U}[\log\{p_{comp}(Y,\tilde{\Lambda},N^k,N^{kU},K|X,\Sigma)\}] \\ &= \sum_{N^U}\sum_{K}\int_{Y}\log\{p_{comp}(Y,\tilde{\Lambda},N^k,N^{kU},K|X,\Sigma)\}p_{miss}(Y,K,N^U|\hat{X},\hat{\Sigma},\hat{\tilde{\Lambda}},N)dY \end{aligned} \tag{4.2.20}$$

The missing data can be split such that

$$p_{miss}(Y, K, N^U|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N) = p(Y, K|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N, N^U)p(N^U|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N) \tag{4.2.21}$$

We treat the Poisson assignment prior mixing terms first. The later two terms in (4.2.12) are independent of the shot counts and thus are their own expectation with respect to N^U.

$$\sum_{N^U} \mathcal{Q}_{\tilde{\Lambda}} p(N^U|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N) = \log\{p(\tilde{\Lambda})\} - \sum_{t=1}^{T} \tilde{\lambda}_t + \sum_{N^U} \sum_{t=1}^{T} \sum_{i=1}^{I+I^U} \sum_{m=0}^{M} n_t^i \mu_t^{im} \log\{\tilde{\lambda}_t^m\} p(N^U|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N) \tag{4.2.22}$$

The last term in (4.2.22) can be split into two an observable component over $1 \le i \le I$ and an unobservable component over $I < i \le I + I^U$. The observable component is independent of the counts of the shot counts for unobserved pixels and thus is its own expectation with respect to N^U. Meanwhile, for the unobservable component, the summand only depends on the shot count for that particular unobserved pixel and thus is independent of the others. These observations allow us to write

$$\begin{aligned} &\sum_{N^U} \sum_{t=1}^{T} \sum_{i=1}^{I+I^U} \sum_{m=0}^{M} n_t^i \mu_t^{im} \log\{\tilde{\lambda}_t^m\} p(N^U|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N) \\ &\quad = \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{m=0}^{M} n_t^i \mu_t^{im} \log\{\tilde{\lambda}_t^m\} + \sum_{t=1}^{T} \sum_{i=I+1}^{I+I^U} \sum_{m=0}^{M} \mu_t^{im} \log\{\tilde{\lambda}_t^m\} \sum_{n_t^i=0}^{\infty} n_t^i p(n_t^i|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N) \end{aligned} \tag{4.2.23}$$

The expectation of n_t^i in the second line of (4.2.23) was shown in (Davey and Gaetjens, 2018) to equal

$$\sum_{n_t^i=0}^{\infty} n_t^i p(n_t^i|\hat{X}, \hat{\Sigma}, \hat{\tilde{\Lambda}}, N) = \frac{f_t^i}{\sum_{j=1}^{I} f_t^j}(n_t + 1) \tag{4.2.24}$$

The expected measurement count $\bar{n}_t^i$ can then be defined as

$$\bar{n}_t^i = \begin{cases} n_t^i, & 1 \le i \le I \\ \frac{f_t^i}{\sum_{j=1}^{I} f_t^j}(n_t + 1), & I < i \le I + I^U \end{cases} \tag{4.2.25}$$

We can then write the Poisson assignment prior component of the auxiliary function with unobserved pixels as

$$\mathcal{Q}_{\tilde{\Lambda}} = \log\{p(\tilde{\Lambda})\} - \sum_{t=1}^{T} \tilde{\lambda}_t + \sum_{t=1}^{T} \sum_{i=1}^{I+I^U} \sum_{m=0}^{M} \bar{n}_t^i \mu_t^{im} \log\{\tilde{\lambda}_t^m\} \tag{4.2.26}$$

We see that including unobserved pixels as missing information amounted to the same form as (4.2.12) but with the addition of terms that use the expected measurement count for the unobserved pixels.

Noting that (4.2.13) is the only term that forms $\mathcal{Q}^m_{X,\Sigma}$ that involves n_t^i and following similar mechanics as for (4.2.12), it is easy to see that the only necessary modification to account of unobserved pixels in (4.2.19) is to use the expected number of measurements in the calculation of ξ_t^{im} and to change the sum over I for the calculation of $||\xi_t^m||$, $\tilde{y}_t^m$ (equation (4.2.15)), and $\tilde{Z}_t^m$ (equation (4.2.18)) to be from $1 < i < I + I^U$.

$$\xi_t^{im} = \bar{n}_t^i \mu_t^{im} \tag{4.2.27}$$

$$||\xi_t^m|| = \sum_{i=1}^{I+I^U} \xi_t^{im} \tag{4.2.28}$$

$$\tilde{y}_t^m = \frac{1}{||\xi_t^m||} \sum_{i=1}^{I+I^U} \frac{\xi_t^{im}}{h^i(\hat{x}_t^m, \hat{\Sigma}_t^m)} \tilde{y}_t^{mi} \tag{4.2.29}$$

$$\tilde{Z}_t^m = \sum_{i=1}^{I+I^U} \xi_t^{im} \tilde{Z}_t^{mi} = \sum_{i=1}^{I+I^U} \frac{\xi_t^{im}}{h^i(\hat{x}_t^m, \hat{\Sigma}_t^m)} \int_{W^i} (y - \tilde{y}_t^m)^T (\Sigma_t^m)^{-1} (y - \tilde{y}_t^m) h(y|\hat{x}_t^m, \hat{\Sigma}_t^m) dY \tag{4.2.30}$$

Again we see that the form of the equations remains the same. This is important for maximization of these auxiliary functions because the approaches used are unchanged.

Limit of the Quantization

The final step of the derivation of the auxiliary function is to take the limit of the quantization to zero. This allows sensor streams that have continuous pixel values, such as radar, to use the given algorithm. We do not have to consider the significance of the prior being affected by the limit as was necessary in (Streit and Lane, 2000) because the target state is not modeled as a random variable in the Quanta algorithm. The data dependence on the prior that is applied in (Streit and Lane, 2000) to fix this problem is also not necessary to apply to the Poisson mixing rate prior because they are already quantized (Gaetjens *et al.*, 2017).

