Detection of Outliers in Navigation Sensor Measurements

Sasha Draganov
Expedition Technology, Inc
45195 Business Court, Suite 450
Dulles, VA 20166

Abstract- An outlier detection, usually called measurement editing, is commonly used by data fusion algorithms. In a typical implementation, a measurement is accompanied by an estimate for its standard deviation. If the measurement residual exceeds some multiple of standard deviations (e.g., 4), the editing algorithm rejects this measurement as an outlier. The standard approach does not provide any guidance for setting the threshold. A threshold that is too low rejects legitimate measurements, and the filter may get "stuck" in a wrong state. A threshold that is too high lets outliers in, affecting the quality of a solution. A modern navigation system integrates data from different sensors that have different error statistics, including the amount and the severity of outliers. A sensor-specific approach for treating outliers becomes a necessity.

For a Gaussian statistics, large residuals are exponentially rare, and outliers are not an issue. Unfortunately, the nature rarely follows Mr. Gauss; any hopes to salvage the situation by invoking the Central Limit Theorem are crushed by a Gaussian's extremely slow convergence at the tails. In practice, "fat tails" are quite common and are at the root cause of solution errors due to outliers.

In this paper, we present two new method of detecting and treating outliers. These methods are consistent with the general philosophy of optimal fusion: process only the data that is needed, with weights that accurately reflect data error statistics.

The first method uses Pickands - Balkema - de Haan (PBdH) theorem to detect fat tails. For any particular sensor, we pre-process large amount of data and estimate the statistics of the tail of the error distribution. We derived a formulation that translates the tail statistics into actionable outlier rejection algorithm and/or into a means for pre-processing measurements before they are fed into a navigation filter. In a simple case, the algorithm is similar to the conventional threshold for measurement editing; however, the magnitude of this threshold is now tailored to the statistics of measurement errors for the sensor in question. We processed real data from multiple navigation sensors to test this algorithm in practice. While some sensors are nearly outlier-free, others (e.g., magnetic compass) are not. The measurement editing threshold for such sensors is significantly lower; for example, for a magnetic compass the optimal threshold is only at approximately two standard deviations of the measurement noise.

The second method uses pattern recognition in the data to detect faulty measurements. At each time epoch, the algorithm processes recent measurements from a brief rolling window. The application of the algorithm includes a training stage, where multiple sets of measurements in the window are collected and categorized. After the training has been completed, the algorithm can detect an outlier at epoch N by looking at the pattern of measurements at epochs (N-n), (N-n+1), …, N. This algorithm was implemented and tested in real time. The results show reliable detection of outliers in a sensor with a small form factor and with limited computational resources.

Finally, we present an approach that integrates the above two outlier detection algorithms. While they may appear unrelated, there is a way to combine them in a mathematically sensible way.

This work was sponsored by DARPA (contract numbers FA8650-13-C-7320 and HR0011-15-C-0045). The views, opinions, and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Approved for public release. Distribution unlimited.

I. Introduction

A modern navigation system integrates data from different sensors that have different error statistics, including the amount and the severity of outliers. A sensor-specific approach for treating outliers becomes a necessity.

An outlier detection, usually called measurement editing, is commonly used by data fusion algorithms. In a typical implementation, a measurement is accompanied by an estimate for its standard deviation. If the measurement residual exceeds some multiple of standard deviations (e.g., 4σ), the editing algorithm rejects this measurement as an outlier. The standard approach does not provide any guidance for setting the threshold. A threshold that is too low rejects legitimate measurements, and the filter may get "stuck" in a wrong state. A threshold that is too high lets outliers in, affecting solution quality [1].

