Pre-Processing Input Data to Augment Fault Tolerance in Space Applications

Jayakrishnan Nair, Zahava Koren, Israel Koren and C. Mani Krishna

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst 01003
E-mail: {jnair,zkoren,koren,krishna}@ecs.umass.edu

Abstract

Today's advanced science applications often process huge volumes of data in real-time under extreme ambiances such as in spacecrafts and surveillance equipment. Faults manifesting in this raw data may corrupt the input given to these applications, causing flawed results. Hence we show that highly significant improvements in application reliability and precision can be obtained if the data is proactively preprocessed, using statistical analysis of input for possible recovery from the bit errors at input. In this paper we implement application-specific algorithms for preprocessing the input datasets for two major space applications. The Next-Generation Space Telescope (NGST) and the Orbital Thermal Imaging Spectrometer (OTIS) are both from NASA's REE suite, and yet have sufficient contrasts in their datasets to demonstrate the versatile range of applications for which our approach is suitable. Our preprocessing algorithms utilize application-semantics, inherent data redundancy, absolute natural bounds, and temporal and/or spatial locality coherence in the natural data for setting dynamic thresholds to identify data-faults. We have compared the performance gain in relation to an ideal system for these algorithms with two standard algorithms used for image smoothing in graphics applications.

1 Introduction

Contemporary science applications, especially those designed for astronomical, geological and surveillance purposes, often involve real-time[1] processing of huge volumes of data obtained from detectors and sensors. With the rapid advances in architectural computing potential and data storage, it has become possible to enhance manifold the resolution at which such applications process their input, resulting in the exponential escalation of the input data volume. As a majority of these applications run on severe ambiances such as onboard spacecrafts and surveillance equipment, these systems are often exposed to extreme conditions that make them vulnerable to many forms of faults, despite the best of protection mechanisms like radiation-hardening. Hardware failure is one such form, which may be tolerated using hardware and software redundancy schemes[1, 2], of which the former is often prohibitively expensive.

Some of the prominent software-redundancy schemes in practice include the Algorithm-Based Fault Tolerance [3], which is particularly suited for matrix operations, and the N-Version Programming [4] for which many voting schemes such as the T/(N-1) VP scheme have been proposed. If the fault occurs during execution, causing an abnormal process termination or an invalid output, the Application-Level Fault Tolerance (ALFT) scheme[5] would replace the primary faulty (or non-existent) output with a partial output produced by a scaled-down secondary run on another node in a distributed system.

Another very significant type of fault occurs when the memory that holds the gigantic volumes of input data gets affected with random bitflips, the relative probability of which is high considering the exorbitant proportion of data to instruction memory sizes, often as high as 10000:1 as in some of the applications we considered. The aforementioned schemes clearly do not handle this fault-model, as there are no process failures here, and also a recomputed or secondary output may only be expected to produce equally spurious or worse results than the primary as the corrupted input affects both, while also incurring a substantial overhead. Thus, though the current schemes have proven useful in recovering from system faults occurring in the instruction memory or processing units of the system, they are inadequate in handling the faults occurring at source, during transit from source, or inside the data memory that induce the input raw data to be corrupted. Hence the ideal solution for this fault model is to identify and correct the bitflips in the raw data before it is given to the application for processing. However, in the absence of any error-correcting codes inbuilt into the source or the application, a bit-wise sanity analysis of the input data becomes imperative. In this paper, we attempt to deal with this fault-model by devising a scheme that does preprocessing of the input datasets using algo-

rithms that utilize application-specific semantics and inherent input redundancy. Such algorithms can use generic concepts discussed in this paper as the backbone, but may have to apply application-dependent fine-tuning to yield optimum results. We have studied this approach using two distributed real-time applications of high contemporary relevance from NASA´s Remote Exploration and Experimentation (REE) [6] suite.

2 The NGST Application Benchmark

The first application is NASA´s Next Generation Space Telescope (NGST) [7] Data Processing Application, which is designed to run onboard the NGST, a large-aperture passively cooled infra-red telescope which is to be launched into the L2-Halo Orbit by 2010 to succeed the Hubble´s Space Telescope, as a powerful observatory to serve as the pivot in the ORIGINS program [8]. Since such a distant orbit puts the NGST beyond the protective influence of any planetary magnetic field, it would be exposed to severe cosmic radiation, which can lead to significant data loss and can also induce cross-talk between the CCD (Charge-Coupled Device) sensors in the detector [9]. When a cosmic ray (CR) hits the detector, false-triggering may happen, and hence it is anticipated that during a baseline 1000-second exposure, an unacceptably high 10% data loss would occur [10], besides reducing the data compression ratio by about 12%. This necessitates having multiple readouts per baseline, leading to high redundancy in the raw data, in order to digitally analyze image data [11] using comparison and integration to obtain one image per baseline to be downlinked to the base station, after compression using Rice Algorithm [12]. Due to the limited downlink bandwidth constraints, this processing has to be done onboard the NGST, for which many Cosmic Ray Rejection Algorithms [10, 11, 12] have been proposed.

While cosmic rays cause the data read at the detector to be irrecoverably lost, bitflips that occur in legitimately read data while in storage or transit before being processed by the CR-rejection algorithm, can cause a further degradation in precision at the output beamed to earth. Since precision and input reliability are at a premium for the myriad scientific experiments [13] conducted on the data obtained from the NGST, the importance of reproducing data as close to the ideal (fault-free and CR-free) situation is evident.