Denote z_t^i as the continuous count for the ith pixel and the quantization level to be c. The expected quantized pixel count is then $\bar{n}_t^i = \text{floor}\left(\frac{z_t^i}{c}\right)$ (Davey and Gaetjens, 2018). By taking the limit of the quantization for $\bar{n}_t^i$ to zero we get

$$\lim_{c \to 0} c\bar{n}_t^i = \begin{cases} z_t^i, & 1 \leq i \leq I \\ \frac{f_t^i}{\sum_{j=1}^{I} f_t^j} z_t^i, & I < i \leq I + I^U \end{cases} \tag{4.2.31}$$

From (Gaetjens *et al.*, 2017), taking the quantization limit to zero causes the quantized Poisson mixing rates $\tilde{\lambda}_t^m$ to be replaced with continuous equivalents λ_t^m. Given that and the result from (4.2.31), we see that several equations must be modified.

$$\xi_t^{im} = \bar{z}_t^i \mu_t^{im} \tag{4.2.32}$$

$$\mu_t^{im} = \frac{\hat{\lambda}_t^m h^i(\hat{x}_t^m, \hat{\Sigma}_t^m)}{f_t^i} \tag{4.2.33}$$

$$f_t^i = \sum_{m=1}^{M} \hat{\lambda}_t^m h^i(\hat{x}_t^m, \hat{\Sigma}_t^m) \tag{4.2.34}$$

where $\bar{z}_t^i$ is the expected continuous count for the ith pixel.

The form of (4.2.19) is unchanged by the limit. The Poisson mixing rate auxiliary becomes

$$\mathcal{Q}_\Lambda = \log\{p(\Lambda)\} - \sum_{t=1}^{T} \lambda_t + \sum_{t=1}^{T} \sum_{i=1}^{I+I^U} \sum_{m=0}^{M} \bar{z}_t^i \mu_t^{im} \log\{\lambda_t^m\} \tag{4.2.35}$$

4.2.2 Maximization steps

Mixing rates Λ

The maximization for the mixing rates, target states, and target spreads can be treated separately. First we will look to maximizing equation 4.2.35 with respect to Λ. The procedure is unchanged from (Gaetjens *et al.*, 2017) and is repeated here. The conjugate prior for the Poisson distribution is the gamma distribution(Gaetjens *et al.*, 2017). Define the gamma shape parameter for component m as α^m and the rate parameter as β^m. The previous, predicted, and updated shape parameter estimate is then denoted as $\alpha^m_{t-1|t-1}$, $\alpha^m_{t|t-1}$, $\alpha^m_{t|t}$ respectively. Similar definitions are made for β^m. Per (Gaetjens *et al.*, 2017), applying the gamma conjugate prior to (4.2.4) yields the posterior distribution for λ_t as

$$p(\lambda_t^m | N_t^m) \propto gamma(\lambda_t^m; \alpha^m_{t|t-1} + ||N_t^m||, \beta^m_{t|t-1} + 1) \tag{4.2.36}$$

$||N_t^m||$ is the total number of measurements associated with component m and can be estimated as (Gaetjens *et al.*, 2017)

$$||N_t^m|| \approx \sum_{i=1}^{I} \mu_t^{im} z_t^i \tag{4.2.37}$$

The prediction and update equations for the gamma parameters are given by Granström in (Granström and Orguner, 2012) as

$$\begin{aligned} \alpha^m_{t|t-1} &= exp\left\{\frac{-\delta_t}{\eta}\right\} \alpha^m_{t-1|t-1} & \beta^m_{t|t-1} &= exp\left\{\frac{-\delta_t}{\eta}\right\} \beta^m_{t-1|t-1} \\ \alpha^m_{t|t} &= \alpha^m_{t|t-1} + ||N_t^m|| & \beta^m_{t|t} &= \beta^m_{t|t-1} + 1 \end{aligned} \tag{4.2.38}$$

where δ_t is the duration between time scans and η determines the weight of the previous parameter estimate. η is referred to as the *forget factor*. We can then estimate λ_t^m as the mode of the gamma distribution

$$\lambda_t^m = \frac{\alpha_{t|t}^m - 1}{\beta_{t|t}^m} \tag{4.2.39}$$

Deterministic Gaussian Appearance

Following (Davey and Gaetjens, 2018) for maximizing (4.2.14) with respect to Σ_m^t, we can apply several derivative identities of matrices to find that the estimate for a batch constant Σ_m is given by

$$\Sigma_m = \left(\sum_{t=1}^{T} ||\xi_t^m|| \right)^{-1} \sum_{t=1}^{T} \sum_{i=1}^{I} \xi_t^{im} \tilde{\Sigma}_t^{mi} \tag{4.2.40}$$

where the cell-level measurement covariance matrix $\tilde{\Sigma}_t^{mi}$ is given by

$$\tilde{\Sigma}_t^{mi} = \frac{1}{h^i(\hat{x}_t^m, \hat{\Sigma}_t^m)} \int_{W_j^i} (x_t^m - y)(x_t^m - y)^T h(y|\hat{x}_t^m, \hat{\Sigma}_t^m) dY \tag{4.2.41}$$

Target state estimate X

The maximization of X under Quanta is significantly different than base HPMHT due to the deterministic target model. Σ is treated as a constant. The terms that do not involve X in (4.2.19) will not appear in the derivative and so we do not need to consider them here. The relevant part of (4.2.19) for each component is the term

$$\sum_{t=1}^{T} ||\xi_t^m|| (x_t^m - \tilde{y}_t^m)^T (\Sigma_t^m)^{-1} (x_t^m - \tilde{y}_t^m) \tag{4.2.42}$$

For simplicity, we continue the maximization of (4.2.42) under the assumption of a diagonal covariance matrix. This assumption is unnecessary and the general state estimation case is discussed in (Davey and Gaetjens, 2018). Combining the spread estimation with a state estimation that allows for a non-diagonal covariance matrix would yield an implementation that is well suited to dealing with non-symmetric targets that can rotate or otherwise deform in the sensor plane.