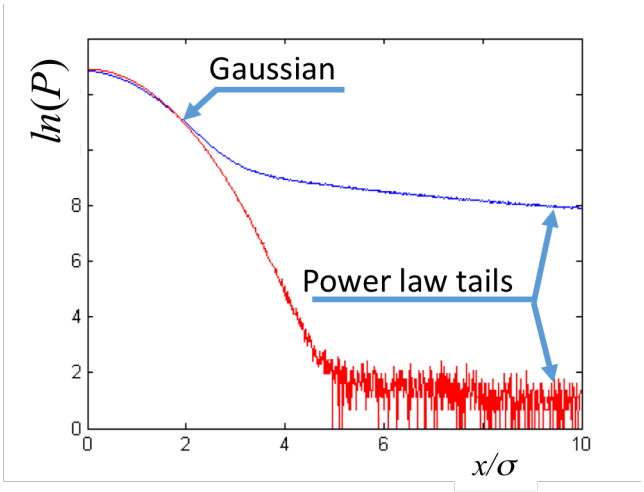

Figure 1. Examples of a Gaussian distribution with power law tails. Depending on the magnitude, the tail dominates the Gaussian at different values of x.

In a Kalman filter, measurement editing detects undeniable occurrences of outliers, but lacks a means of doing that in the "gray area". If a 4σ residual indicates an outlier, what can we say about a 3.8σ residual? Note that there are probably more data in the "gray area" than that well above a threshold, which exacerbates the problem. A similar challenge arises for a particle filter, where the filter algorithm can use the full measurement error statistics, including that for the tails, as long as this statistics is known. However, an unknown tail statistics can quickly lead to degeneration of the particle population.

In this paper, we present two algorithms for detecting outliers:

1. The first algorithm is based on recent advances in detecting and characterizing "fat tail" distributions, which helped the insurance industry to quantify the probability of rare, but catastrophic events using Pickands-Balkema-de Haan theorem [2]. To the best of our knowledge, the first use of this theory to navigation applications was presented in [3].
2. The second algorithm uses pattern recognition for a sliding window of measurements. The idea is to process a sufficiently large dataset, where outliers are identified (e.g., due to a comparison with a high-quality "truth" data) and to learn the temporal behavior of successive measurements. After the learning stage is complete, a system detects outliers by comparing the current sequence of measurements with the learned one. Previously, a similar approach was used in [4].

II. Outliers and the Distribution Tails

A. Tail Distributions in the Measurement Error Statistics

For a Gaussian statistics, large residuals are exponentially rare, and outliers are not an issue (this is discussed in more detail below). Unfortunately, the nature rarely follows Mr. Gauss; any hopes to salvage the situation by invoking the Central Limit Theorem are crushed by a Gaussian's extremely slow convergence at the tails. In practice, "fat tails" are quite common and are at the root cause of solution errors due to outliers. Fig. 1 shows two examples of a Gaussian distribution with power law tails that we obtained from a Monte-Carlo simulation. The pdf is plotted using the logarithmic scale; on this scale a plot of a Gaussian would look like a parabola. We can see that the hump of the distribution does follow a Gaussian, but a tail violates this pattern. Depending on the power law exponent and the relative frequency of outliers, the transition between the Gaussian portion and the tail occurs at different values: at approximately 2σ for the blue curve and at 5σ for the red. If we process two data streams with the two statistics shown in Fig. 1, we may need to set outlier editing thresholds differently for them.

We propose a new method of detecting and treating outliers, which is custom-tailored to the error statistics for each sensor. This method is consistent with the general philosophy of optimal fusion: process only the data that is needed, with weights that accurately reflect data error statistics. It is based on the Pickands - Balkema - de Haan (PBdH) theorem. Just as the Central Limit Theorem shows that the main hump converges to a Gaussian, PBdH shows that tails converge to the generalized Pareto distribution (GPD). We will use PBdH to monitor the measurements and to set outlier thresholds dynamically.