2.1 The System Architecture

The CR-rejection process onboard the NGST is designed as a real time distributed system [10, 11, 12], which has been estimated by STSci as a 16-processor workstation interconnected with a high speed network such as the Myrinet, using Commercial-Off-The-Shelf (COTS) processors. The slack CPU time in the slave nodes can be very well utilized for a suitable fault-tolerance scheme, for which the application layer of NGST offers ample scope. The detector has a 1024*1024 sensor array, and all the input images of this resolution are fragmented into 128*128 pixel image segments and handed down to the slaves for processing as shown in Figure 1. The processed fragments are returned to the master for re-integration and compression, before being transmitted to the base station on earth.

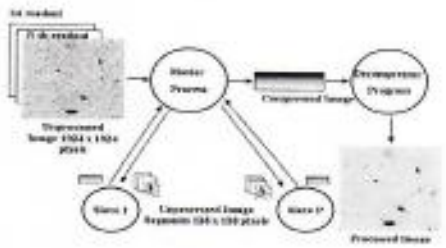

Figure 1. The System Architecture for CR-rejection onboard the NGST

2.2 Analytical Models

2.2.1. The Input Data Format and Analytical Model. The inputs to NGST, constituted by the multiple readouts of the detector within each baseline of 1000-seconds, are stored as FITS images (Flexible Image Transport System [14]) consisting of one or more Header + Data Units (HDUs). Though the FITS header consumes only a small proportion of the file size, header integrity is vital as the master and slave nodes decode it for interpreting the bytes found in the corresponding data unit. Hence, a data-fault caused by a bitflip occurring in the header region of a FITS file has the potential to cause catastrophic failures. For example, if keywords such as NAXIS or BITPIX are misinterpreted at the node, the dimensions of the data array or the bit resolution of the pixels may not be known respectively, resulting in corrupting the entire data unit.

The temporal redundancy in the input comes from the multiple (N, = 64 or 65 depending on the chosen scheme [10, 11, 12, 13] images, sampled within short intervals, in each dataset during its baseline. Hence for each coordinate Π in the dataset, there are N *pristine* temporal variants $\{ \Pi(i),\ i = 1 \ldots N \}$ in an *ideal system*, such that these variants exhibit strong temporal correlation for which we assumed a Gaussian statistical

model as follows:

$$\Pi(i+1) = \Pi(i) + \Theta_i, \quad i = 1 \ldots N-1 \qquad (1)$$

where Θ_i is a Gaussian Random Variable, with mean $\mu = 0$ to allow for unbiased positive and negative variations and standard deviation σ representative of the simulated datasets from the NGST Mission Simulator (NMS). We consider two models for bitflips:

2.2.2. The Uncorrelated Fault Model. This model assumes that bitflips occur randomly in the input datasets based on a given static probability Γ_0, either at source, during transit from source to the system, or while residing in memory. The N 16-bit temporal variants after bitflips at the same coordinates in the image are denoted by $P(i)$, $i = 1 \ldots N$.

2.2.3. The Correlated Fault Model. When bitflips occur in memory, due to multifarious reasons such as alpha particles striking the memory region onboard a spacecraft that is exposed to hazardous radiation, or heavy polarization due to free-moving charged particles causing capacitance effects in memory chips, or power glitches causing bit errors in memory, the pattern of bitflips occurring in such affected memory is often concentrated around the worst-hit center, with edges of the affected region siphoning off in all directions in the memory organization. This could lead to such complications as a sequence of flipped bits voting to flip a good bit causing a false alarm, and hence posed a very interesting problem to explore. We, therefore, consider the following model for bitflips:

- The probability of each bit being flipped must depend on the number of consecutive flips (run) that have preceded in the horizontal and vertical dimensions, which increases with the length of run.
- The probability is computed in both directions and the higher of the two, corresponding to the direction having a longer run at that bit position, is taken.

In the model we used, if the bit at position ω, is preceded by a run of R bitflips (in the direction being considered), then the probability $\Gamma_{corr}(\omega)$ for that bit to be flipped is

$$\Gamma_{corr}(\omega) = \sum_{j=1}^{R} (\Gamma_{ini})^j \qquad (2)$$

where Γ_{ini} is the base probability with which a fresh run initiates. For infinite runs, the RHS converges to an infinite geometric series: $\frac{\Gamma_{ini}}{1-\Gamma_{ini}}$, which will always be less than 1 for any Γ_{ini} less than 0.5.

3 A Dynamic Preprocessing Algorithm

3.1 Concept Behind Bit Windows

The 16-bit representation of a potentially damaged pixel $P(i)$ can be partitioned into three bit windows A, B and C in a temporal locality. Window A is the first from left, comprising of the most significant bits that are the least likely to change naturally across a dataset and hence exhibit very strong bitwise correlation with its close temporal variants, after taking carry propagation effects into consideration. Window C comprises of the least significant bits that may keep naturally changing with every pixel and hence can be ignored from preprocessing as it is impossible to identify the bitflips here. The middle window B usually determines how well a correction algorithm improves the reliability of the raw data. The binary weights of the bits in this window are too considerable to be ignored as in C, although the bits are not as consistent as in A. Statistically, the best results out of this region are obtained by using a sliding model for temporal locality. The underlying concept is that even though two arbitrarily distant $P(i)$ may not have sufficient coherence in this region, *consecutive* variants in a sliding window can be expected to show much better correlation In other words, Window B is analogous to A if the region of locality being considered has a sufficiently smaller number of variants than N.

3.2 Sensitivity Parameter for the Preprocessing Algorithm

A good fault tolerance scheme needs to be scalable depending on the susceptibility to faults and the trade-off with overhead in execution time and associated power consumption. We introduced the input parameter "sensitivity" (Λ) that can be set from 0 to 100 depending on the design requirements. At null sensitivity the algorithm does nothing but a simple sanity analysis of the FITS header and hence has negligible overhead compared to the NGST application´s execution time. Λ has a direct bearing on the width of window B, which needs maximum computational effort.