A diagonal covariance matrix allows us to maximize (4.2.42) with respect to each dimension $j = \{1, ..., D\}$ separately. Denoting the variance of the jth dimension for component m as $\sigma_{tj}^{m^2}$, (4.2.42) can be rewritten as

$$\sum_{t=1}^{T} \frac{||\xi_t^m||}{\sigma_{tj}^{m^2}} (x_t^m(j) - \tilde{y}_t^m(j))^2 \tag{4.2.43}$$

Recalling (4.2.1), we now maximize (4.2.43) by taking the derivative with respect to $x_{begin}^m(j)$ and $x_{end}^m(j)$ and setting the results equal to 0.

$$\begin{aligned} \frac{d\mathcal{Q}_x}{dx_{begin}^m(j)} &= \sum_{t=1}^{T} \frac{||\xi_t^m||}{\sigma_{tj}^{m^2}} (x_t^m(j) - \tilde{y}_t^m(j))(1-\phi) = 0 \\ \frac{d\mathcal{Q}_x}{dx_{end}^m(j)} &= \sum_{t=1}^{T} \frac{||\xi_t^m||}{\sigma_{tj}^{m^2}} (x_t^m(j) - \tilde{y}_t^m(j))\phi = 0 \end{aligned} \tag{4.2.44}$$

Subbing (4.2.1) into both parts of (4.2.44) and some simple manipulations produces the system

$$\left\{ \sum_{t=1}^{T} \frac{||\xi_t^m||}{\sigma_{tj}} \begin{bmatrix} (1-\phi_t)^2 & (1-\phi_t)\phi_t \\ (1-\phi_t)\phi_t & \phi_t^2 \end{bmatrix} \right\} \begin{bmatrix} x_{begin}^m(j) \\ x_{end}^m(j) \end{bmatrix} = \sum_{t=1}^{T} \frac{||\xi_t^m|| \tilde{y}_t^m(j)}{\sigma_{tj}} \begin{bmatrix} 1-\phi_t \\ \phi_t \end{bmatrix} \tag{4.2.45}$$

(4.2.45) must be solved for each dimension. A pseudo-inverse approach can be used to find a solution (Davey and Gaetjens, 2018).

Summary of EM algorithm steps

A summary of the steps for the Quanta algorithm with both extensions is given in Algorithm 1. The steps for the original Quanta algorithm are given in Section 10 of (Davey and Gaetjens, 2018).

Algorithm 1 Quanta MTT - Poisson Mixing Rates and Unknown Deterministic Spread

Inputs: T frames. Initial values for $\hat{x}_t^m$, $\hat{\lambda}_t^m$, and the Poisson hyperparameters for each target at each frame. Initial values for $\hat{\Sigma}^m$ for each target.

```
1: Compute $h^i(x_t^m, \Sigma_t^m)$ at all $I$ for each target $m$
2: For all $T$, $I$ and $M$ compute $f_t^i(x_t^m, \Sigma_t^m)$ (equation (4.2.34)) $\mu_t^{im}$ (equation (4.2.33)), $\xi_t^{im}$ (equation (4.2.32)), $\tilde{y}_t^{mi}$ (equation (4.2.16)), and $\tilde{\Sigma}_t^{mi}$ (equation (4.2.41))
3: For all $T$ and $M$ compute $y_t^m$ (equation (4.2.29))
   **Update Poisson mixing rates for each target at each timestep:**
4: Estimate the number of measurements $||N_t^m||$ by (4.2.37)
5: Predict and update gamma parameters by (4.2.38)
6: Compute the new $\lambda_t^m$ by (4.2.39)
   **Update state estimate for each target:**
7: Solve the system (4.2.45) using a pseudo-inverse for each dimension
   **Update spread estimate for each target:**
8: Compute the new $\Sigma_m$ by (4.2.40)
9: if Not converged then
10:    Set $\hat{x}_t^m$, $\hat{\lambda}_t^m$, $\hat{\Sigma}^m$ to the new estimates
11:    Return to step 1
12: else
13:    Break
14: end if
```

4.3 Results

The extensions integrated into Quanta attempt to augment the algorithm in different ways and so they are analyzed separately.

4.3.1 Poisson mixing rates

We follow a similar experimental setup to that from (Gaetjens *et al.*, 2017) to demonstrate that the benefits seen in HPMHT are also realized in Quanta by integrating the Poisson measurement model. The primary benefit of the Poisson model is the robustness of the SNR estimate to fluctuating target amplitudes. To simulate a fluctuating target, the instantaneous target amplitude is sampled as a random variable that follows a Swerling I model

$$A_t = -A \ln \left(u_t \right) \tag{4.3.1}$$

where A is the mean and u_t is a uniformly distributed random variable. The amplitude of the target is defined as the intensity of the pixel at the center of the targets Gaussian spread.

The peak SNR for the target is computed in dB according to

$$\text{SNR}_{peak,dB} = 20 \log_{10} A \tag{4.3.2}$$

The amplitude used to compute the peak SNR must be computed from the mixing parameters estimated by Quanta for both the Poisson and normalized mixing case. This can be done by first forming an estimate of the total number of shots associated with the target. In the Poisson mixing case, the total shot estimate is the same as the estimated Poisson parameter. In the normalized mixing case, the total shot estimate is equal to the estimated mixing proportion multiplied by the total shots in the image.

Given the estimated total target shots, the target amplitude can be computed as the product of the estimated total target shots with the probability mass of the pixel that the targets center is in. The later value is equivalent to the integral of the Gaussian distribution defined by the targets position and spread over the pixel that the targets center is in.

Before we can analyze the performance of Quanta with Poisson mixing rates, an important, but not entirely obvious aspect of what determines its performance must be discussed. Recall that the computation of the Poisson parameters for a timestep involves a previous estimate for the Poisson parameters. For Quanta, these are usually the estimated parameters from the previous algorithm iteration. However, this is not true for the oldest frame in the batch, since its prior is outside of the batch and thus is not affected by the estimation process. However incorrect the equivalent amplitude for these parameters is with respect to the true amplitude mean will have an effect on the performance of the algorithm. To analyze these effects, the equivalent amplitude for the prior frame for the oldest frame will be set to either 2 or 4 less than the truth, as noted on the proceeding figures. The initial equivalent amplitude for all frames in the batch will be set equal to the true fluctuating amplitude A_t.

Figure 4.1 shows the average peak SNR estimate at each frame for a 40 frame batch for a prior equivalent amplitude error of -2. Both the Poisson and basic Quanta algorithm are shown to underestimate the true peak SNR of the target. However, basic Quanta underestimates the peak SNR significantly more and its estimate is clearly inconsistent, unlike Quanta with the Poisson mixing rates. Figure 4.2 shows the -4 prior equivalent amplitude error case. Here we see that on average the Poisson Quanta estimate is slightly worse or the same on average. There are three factors to consider in light of this result. The first is that in light of the variance analysis to follow, it would still almost certainly be preferable to use the estimate from Poisson Quanta. The second is that we can see the SNR estimate for Poisson Quanta continued to increase as the batch progressed. Given that batch algorithms like Quanta are generally applied as sliding windows, we can expect that this larger error due to the lower quality initialization will disappear over a series of batches. The third is to consider that a forget factor of 7 was used to generate the data for these figures. This high forget factor causes a high degree of smoothing over time, which stops the algorithm from being able to correct for the initial error over the course of the batch. Figure 4.3 shows the same tests where the forget factor has been reduced to two. The lower forget factor allowed the SNR estimate to converge to a steady state that is even closer to the truth than was seen in 4.1 and is a clear improvement over the estimate from Basic Quanta.