To illustrate the way the tail shape affects the outlier threshold, we consider a simple one-dimensional case. We assume that

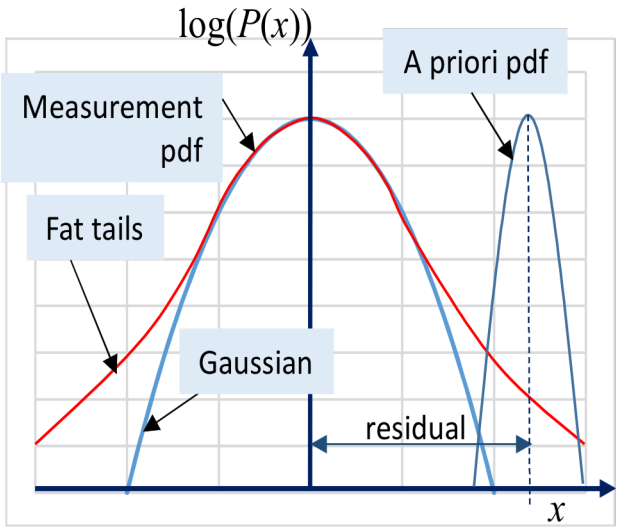

Figure 2. A priori pdf may lie on the tail of the measurement distribution.

the *a priori* state is Gaussian, and the measurement distribution function has a Gaussian main hump and a power law tail $P_{tail}(x) \propto (x - x_m)^{-\alpha}$. Note that for large deviations from the mean, a power law tail always "beats" the Gaussian exponent: the distribution function there is dominated by the tail.

For a joint probability, we compute the product of these two distribution functions. If the measurement residual is not large as compared to both standard deviations, the product of two distribution functions will have a Gaussian hump. This is the case of a normal processing, and it is a standard exercise that yields Kalman filter equations. However, if the residual is large, the Gaussian hump of the *a priori* distribution will be located on the power law tail of the measurement distribution function (Fig. 2). Then the posteriori likelihood is given by (the normalization constant is not shown for brevity):

$$P_p(x) \propto P_{tail}(x) \cdot P(x) \propto \frac{(x - x_m)^{-\alpha}}{2\pi\sigma} \exp\left[-\frac{(x - x_0)^2}{2\sigma^2}\right] \qquad (1)$$

The mean of this distribution can be computed via the Gamma function, but it is easier to compute the most likely value, with the result being almost the same:

$$x_p = x_0 + \frac{\Delta}{2} - \sqrt{\frac{\Delta^2}{4} - \alpha\sigma^2} \qquad (2)$$

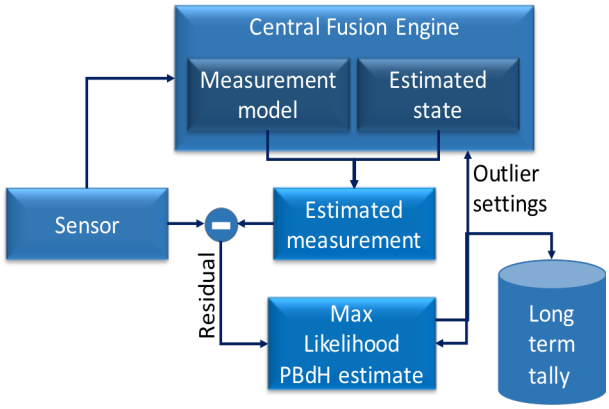

Figure 3. A block diagram of the outlier detection algorithm.

where $\Delta = x_m - x_0$ is the measurement residual. The key feature of this estimate is that if the measurement residual is large, then $\frac{\Delta^2}{4} >> \alpha\sigma^2$ and $x_p - x_0 = \frac{\Delta}{2} - \sqrt{\frac{\Delta^2}{4} - \alpha\sigma^2} \approx 0$. This means that in the case of fat tails, processing a measurement with a large enough residual does not update the filter state. We may as well achieve the same result by not processing this measurement at all. This is the true reason for measurement editing in a Kalman filter, which is usually applied as an *ad hoc* algorithm.