3.3 An Implementation Perspective

The main advantage of our algorithm compared to the generic algorithms discussed in [Section 4],is that it is entirely dynamic in its criteria for identification of faulty pixels, and automatically adjusts the filtering parameters to suit the coherence in the region of data being operated upon. It sets tighter bounds for regions in the datasets that show little variation over space and time, as compared to very turbulent regions. It achieves

Algorithm 1 The Preprocessing Algorithm "Algo_NGST"

Require: The number of neighbors Υ each pixel must consult for temporal conformance, i.e. $\frac{\Upsilon}{2}$ neighbors each in the forward and reverse directions.

Require: The Sensitivity parameter Λ, which scales the algorithm to suit the environment requirements.

/* Obtain the Voter Matrix V having Υ sets $\{V_z, \forall z \in [1 \ldots \Upsilon]\}$ of $\frac{N}{2}$ elements each, such that $\forall \alpha \in [1 \ldots \frac{\Upsilon}{2}]$ the sets $V_{(2\alpha-1)}$ and $V_{(2\alpha)}$ uniquely contain the bit-incongruences (found by XOR) of all possible pairings of pixels $\{P(i)\ \forall i \in [1 \ldots N]\}$ that are at most $\lceil \frac{\Upsilon}{2} \rceil$ time variants apart */

for $\alpha = 1 \rightarrow \frac{\Upsilon}{2}$ **do**
  if α is odd **then**
    for all $i \in [1 \ldots \lfloor \frac{N-\alpha}{2} \rfloor]$ **do**
      $V_{(2\alpha-1)}(i) = P(2i)$ XOR $P(2i + \alpha)$
    end for
    for all $i \in [\lceil \frac{\alpha}{2} \rceil \ldots N]$ **do**
      $V_{(2\alpha)x}(i) = P(2i)$ XOR $P(2i - \alpha)$
    end for
  else
    for all $i \in [\lfloor (\frac{\alpha}{2} + 1) \rfloor \ldots \lfloor \frac{(N-\frac{\alpha}{2})}{2} \rfloor]$ **do**
      $V_{(2\alpha-1)}(i) = P(2i - \frac{\alpha}{2})$ XOR $P(2i + \frac{\alpha}{2})$
    end for
    for all $i \in [(\frac{\alpha}{2} + 1) \ldots N]$ **do**
      $V_{(2\alpha)x}(i) = P(2i)$ XOR $P(2i - \alpha)$
    end for
  end if
end for

/* Prune the Voter Matrix based on the Sensitivity Λ */

Compute $\Phi = [\frac{N}{4} + [\frac{(50-\Lambda)}{150}] * (\frac{N}{4} - 1)]$

for $\alpha = 1 \rightarrow \Upsilon$ **do**
  V_val_α = The lowest power of two $\geq \Phi^{th}$ greatest element in the set V_α
  Discard all $V_\alpha(i)$ such that $V_\alpha(i) \leq V\_val_\alpha$
end for

/* LSB-MASK is delimiter between B and C bit windows */

LSB-MASK = $\{2^{\frac{N}{4}} - 1\}$ XOR $\{^{Min}_{\alpha}\{V\_val_\alpha\} - 1\}$

/*MSB-MASK is delimiter between A and B bit windows */

MSB-MASK= $\{2^{\frac{N}{4}} - 1\}$ XOR $\{\ ^{Max}_{\alpha}\{V\_val_\alpha\} - 1\}$

/* GRT function is defined as:

$$GRT(\beta_1 \ldots \beta_\Upsilon) = {}^{Max}_{\Upsilon}\left\{ \ \forall k : 1 \rightarrow \Upsilon \ \ \{^{j=1 \rightarrow \Upsilon}_{j \neq k} \Xi\{\beta_j\} \ \} \ \right\}$$

where the Ξ symbol denotes the operation

$^{\Upsilon}_{j=1}\ \Xi\{a_j\} = (a_1)$ AND (a_2) AND ... AND (a_Υ) */

/* Iterate through all pixels to identify bitflips based on dynamic thresholds */

for all $i \in [1 \ldots N]$ **do**
  /* Compute the Correction Vector and the Auxiliary Correction Vector*/
  $Corr_{Vect} = {}^{\Upsilon}_{j=1}\ \Xi\{\varphi_j\}$ where the set $\{\varphi_j, \ \forall j \in [1 \ldots \Upsilon]\} \equiv$ the set $\{V_\alpha(i), \ \forall (i, \alpha)$ such that $V_\alpha(i)$ is the result of an XOR operation on P(i)$\}$
  $Corr_{Aux} = \mathrm{GRT}(\varphi_j)$
  /* Combine the two vectors and apply the bit masks to obtain a bit-adjusted Correction Vector */
  $Corr_{Vect} = \{Corr_{Vect}$ OR $\{Corr_{Aux}$ AND MSB-MASK$\}\}$ AND LSB-MASK
  /* Apply the Correction Vector to the damaged Pixel. */
  P(i) = P(i) XOR $Corr_{Vect}$ /* $Corr_{Vect}$ will be zero if P(i) is not found to be damaged, */
end for

this is by having a dynamic statistical pre-analysis of the entire dataset prior to iterating through the dataset using a neighborhood window. This is achieved by first implementing an Υ-way pairing for each pixel, where Υ is the number of neighbors that must be consulted for identifying each temporally non-conforming bit, such that for each pixel. there will be $\frac{\Upsilon}{2}$ pairings with the neighbors on front and the other $\frac{\Upsilon}{2}$ with those on the back (hence Υ must be even). Exclusive-OR operation is done on each pair, and then for each way, the value at a particular index, derived from the sensitivity parameter Λ is selected to be representative of the correlation in the data pixels in that ordering. Given that the XOR results show the bit incongruities between two pixels, the higher the index value the more the turbulence expected to be natural in the dataset. The given methods of pairing has the least average distance from its Υ neighbors for any given pixel, as it is bit-compared with the $\frac{\Upsilon}{2}$ pixels immediately in front and back.