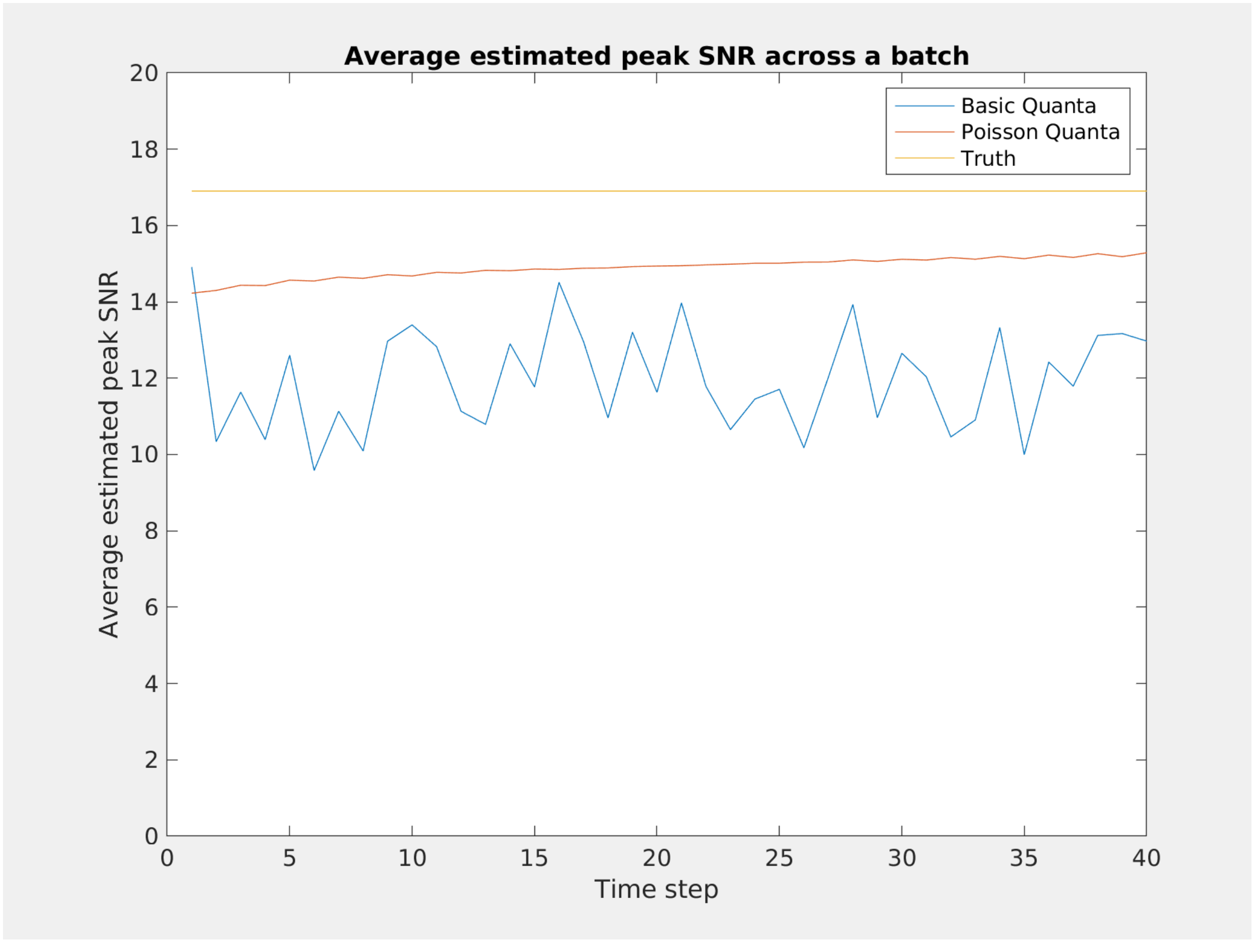

Figure 4.1: Average estimated peak SNR across a batch for a -2 prior equivalent amplitude error. 100 samples were used to form the average and the forget factor was set to 7.

Figure 4.4 shows the variance of the SNR estimate across all frames and all tests for each forget factor value. We see that the variance of the peak SNR estimate decreases exponentially as the forget factor is increased. This makes sense, given that previous Poisson parameter estimates are weighed more heavily in the current estimate as the forget factor is increased. That is, increasing the forget factor causes the estimate to be smoothed more over time. As we can see, this helps to eliminate much of the error introduced by the Swerling