The reason for this rudimentary one-dimensional mathematical treatment is that it opens doors to an entirely new way of handling outliers. Measurements with small residuals should be processed by a conventional Kalman filter algorithm (or by its nonlinear/non-Gaussian extension). However, optimal processing of measurements with large residuals should use specific features of that sensor's error statistics. If the residual appears "large" (e.g., is equal to 4σ), but the error statistics is close to a Gaussian, this still may be a valid measurement to process by the filter (i.e, this measurement is not an outlier). However, if we are able to determine that for a particular measurement, the error statistics is dominated by a fat tail (e.g., power law), the effect of that measurement on the state must be reduced (though not necessarily completely dropped). In this scheme, outlier processing is optimally tailored to the true error statistics for each particular sensor.

Fortunately, recent advances in statistics produced new tools for estimating tail distributions. The gold standard is provided by the Pickands – Balkema – de Haan (PBdH) theorem. Just as the Central Limit Theorem shows that the main hump converges to a Gaussian, PBdH shows that tails converge to a Generalized Pareto Distribution. The Generalized Pareto Distribution (GPD) is given by

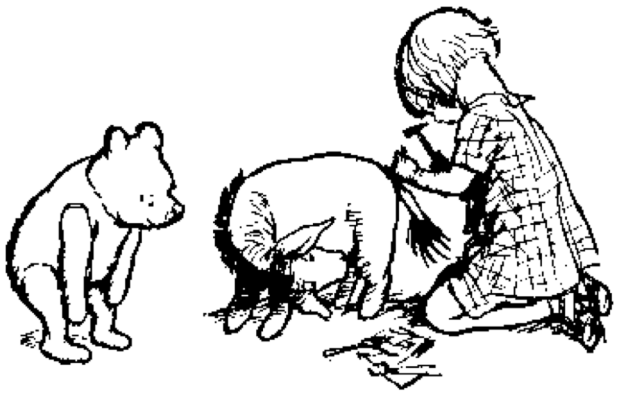

Figure 4. "Fond of [the tail]?" -- "Attached to it," said Winnie-the-Pooh sadly.

$$G(x,\xi,a) = 1 - \left(1 + \xi\frac{x}{a}\right)^{-1/\xi} \quad (3)$$

and has two parameters, a and ξ. Depending on the values of these parameters, GPD can model power and exponential tails. From a set of observed outliers, parameters a and ξ can be estimated by maximizing a likelihood. We will use PBdH to monitor the measurements and to set outlier thresholds dynamically.

By definition, measurements in the tail of distributions are rare, so data must be collected over a relatively long period. The algorithm keeps the tally of outliers that may extend beyond a single mission when necessary. A block diagram of this algorithm is shown in Fig. 3. Residuals are collected over time and processed by a cumulative least squares estimator. The estimator produces parameters of the Generalized Pareto Distribution for the tail. These parameters are supplied to the filtering algorithm that uses them to de-weight outliers or reject them outright.

B. Where is the tail attached?

The maximum likelihood estimation of GPD parameters is not sufficient by itself to be used it in navigation applications. To fill the gap to its practical use, we need to do two things:

1. Establish a way to select a threshold dynamically. Note that the GPD parameter estimation uses samples above a threshold, but how this threshold is selected in practice?
2. Develop an algorithm for rejecting or processing specific measurements, depending on the estimated probability distributions (both the main hump and the tail)

Here we can kill two birds with one stone. A threshold will serve two purposes: samples above it (1) are used for GPD tail estimation using maximum likelihood, and (2) are rejected from processing by the navigation filter. The idea is to find a value, where the "tail attaches to the hump" (Fig. 4).

We assume that the main hump has a Gaussian shape, and that its standard deviation is known (it is easy to estimate it in practice). The tail obeys the GPD distribution. To set the threshold, we seek the value, where the distribution starts to be dominated by the tail, rather than by the Gaussian main hump. It is surprisingly easy to show that this condition is given by:

$$\tilde{u} = \sigma\sqrt{2\ln\frac{a}{w\sigma\sqrt{2\pi}}} \quad (4)$$

where $\tilde{u}$ is the estimate for the threshold, a is the parameter of the GPD tail, and w is the fraction of the samples above the threshold.