All the values above the cut-off value for each pairing are logically mapped into an $\frac{N}{2}$ x 2Υ voter matrix, with the view that for any given pixel in the dataset, there will exist an unique permutation in the matrix corresponding to that pixel if and only if its value is more deviant from its neighbors than is naturally expected at that location. If the sensitivity is higher, the total voters in the voter matrix will increase, and hence more bitflips could be identified. However, the number of voters required to accord to a change must vary in different bit windows: A does not require an absolute vote from all participating voters for each bit to be changed as it is the most stable region, whereas B does, while window C is masked off from any changes. These three windows are identified using the bit masks MSB-MASK and LSB-MASK.

The LSB-MASK, which corresponds to the right-most bits that form Window C, is logically defined as the lowest bit position below which the XOR results does not yield any locality information irrespective of the way the pixels are paired, as identified by the minimum binary weight indexed among all of the Υ ways of pairing. On the other hand, the MSB-MASK corresponds to the window A formed by the left-most bits in the pixels' binary representation, that are so consistent that a bit-wise inconsistency with the neighbors as shown by the XOR of corresponding pairs is likely to be a bitflip, and is identified by the maximum binary weight indexed among all the Υ ways the pixels can be paired such that the temporal distance between any two pixels are not more than $\frac{\Upsilon}{2}$ pixels at the most. It has been observed from extensive experimentation that $\Upsilon = 4$ yielded best performance for the two benchmarks presented here, i.e. the simulated datasets of the NGST application and the natural datasets from the OTIS application, though the system designer can subjectively decide the value for Υ and Λ optimally suited based on the statistical model of the datasets and the vulnerability to bitflips of the system being designed.

4 Preprocessing using Standard Algorithms

We compared *Algo_NGST* to two image smoothing algorithms tailored to suit the NGST datasets, which inherently seek out any discrepancies at input, and give a smoothened output that has lower variance. Some other commonly used smoothing algorithms include negative exponential, loess, running average, inverse square, bi-square etc. In graphics applications, they are used to give the image a smoother look by avoiding stark contrasts, especially at boundaries between surfaces, and thus to give a decidedly slightly-blurred appearance.

4.1 The Optimal Median Smoothing Algorithm

This [Algorithm 2] is an entirely value-based simple algorithm that has a sliding window approach. Results for our benchmarks have indicated that a sliding window of three pixels yields best results in terms of smaller relative error, as it cuts down on the false alarms caused by windows of higher width while still retaining nearly identical correction potential. It yields far better results than Mean Smoothing, due to the better robustness of median over mean. Mathematical techniques have been proposed for optimizing median smoothing [15].

Algorithm 2 Simple median smoothing algorithm using a window of width three pixels

```
P(1) = Median {P(1), P(2), P(3)}
for i = 2 → (N − 1) do
  P(i) = Median { P(i-1), P(i), P(i+1)}
end for
P(N) = Median {P(N-2), P(N-1), P(N)}
```

4.2 The Sliding Window Majority Bit Voting Algorithm

Since the fault models we consider here deal with bitflips in large chunks of input data, it makes sense to do a bitwise majority voting [Algorithm 3] rather than taking only values into account. Most often a bad pixel becomes flipped only at one bit, offsetting it by a value equivalent to the binary weight of the bit position as the representation here is for long integers. By labeling that pixel as an outlier because it has a deviant value, and discarding its current 16-bit representation altogether, we are losing potential information from the other 15 uncorrupted bits. However, if we were to consider each bit as a separate entity and compare it with the bits at the same binary weight in its temporal variants' bit representations, this can be avoided.

Algorithm 3 Bitwise Majority Voting Algorithm using a window of width three pixels

```
/* P(i, j) denotes the bit at positional offset j from the MSB of
the pixel P(i) */
P(0) = P(3)
P(N+1) = P(N-2)
for i = 1 → (N) do
  for j = 0 → 15 do
    if (( P(i, j) AND P(i-1, j)) OR (P(i-1, j) AND P(i+1, j)) OR
    (P(i, j) AND P(i+1, j) )) then
      P(i, j) = 1
    else
      P(i, j) = 0
    end if
  end for
end for
```

5 Numerical Results for the NGST Benchmark

The performance gained by using input preprocessing was studied for both the uncorrelated [Figure 2] and correlated fault models [Figure 4],as in [2.2.2] and [2.2.3], for the application-specific preprocessing algorithm Algo_NGST [Algorithm 1] in comparison with that of a generic median smoothing algorithm using simulated datasets, for a wide range of bitflip probabilities (Γ_0). The data that has been used for these simulations have the Gaussian correlation model in [2.2.1], and have the standard deviation σ corresponding to the sample NGST datasets. The performance in terms of gain in precision was measured as the average relative error remaining in the data after applying the respective preprocessing algorithm.