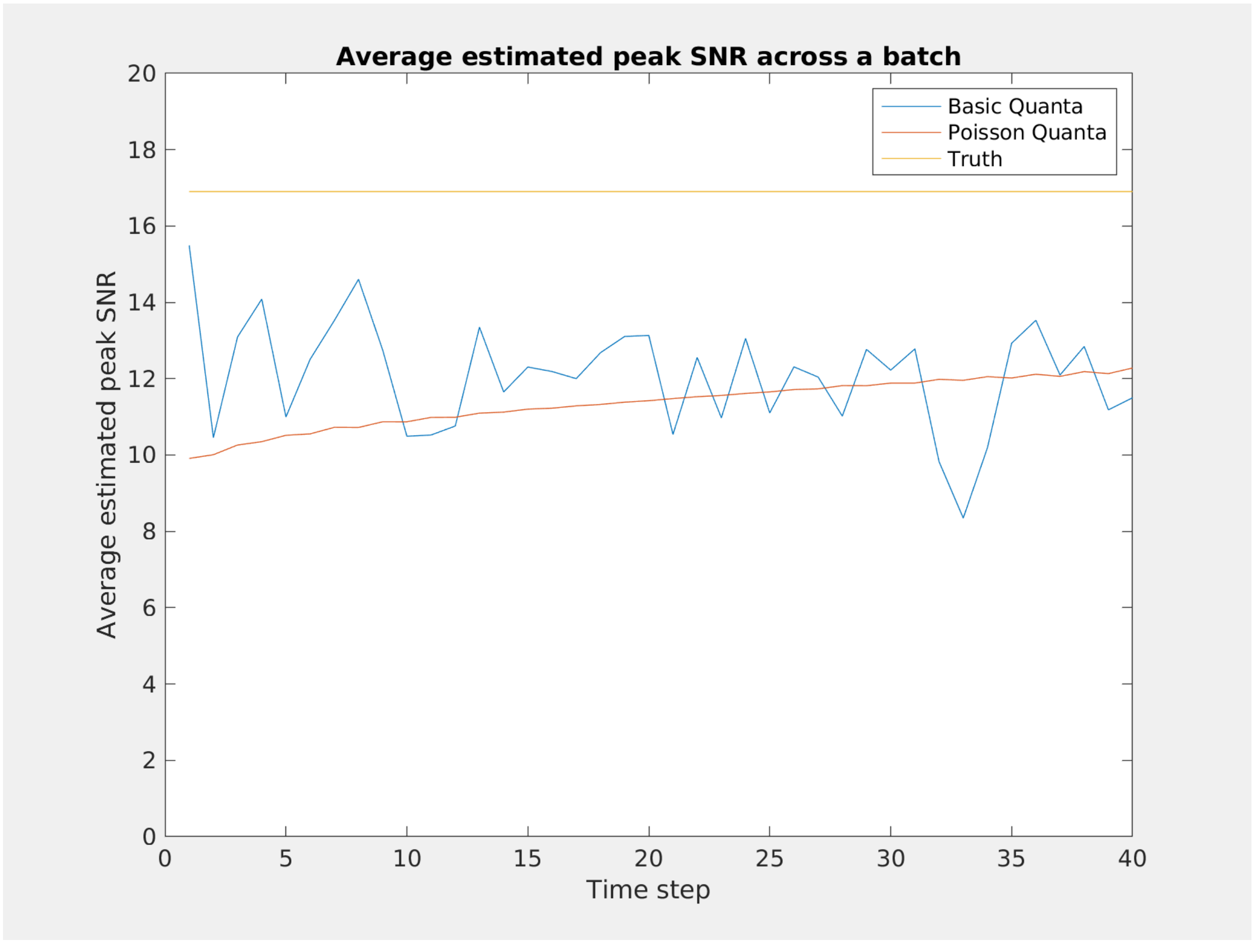

Figure 4.2: Average estimated peak SNR across a batch for a -4 prior equivalent amplitude error. 100 samples were used to form the average and the forget factor was set to 7

amplitude model. Note that the variance in the estimate for the basic Quanta algorithm in the -2 and -4 error case was 630 and 708 respectively. These results show that using the Poisson mixing model with a forget factor as low as 1 can reduce the variance in the SNR estimate by more than an order of magnitude. Using the mixing parameters to estimate the SNR of the target should thus be significantly more reliable and practical when the target amplitude can fluctuate if the Poisson mixing model is used.

4.3.2 Constant spread estimation

The goal with the experiments for the Constant spread estimation was to evaluate the performance of the estimation across a range of peak SNRs and spreads, which is not something

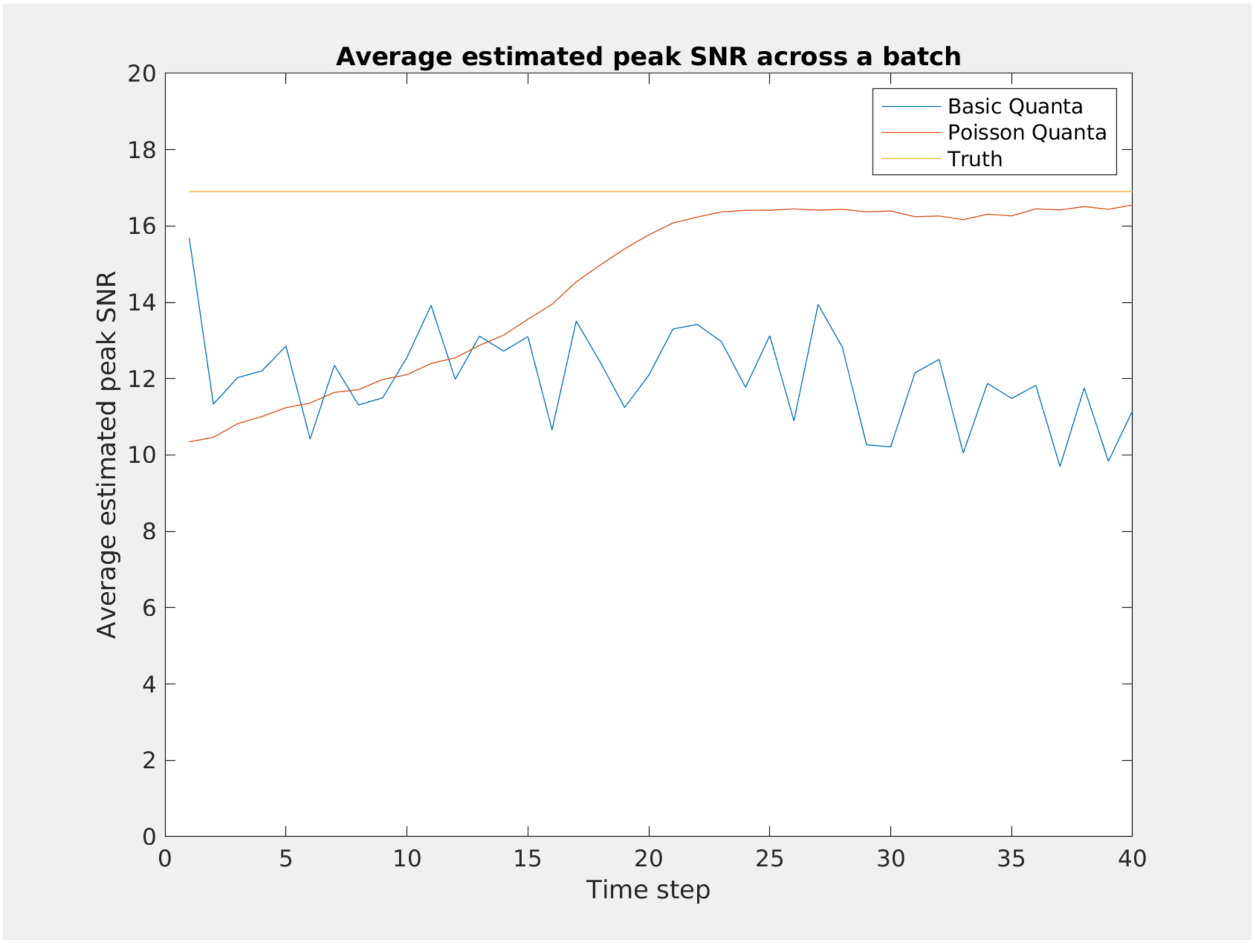

Figure 4.3: Average estimated peak SNR across a batch for a -4 prior equivalent amplitude error and a forget factor of 2. 100 samples were used to form the average

that has been done even in the HPMHT context to the best knowledge of the authors.