III. Real Data Processing

A performance for real data is always the key test of any new algorithm. We used the data, which were originally collected for the DARPA's ASPN program [5]. There are several datasets available, each with multiple sensors; we selected a dataset for a driving scenario in a city. We used an odometer and a magnetic compass sensor for measurement updates and IMU data for the time update. If the initial user position is known, and if the altitude of the driving user is confined to the terrain, then heading (from the compass) and distance travelled

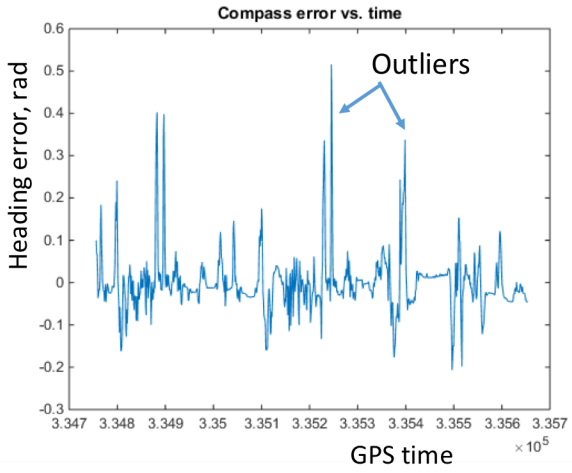

Figure 5. Time series plot of the heading measurement error showing outliers in the data

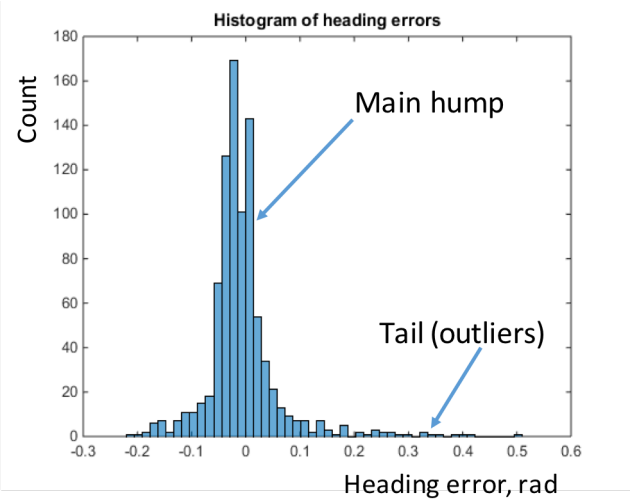

Figure 6. A histogram of heading measurement errors.

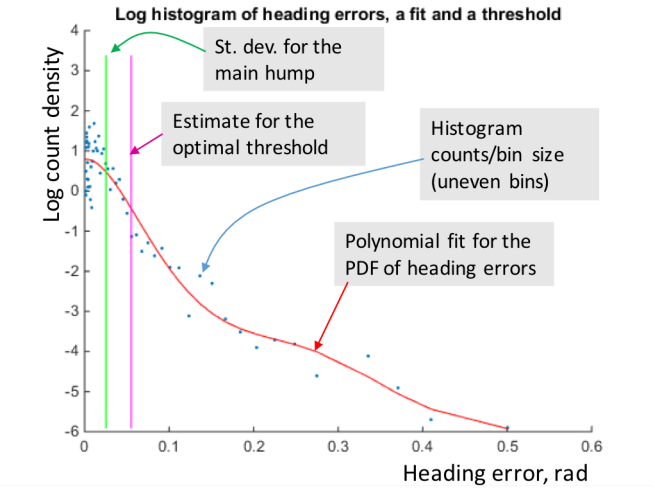

Figure 7. Log histogram of heading measurement errors. The main hump has a parabolic shape near the maximum. The optimal threshold is computed from a least square fit to a GPD