For $i = 1$ to $N(=64)$, $\Pi(i)$ is the uncorrupted dataset from the detectors in the ideal scenario, P(i) is the original potentially corrupted dataset, and $\Omega((i)$ is the datasets input to the application after preprocessing. The average relative errors $\Psi_{NoPreprocessing}$ and $\Psi_{Algorithm}$ are defined as:

$$\Psi_{NoPreprocessing} = \frac{\sum_{i=1}^{N} \frac{|P(i)-\Pi(i)|}{\Pi(i)}}{N} \quad (3)$$

$$\Psi_{Algorithm} = \frac{\sum_{i=1}^{N} \frac{|\Omega(i)-\Pi(i)|}{\Pi(i)}}{N} \quad (4)$$

Figure 3 shows the variation in execution overhead at various sensitivities (measured on a Pentium III 750 MHz system) as compared with that of the generic algorithms discussed in [4]. The results indicate that if the sensitivity is increased beyond the optimum value for that particular fault probability at input, the performance in terms of minimum achievable input relative error of the pixels actually deteriorates, along with the obvious increase in execution overhead. This is because the probability for false alarms also increases with sensitivity, and hence after a data-dependent threshold, it

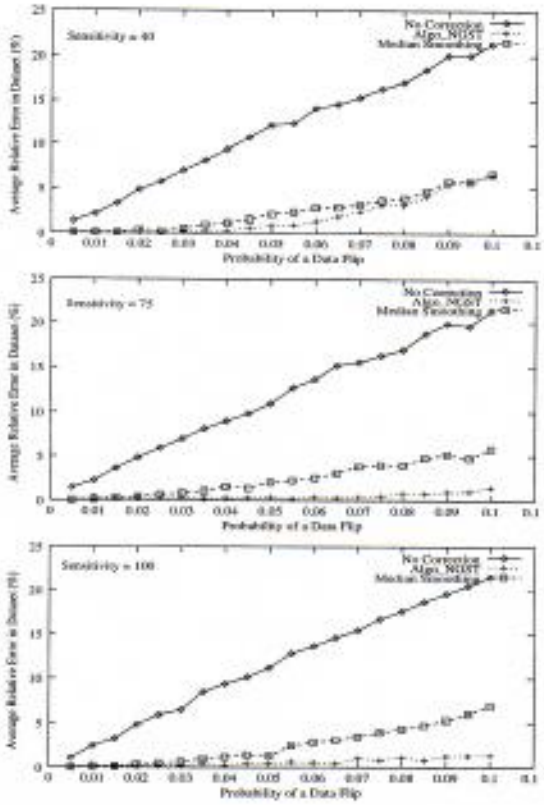

Figure 2. Performance comparison at varying sensitivities for Algo_NGST with the median smoothing algorithm

only adds to false alarms without contributing to any more corrections as shown by Figures 2 and 3.

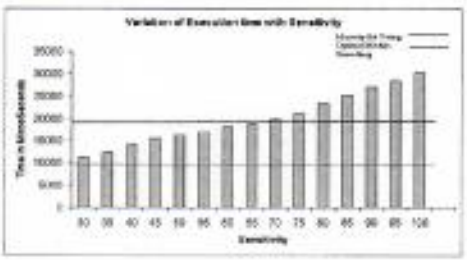

Figure 3. Preprocessing overhead for ALFT_NGST as a function of sensitivity Λ

Figure 4 shows that for the correlated fault model in [2.2.3], Algo_NGST does much better in combating the correlated failures in a bit-locality than the two smoothing algorithms, both of which show quite similar performance. The experimental results from Figures 2 and 4 for both fault models clearly highlight that using the proposed concept of input preprocessing enhances the precision and reliability of the input datasets by a highly desirable margin, compared to using the data as-is. For a practical range of Γ_0, the margin is shown to be as high as the order of 10%, depending on the preprocessing algorithm chosen. As is intuitive, even a small number of bitflips in the higher significant bits (window A) could cause an unacceptable error in the datasets, which could easily be obviated using preprocessing. Though a matter of choice for the designer, the dynamic input-specific preprocessing algorithm has shown to fare much better than the standard ones with static parameters, as it eliminated most of their possible false alarms while still garnering much more of locality information.

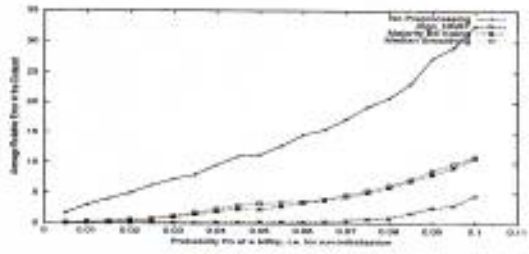

Figure 4. Performance comparison for NGST datasets affected with a correlated fault-model

Figure 5 shows how the preprocessing algorithms perform when the mean intensity of a dataset of N pixels varies across the entire gamut of possible values. For NGST, as in similar cases, there will always be some background noise present at the detector causing non-zero reads. The results are averaged over 100 datasets by injecting faults with $\Gamma_0 = 2.5\%$, with $\Upsilon = 4$ and $optimum\Lambda$ for each dataset for Algo_NGST.