Figure 4.5 shows the results for an experiment that analyzed the average spread estimation error in terms of the standard deviation. Peak SNRs from approximately 0.8 to 17 were analyzed. Tracks were initialized at the true location in each frame in an attempt to purely analyze the capabilities of the spread estimation. The averages were formed by averaging the results across the batch of 40 frames and across 20 separate test executions. The algorithm was allowed to iterate 100 times, which was more than sufficient to achieve convergence given that the initial track positions were perfect. The true spread standard deviation was set to 4 for the X dimension and 6 for the Y dimension. The spread standard deviation for the tracks was initialized to half of the truth. That is, 2 for the X dimension and 3 for the

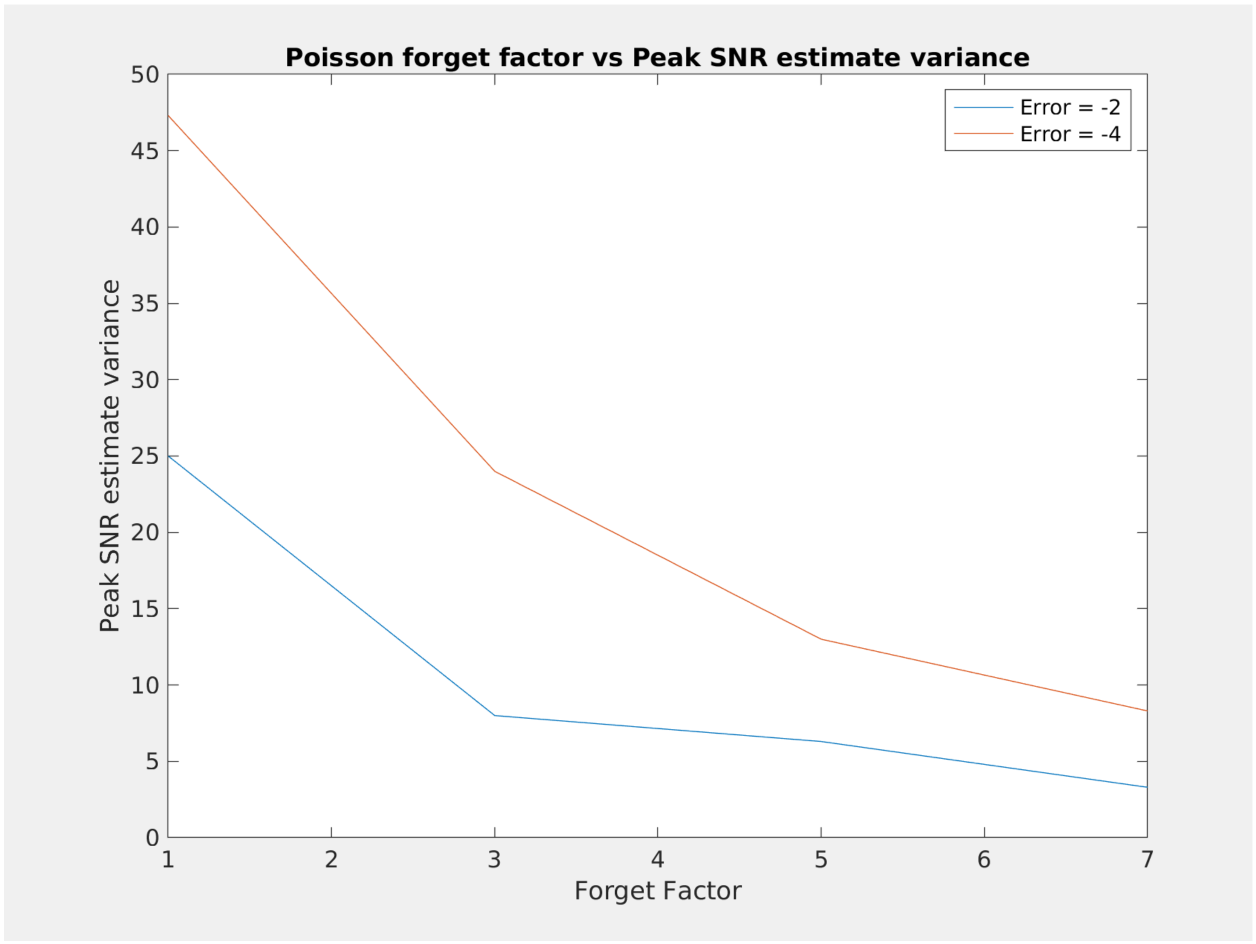

Figure 4.4: The effect of varying the Poisson forget factor on the SNR estimate variance

Y dimension. We see a mostly linear relationship between the Peak SNR and the spread estimation error for each dimension. The error in the Y dimension is consistently around 1.5 times higher than the X dimension which follows from the linear relationship given that the true standard deviation for the Y dimension is 1.5 times that of the X dimension. The results demonstrate that even for very low SNRs of 0.8, the spread estimation was still able to significantly improve upon the initialization on average, reducing the error by about one third. However, we see that even in the higher SNR cases, the estimation was not able to eliminate the error.