(from the odometer) is sufficient to estimate the trajectory. This computation is hindered by outliers in the magnetic compass measurements, which are relatively common. Fig. 5 shows measurement errors with respect to the true heading (which is also available in the data from a set of highly accurate sensors). We can see, that outliers are common in the data; histogram of these errors is shown in Fig. 6. Fig. 7 shows the same histogram on the log scale (red curve); while most measurements in the main hump have errors of few hundredth of a radian, there are outliers, with errors an order of magnitude higher. It is precisely the outlier editing threshold for the magnetic compass measurements, which we estimated in this test. For reference, we show the standard deviation of the main hump of the distribution function (a vertical green line). The PBdH tail estimation and the subsequent "tail attachment" point computation (see Equation (4)) put the estimated threshold at only 2.19 multiple of the standard deviation for the main hump of the distribution function (magenta). This is less than multiplier values that are commonly used in filtering (e.g., 4). Note that we compute the standard deviation for the main hump only, defined as the point, where the pdf decreases by a factor of $exp(-1/2)$; this effectively cuts off the effect of outliers on the computation of standard deviation. Had we computed the standard deviation for the entire set (including the tail), the standard deviation would be larger, and the optimal outlier threshold would be at even lower multiplier value.

The navigation solution algorithm is as follows: at each epoch, we compare the IMU-propagated heading with the current magnetic compass measurement. If the residual is below the threshold, the compass measurement is processed and the solution is updated. The solution is advanced using heading and odometer measurements. However, if the residual is above the threshold, the compass measurement is edited (dropped) and the heading is set to the value, which is predicted by IMU propagation.

This algorithm creates a tradeoff for setting an outlier editing threshold. If a threshold is set low, relatively many compass measurements are rejected as outliers; therefore, the trajectory solution will rely more on IMU propagation, which is prone to error accumulation. On the other hand, if the threshold is set high, some outliers may trickle through to filter processing, corrupting the solution. Intuitively, there should be an optimal outlier rejection threshold, which achieves the best accuracy.

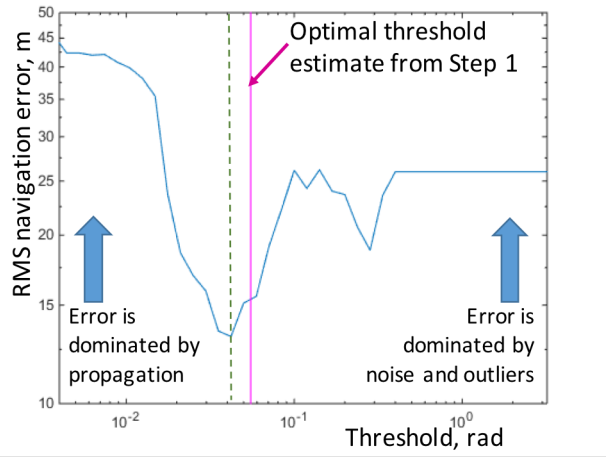

Figure 8. The navigation error as a function of outlier threshold. The optimal threshold corresponds to a lower error.

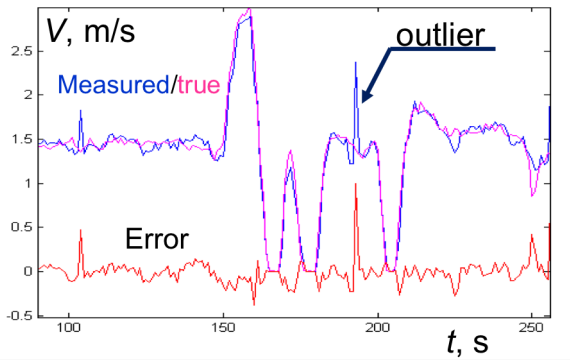

Figure 9. Walking speed measurements containing an outlier.