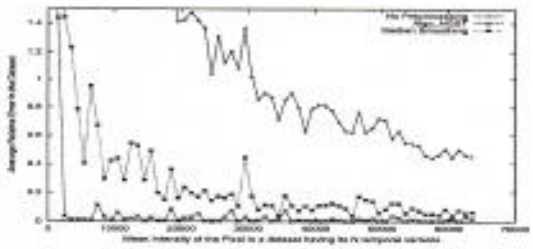

Figure 5. Performance Characteristics across entire gamut of datasets

6 Experimental Analyses of Theoretical Datasets

For the NGST benchmark having the assumed input analytical model in [2.2.1], the results for Algo_NGST

showed a very promising reduction in input average relative error Ψ, by an order of magnitude in the range ~ 50 to ~ 1000 on an average [Figure 2], for a probability of bitflip $\Gamma_0 \leq 10\%$ which is in the range of practical interest. These results were obtained by using experimentally optimized values of Υ and senstivity Λ, and hence it was of theoretical interest to see how the optimality of parameters would change when the input statistical model changed, to reflect the impractical cases also. Hence we simulated different quasi-NGST datasets, with σ varyingfrom very low to very high, and the same initial value $\Pi(1) = 27000$ for consistency.

Figure 6 shows that for very low natural variations, higher values of Υ give better performance, as is intuitive. When the natural variations that can lead to pseudo-corrections are extremely small, the more the number of neighbors consulted, the better. Hence for the synthetic dataset where the pixel intensity remains constant across time in the dataset, $\Upsilon = 6$ yields better results than $\Upsilon = 4$, which in turn is better than $\Upsilon = 2$, especially for higher Γ_0. However, when the natural variation gets larger in the dataset, using a large number of voters leads to many pseudo-corrections, as the temporal window of locality is exceeded too widely by the window of participating voters. The larger such variations, the more the pseudo-corrections from a larger Υ, leading to the degradation of performance with increasing Υ. Hence we see in Figure 6 that for larger σ, the performance of $\Upsilon = 2$ gets closer to that of $\Upsilon = 4$ and 6. The optimum value of Υ will also depend on the probability of bitflip Γ_0 when the natural variations in data cannot be a deciding factor all by itself, as with $\sigma = 250$ in Figure 6. Here, σ is such that for $\Gamma_0 \sim 0.04$, $\Upsilon = 6$ and $\Upsilon = 4$ has a cross-over in terms of optimality. Last row shows that for extremely turbulent datasets ($\sigma = 8000$, overflows are truncated to the maximum value), $\Upsilon = 6$ performs quite badly for low Γ_0, as the false-alarms do more harm than actual bitflips in this region, though it eventually gives the best performance when Γ_0 is very high. The overall 'steepness' of the curves may be inferred to as being inversely proportional to Υ, as supported by $\Upsilon = 6$ having the flattest curve among the three. Overall, $\Upsilon = 2$ seems to be the best choice for this scenario, as it does much better than the others for small-to-medium range of Γ_0.

7 The OTIS Application Benchmark

The second application for which we implemented our concept is the Orbital Thermal Imaging Spectrometer (OTIS), which is an orbiting probe-based distributed software that collects radiation data from the atmosphere using onboard sensors and processes it to obtain temperature and emissivity mappings of the geographical location. Due to the potentially destructive environment that the system is exposed to in its orbital flight, such as bombardment by free particles even so close as a few miles from the surface as in the South Atlantic Anomaly[16], the necessity for an accurate fault tolerance scheme becomes immediately apparent. The system lends itself to the technique of Application-Level Fault Tolerance (ALFT) [5] using the provision for a scaled-down secondary being run on another node as a backup to recover from faults occurring in the primary nodes, due to its Distributed System Design. Further studies have proven that a better level of fault-tolerance, and also at a lower overhead, can be obtained by extending this approach, by developing suitable filters for the primary output to determine whether to run the secondary, and then to decide on which output to choose based on a logic grid approach [29]. Such a scheme not only attempts recovery from process-killing faults as targeted by the aforementioned basic ALFT method, but also from faults that cause processes to generate incorrect output, without necessarily causing the node to be hung. We show next that the fault tolerance can be augmented to a much greater degree by incorporating input preprocessing algorithms into the system, as it attempts to eliminate the case when the other schemes fail catastrophically, which happens when both primary and secondary generate spurious output.

7.1 Input and Output Data Formats of OTIS

Input to OTIS is as a three-dimensional array [18], where the x and y axes depict the geography in two-dimensions and the z-axis depicts the radiation intensity of the same region in various wavelengths. Though the data is stored in the form of simple 32-bit floating point representation, the amount of data consumes several megabytes per frame of field of view and hence the input data volume is quite cumbersome for a large FOV. OTIS gives its output in various forms that include a two-dimensional temperature diagram in Kelvin and a three-dimensional emissivity diagram [18], similar to the input format. Since there is no inherent averaging or multiple imaging as in NGST, the correlation between precision at output and input is much higher in OTIS. Being representative of a physical phenomenon as radiation in a close vicinity, the data can be naturally expected to have a correlation pattern among neighboring pixels (spatial locality model) and also between pixels at the same location for different wavelengths (spectral locality model). Our experiments have shown that the former yields better expediency to our approach than the latter, as spectral correlation falls drastically on either side of a band of wavelengths.