Figure 4.6 attempts to more generally analyze how the error in the estimation varies as both the target peak SNR and spread standard deviation are varied by analyzing the relative error in the estimation brought about for a given set of target parameters. Relative error

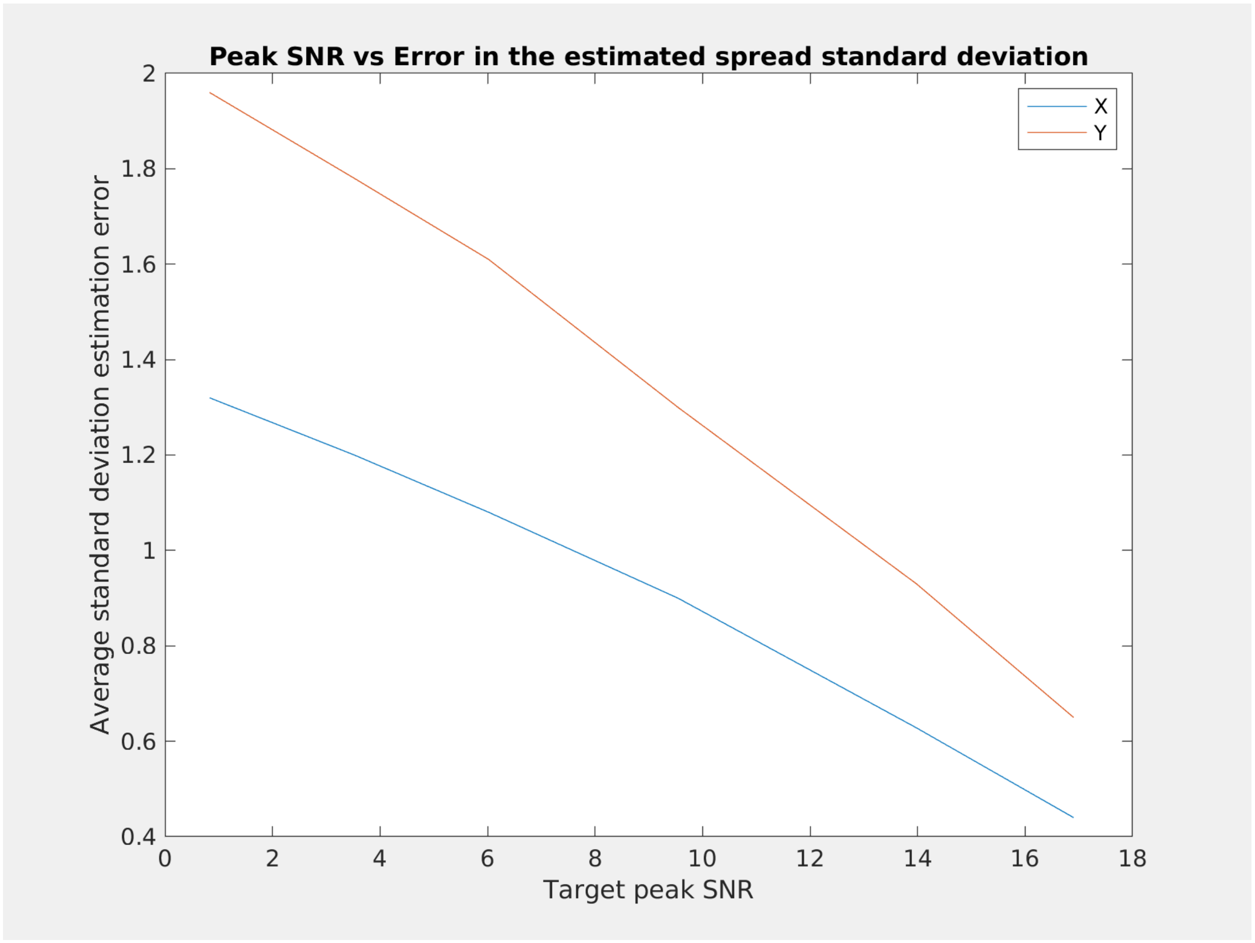

Figure 4.5: Relationship between target peak SNR and target spread standard deviation estimation error

was computed for the true σ_{true} and estimated $\sigma_{estimated}$ spread standard deviations as

$$\text{Relative Error} = 100 \cdot \frac{|\sigma_{true} - \sigma_{estimated}|}{\sigma_{true}} \tag{4.3.3}$$

For simplicity, the true X and Y dimension spread standard deviation were set to the same value for this experiment.

There are several interesting pieces of information that can be gleaned from Figure 4.6:

- The relative error in the spread estimation increases gradually as the SNR is varied
- For a given SNR, the relative error appears consistent for standard deviations of > 2