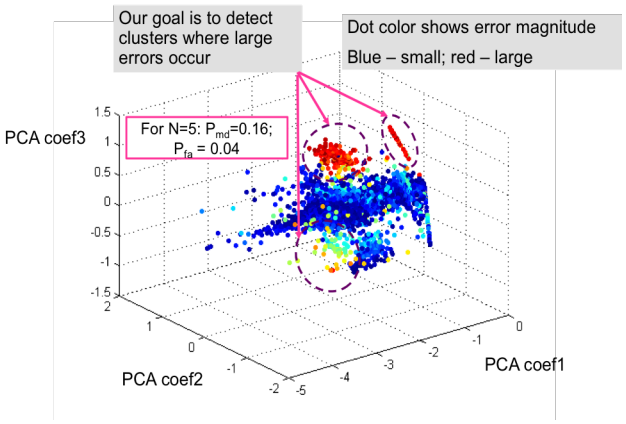

Figure 10. Outlier clustering in the PCA space

Indeed, as we ran the trajectory estimation algorithm multiple times using a range of outlier thresholds, the tradeoff between small and large thresholds becomes evident. Fig. 8 shows the RMS navigation error vs. magnetic compass threshold (blue curve). There is a clear minimum where the error is not dominated by propagation, nor by outliers. This minimum is shown by a dashed green line. The estimate for the optimal threshold from our algorithm is shown as a magenta line (same as in Fig. (7)). They do not coincide, but are still close in value. We can see that our algorithm estimates an outlier threshold, which allows to cut off most outliers, preserve most legitimate measurements and thus achieve a better navigation performance.

IV. Pattern Recognition Approach

The basic idea for the second approach is to train a pattern recognition algorithm using large amounts of data where true velocity measurements are available. After training the algorithm, it should be able to identify similar patterns in the future. In essence, the algorithm assumes that there are certain patterns in the legitimate data, and a drastic departure from these patterns may indicate an outlier. For any epoch, the algorithm processes several previous epochs to estimate the probability of the current measurement to be an outlier. If the probability exceeds a preset threshold value, the algorithm generates an outlier flag.

To train the algorithm, we used approximately 24,000 epochs of data from a sensor that measures the speed of a walking user. Fig. 9 shows a sample time series of measurements that contains an outlier. (The details about the design and operations of this sensor are outside of scope of this paper.) Data were collected at several user speeds, from standing still to running. Data were split into rolling window segments with N_w measurements in each segment. Thus, each segment is a point in a space with N_w dimensions. These points were processed using the Principal Component Analysis, and we retained $n_w \leq N_w$ principal components in the data. The n_w-space was split into bins, and for each bin we tallied the number of valid and invalid (outlier) measurements using the truth measurements. Based on this analysis, bins were categorized as "valid" or "outlier". This completes the training portion of the algorithm development.

Outlier detection becomes a matter of a lookup for a bin. Specifically, during subsequent data collects, we apply the algorithm to every measurement. We form a vector of N_w most recent measurements (including the current one), and use PCA components to find this vector's projection on the n_w space. We determine which bin this vector is in, and look at the fraction of invalid measurements in that bin from the training portion. This fraction is the estimated probability of an outlier. Indeed, if in the training data all similar segments ended with very few outliers, one can expect that the segment in question ends with a valid measurement, and vice versa.

Fig. 10 shows an example of data from a limited number of measurements for $n_w = 3; N_w = 5$. (Here, we selected 3 PCA components to be able to plot the data. The amount of data shown is limited to reduce clutter in the figure.) Each dot corresponds to a data segment. The dot color shows the error in the last measurement of the segment as determined from the truth data; blue corresponds to low errors, and red corresponds to large errors (outliers). It is clear that red dots cluster in three regions, shown by dashed ellipses. When the algorithm is applied to detect outliers, it computes a point in this PCA space for each measurement. As long as this point does not fall into one of the "outlier" regions, the measurement is deemed valid and vice versa.

We varied values of n_w and N_w to achieve best performance. By trial and error, we arrived at the values $n_w = N_w = 5$. Another knob to turn was the bin size. Fig. 11 shows results for optimal values. We plot the true probability of an outlier vs. an estimate of the same. For an ideally performing algorithm, these two values would be identical, and the plot would be a straight line at 45°. The actual performance reasonably well approximates the ideal one.