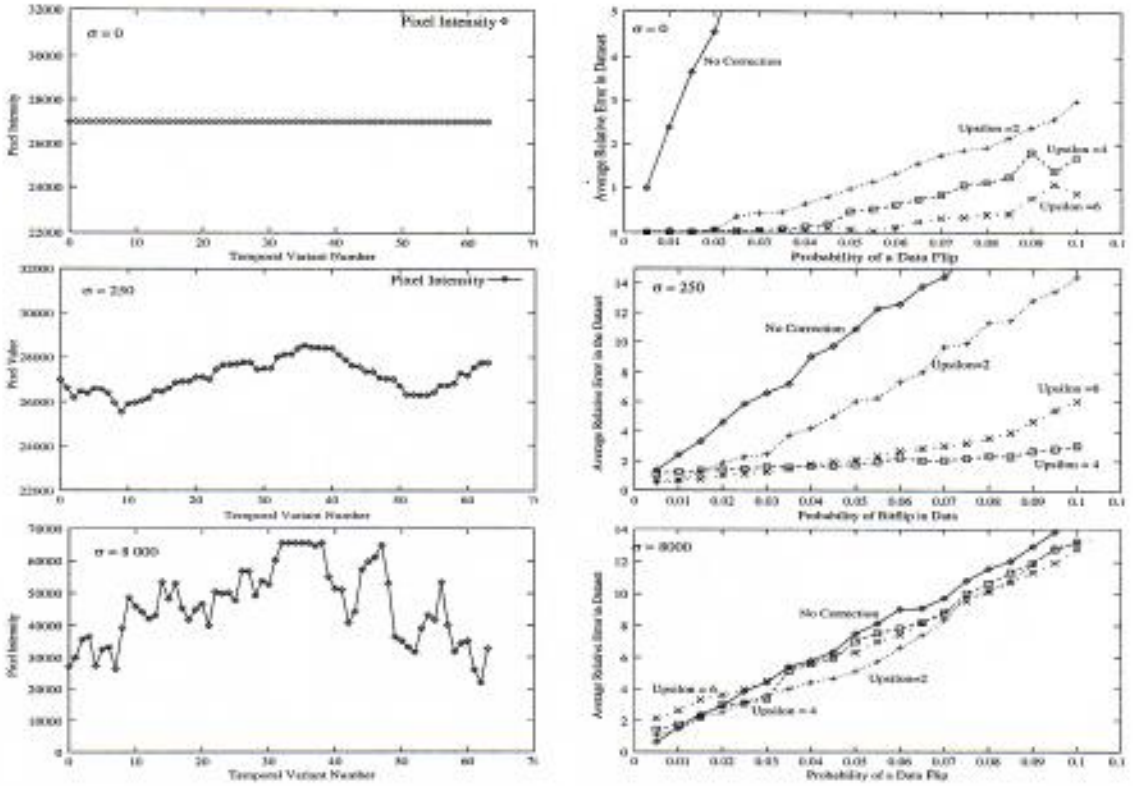

Figure 6. The synthetic datasets having the Gaussian temporal correlation model with highly varying values of σ, and the corresponding performance comparison graphs for Upsilon (Υ) = 2,4 and 6 on right

7.2 Fine Tuning of the Algorithm to suit OTIS

The lack of multiple datasets implies that the effect due to any possible false alarms (or pseudo-corrections), unless preempted, becomes far more pronounced at the output. False alarms can occur if the algorithm is too *tight*, which can cause exceptional data that is naturally too deviant and non-conforming to the values at its neighborhood, to be misinterpreted as faults and erringly made to conform to its vicinity. For OTIS, such genuinely non-conforming input can often be caused by extraordinary natural phenomena exhibiting hyper- or hypo-thermal activity, like geysers or volcanic eruptions, which is hence imperative to be retained at input. Thus the preprocessing algorithm needs to relax the dynamic threshold that is set for identifying outliers. There is a need for clear guidelines by which such fine tuning must be able to differentiate the two types of statistical anomalies at input: the naturally occurring and spurious outliers. We made the following assumptions, based on extensive statistical analysis and inferences from actual OTIS input datasets.

(1) *Valid exceptions occur as natural trends in the input dataset - exceptions manifested as very few non-conforming bit positions are faults:* A natural cause for outliers will manifest as a trend in a neighborhood, but is highly unlikely to be confined to a single pixel. A natural thermal phenomenon that does not have any effect on the temperature in its immediate vicinity is thermodynamically impossible, if the input is assumed to have high enough resolution.

(2) *Any theoretically out-of-bounds value is a fault:* There are theoretical absolute limits for the naturally occurring data sensed by OTIS, set by the laws of thermo-physics, and hence inconsistent pixels can be outright identified as faulty. In addition to the global absolute theoretical limits, there can also be logical cut-off bounds, depending on the localized geographical characteristics of the target area being scanned by the OTIS satellite, such as "tropical" or "arctic" bounds.

7.3 Study of Diverse Data Regions in OTIS input

With these hypotheses providing the framework to eliminate false alarms as much as possible, we developed *Algo_OTIS* and modified the algorithms in [4.1] and [4.2] to suit the OTIS datasets. We used three different datasets of OTIS [17], conveniently termed as "Blob", "Stripe" and "Spots" [Figure 7]. These datasets were specially chosen due to their physical characteristics