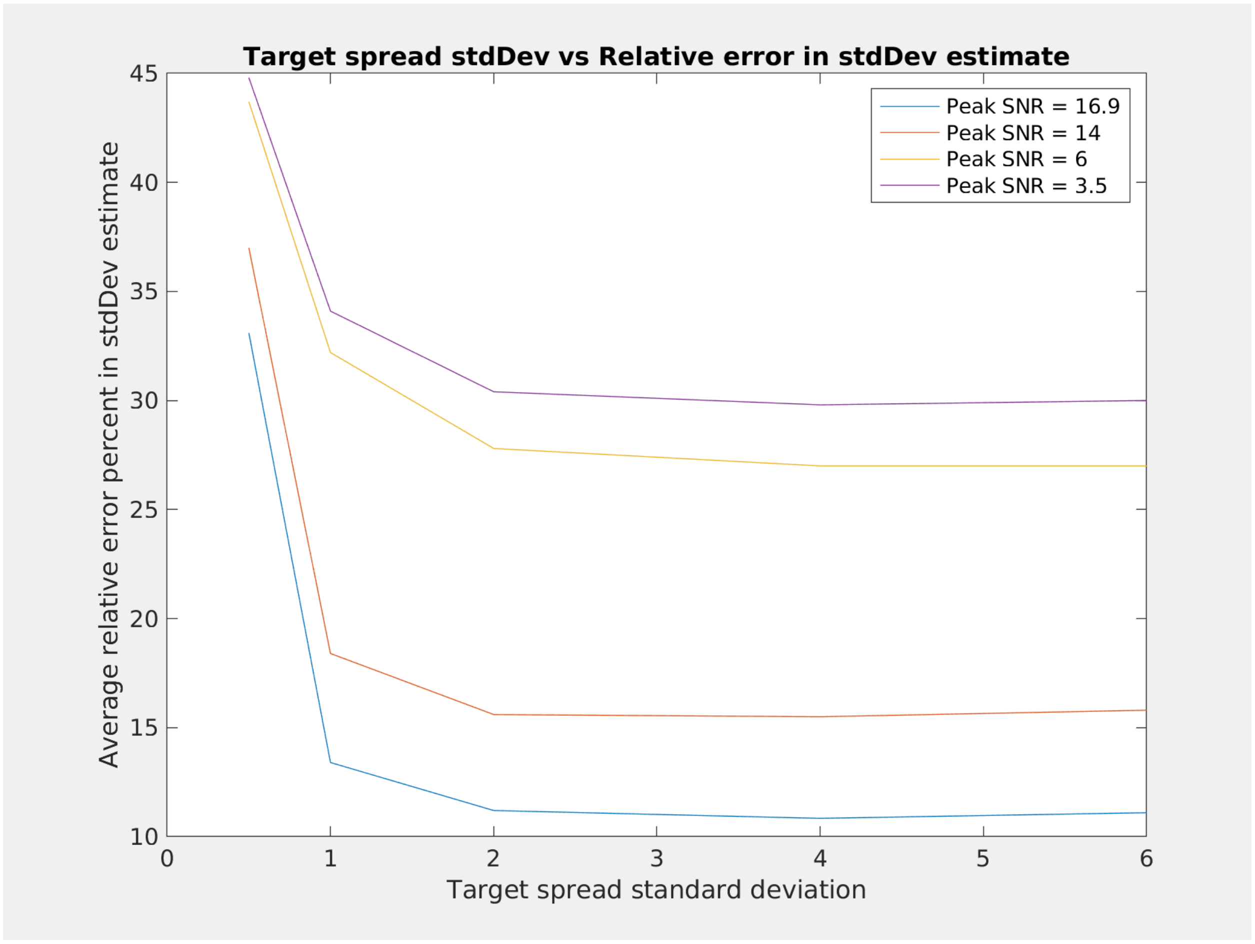

Figure 4.6: Relationship between target spread standard deviation and the relative error in the spread estimate

- Going from a standard deviation of 2 to 1 sees a marginal decrease in spread estimation performance, whereas below that the performance appears to drop off rapidly

The dropoff in performance that we see makes sense because the target size is approaching the sensor resolution. At that point, especially with noise, there is little information from which the true spread of the target can be estimated from. This is in contrast to targets with larger spread standard deviations, where the true size of the target is often plainly visible even to the naked eye, given that the SNR is sufficient.

Chapter 5

Conclusion

5.1 Analysis of Basic Quanta

The estimation performance of the basic Quanta algorithm was evaluated for one and two-point initialization for track positions. This evaluation took the form of a convergence analysis across various SNRs and target sizes via application of a generalized error definition. The iterations required to reach convergence in the one-point initialization case was found to likely be impractical for real application except potentially for scenarios where the targets have low velocity. The convergence was found to be radically faster with two-point initialization for the same parameters. The reasons for the faster convergence when applying two-point initialization were discussed. Implementation considerations aimed at minimizing the computational cost of Quanta for use in real applications were discussed.

5.2 Extending Quanta

The Poisson measurement model and deterministic spread estimation extensions were integrated into Quanta via a re-derivation of the algorithm. The Poisson measurement model extension was successful in significantly improving the quality and usability of the target SNR estimates that can be extracted from the parameters estimated by Quanta. The error in the spread estimation was analyzed for a variety of relevant SNR values and was shown to consistently break down in performance as the target size approached the sensor resolution.

5.3 Future Work

Future work will focus on developing a tracking framework around Quanta and applying this framework to real data. This work will explore creating a computationally efficient tracking framework by applying the parallelism exposed by the implementation strategies discussed in Section 3.4. The comparative performance of CPU and GPU implementations of the framework will be explored.

Bibliography

Bar-Shalom, Y., Daum, F., and Huang, J. (2009). The probabilistic data association filter. *IEEE Control Systems Magazine*, **29**(6), 82–100.

Blackman, S. S. (2004). Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, **19**(1), 5–18.

Blume, M. (2002). Expectation maximization: A gentle introduction.

Borman, S. (2004). The expectation maximization algorithm a short tutorial.

Davey, S. (2015). Efficient histogram pmht via single target chip processing. *Signal Processing Letters, IEEE*, **22**.

Davey, S. J. (2010). Detecting a small boat using histogram pmht.

Davey, S. J. and Gaetjens, H. X. (2018). *Track-Before-Detect Using Expectation Maximisation.*

Davey, S. J. and Rutten, M. G. (2007). A comparison of three algorithms for tracking dim targets. In *2007 Information, Decision and Control*, pages 342–347.

Davey, S. J., Rutten, M. G., and Cheung, B. (2008). A comparison of detection performance for several track-before-detect algorithms. In *2008 11th International Conference on Information Fusion*, pages 1–8.

Davey, S. J., Vu, H. X., Arulampalam, S., Fletcher, F., and Lim, C. . (2014). Clutter mapping for histogram pmht. In *2014 IEEE Workshop on Statistical Signal Processing (SSP)*, pages 153–156.

Dunham, D. T., Ogle, T. L., and Willett, P. K. (2019). Tracking very low snr targets with the quanta tracking algorithm. In *2019 IEEE Aerospace Conference*, pages 1–9.

Gaetjens, H. X., Davey, S. J., Arulampalam, S., Fletcher, F. K., and Lim, C. (2017). Histogram-pmht for fluctuating target models. *IET Radar, Sonar Navigation*, **11**(8), 1292–1301.

Granström, K. and Orguner, U. (2012). Estimation and maintenance of measurement rates for multiple extended target tracking. In *2012 15th International Conference on Information Fusion*, pages 2170–2176. IEEE.

Grünwald, P. (2004). A tutorial introduction to the minimum description length principle.

Konstantinova, P., Udvarev, A., and Semerdjiev, T. (2003). A study of a target tracking algorithm using global nearest neighbor approach. In *Proceedings of the 4th International Conference Conference on Computer Systems and Technologies: E-Learning*, CompSysTech '03, page 290–295, New York, NY, USA. Association for Computing Machinery.

Streit, R. and Lane, W. (2000). Tracking on intensity-modulated data streams.

Welch, G. and Bishop, G. (2006). An introduction to the kalman filter. *Proc. Siggraph Course*, **8**.

Wieneke, M. and Davey, S. (2014). Histogram-pmht for extended targets and target groups in images. *IEEE Transactions on Aerospace and Electronic Systems*, **50**(3), 2199–2217.

Willett, P., Balasingam, B., Dunham, D., and Ogle, T. (2013). Multiple target tracking from images using the maximum likelihood HPMHT. In O. E. Drummond and R. D.

Teichgraeber, editors, *Signal and Data Processing of Small Targets 2013*, volume 8857, pages 167 – 178. International Society for Optics and Photonics, SPIE.