V. Marrying the two Algorithms

Above, we presented two algorithms on detecting outliers in navigation sensor measurements. While these algorithms may appear unrelated, it is best to consider them complementary. The optimal algorithm may be a combination of the two, which exploits the best features of both. Specifically, the sliding data window in the second algorithm forms a multidimensional space, and outliers occupy one or more region in that space. Separately, outliers are expected to correspond to large residuals. It is logical to design an outlier detection scheme that combines these two criteria.

VI. Conclusion

The basic philosophy of data processing is: "the more data, the better result". In practice, this is correct only if the data are processed with a full knowledge of their statistics. Moreover, some of the data have marginal value and may be skipped without a noticeable effect on the result. Outliers are a good illustration of this thesis:

1. If processed without being identified as outliers, they can damage the results
2. If processed while identified as outliers, their impact on the result is small and often can be neglected.

This establishes a trade-off between processing more data and excluding outliers. Since outlier statistics is sensor-dependent, it is clear that outlier detection and exclusion should be tailored to the data source. We present two algorithms to learn the outlier statistics and features from the data and apply it to processing. One such algorithm estimates GPD parameters for the tails of the measurement error distributions; the second one applies a very simple machine learning technique for detecting outliers from a recent measurement history.

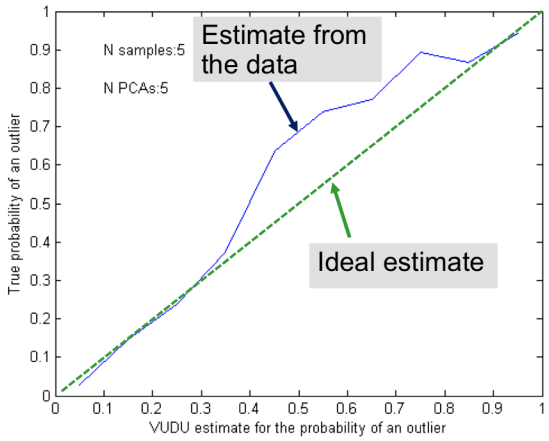

Figure 11. Probability that a particular measurement is an outlier: estimate vs. true

For the first algorithm, a real data test confirms that overall results for the navigation accuracy are improved when applying a custom-set outlier threshold. The theoretic criterion for setting the optimal threshold uses estimated GPD parameters and the "tail attachment point". This step can be performed at the sensor calibration stage. We performed a parametric study to detect an optimal threshold for outlier processing; this study was made possible due to a "true" data set that was available for testing algorithm performance. Application of the estimated optimal threshold in a real data test shows that navigation accuracy is improved and that the estimated optimal threshold is close to the true value.

The second algorithm successfully detected most outliers in the speed data. The test was performed in real time, on a device with a small size, weight and power.

These algorithms are not limited to navigation and are likely applicable to other applications where Kalman or similar filters (particle filters, trackers, etc.) are used.

VII. Acknowledgments

The pattern recognition algorithm was developed with invaluable help from a team at the Boeing Company. The implementation of this algorithm on a sensor and the data collection by this sensor was performed by Boeing engineers.

VIII. References

1. Veth, M.J., *Effects of Non-Gaussian and Non-White Noise Processes on Image-Based Targeting for Mission-Critical Applications*, Proceedings of the 26th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2013), Nashville, TN, September 2013, pp. 1988-1995.
2. D. Sornette, *Critical Phenomena in Natural Sciences*, Springer, 2006, 539 pp.
3. P. B. Ober, D. Imparato, S. Verhagen, C. Tiberius, H. Veerman, A. Van Kleef, F. Wokke, A. Bos, and A. Mieremet, *Empirical Integrity Verification of GNSS and SBAS Based on the Extreme Value Theory*, Navigation, v. 61, No. 1, pp. 23-38.
4. Veth, M.J., Soloviev, A., Yang, C., Taylor, C., Robustified, *Multi-epoch Stochastic Constraints for Outlier Detection and Removal in Online Multi-sensor Inertial Navigation Systems,* Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014), Tampa, Florida, September 2014, pp. 2212-2219.
5. http://www.wired.com/2012/06/darpa-gps