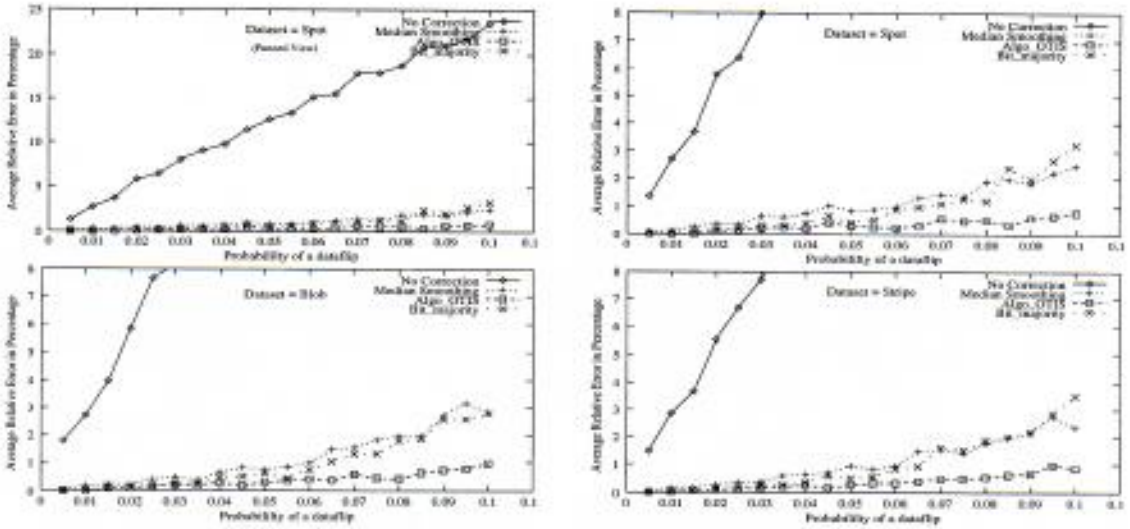

Figure 7. Performance Comparison, in terms of average relative data error, of Algo_OTIS with other algorithms

that exemplify nearly the entire gamut of variations likely to be encountered on site.

Blob : This dataset has broad areas of unchanging temperature, with a few dark spots scattered in the plot, representative of the majority of OTIS datasets, while the two others mentioned below constitute important special cases.

Stripe : This dataset has a very prominent vertical region of turbulent data through the center of the plot, but the surrounding regions are quite normal. The concentration of the turbulence into a relatively small region makes this data interesting.

Spots : This dataset has a plethora of very conspicuous spots, large and relatively small, all over the plot. Though more turbullent than "Stripe", the fact that this turbulence is spread over the entire region makes it more amicable to the correction schemes.

Figure 8. Three OTIS datasets with interesting characteristics

8 Numerical Results for the OTIS Benchmark

The results from Figure 8, generated for the uncorrelated fault-model in [2.2.2], clearly indicate that a very high level of gain in precision [Equations 3 and 4] is obtained when a preprocessing scheme is applied at input. For example, even with a probability of only 0.05 for bitflip, the input error $\Psi_{NoPreprocessing}$ is as high as $\sim 12\%$, which if directly used by science applications for processing could have yielded erroneous and misleading information and hence to invalid inferences in crucial scientific tasks, such as predicting weather patterns. With preprocessing, the error could be brought down to well below one percent. The performance of the Majority Bit Voting Algorithm with a Sliding Window of three pixels appears to be overall better than that of the Median Smoothing Algorithm, though the custom-designed Algo_OTIS performs far better than either of them in regions of $\Gamma_0 \geq 0.025$.

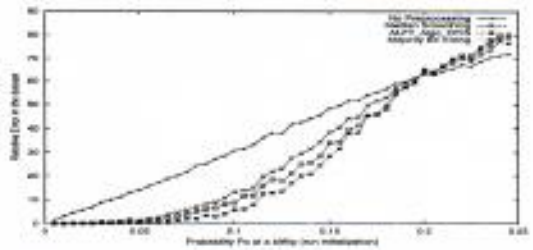

Figure 9. Performance comparison when OTIS datasets have correlated faults

Using the fault correlation model [Equation 2] explained in [2.2.3], we experimented with the three aforementioned OTIS datasets and found the performance comparisons shown in Figure 9. Interestingly, all the three algorithms had their breakdown points at just about the same probability (~ 0.2) for run initialization. At higher Γ_{init}, though only of theoretical interest

due to impracticality, the preprocessing actually deteriorates the error due to bitflips. As the proportion of corrupted bits to clean bits in a dataset is increased, a point is reached when the reverse phenomenon happens: the corrupted bits cause some of the remaining clean bits to be pseudo-corrected leading to further corruption. As all three preprocessing algorithms use the method of interpolation from neighboring bits as their core approach, it is only intuitive that the breakdown points also be at similar probabilities. In an uncorrelated model as in [2.2.2], this threshold probability is obviously approached at ~ 0.5. We recommend the technique of storing the neighboring pixels using a preset mapping into different physical regions in the memory organization, so that when they are retrieved for preprocessing, the correlated block faults occurring in contiguous regions in memory will not affect the temporal or spatial redundancy preserved elsewhere.

9 Conclusions

We showed that highly significant improvements in reliability and precision can be achieved by dynamically preprocessing input data for preserving its science information, substantially reducing the need for expensive hardware and software redundancy. A wide range of applications process naturally correlated data that exhibit a high level of temporal, spatial, spectral, and other forms of inherent redundancy at their input, which enables isolation of significant data faults using interpolation with respective neighbors. Our approach can be a versatile and scalable complement to other fault-tolerance schemes for safety-critical systems where input reliability is paramount. Integrating our algorithm into conforming applications while in the design phase itself, rather than as a separate preprocessing layer in the fault-tolerance scheme, can further lower the overhead.

Acknowledgment

The authors wish to thank Dr. Vijay Lakamraju for stimulating discussions and suggestions. This research on fault tolerance in the application layer is supported by NSF under the grant number CCR-0104482.

References

[1] C.M. Krishna and K.G. Shin, "Real-Time Systems", McGraw-Hill, 1997

[2] D. K. Pradhan, "Fault Tolerant Computer System Design", Prentice-Hall Inc., 1995

[3] K.H. Huang and J.A. Abraham, "Algorithm-Based Fault Tolerance for Matrix Operations", IEEE Transactions on Computers, Vol. c-33, No.6, June 1984

[4] A. Avizienis, "The Methodology of N-Version Programming," Chapter 2 of Software Fault Tolerance, M. R. Lyu (ed.), Wiley, 23-46, 1995.

[5] J. Haines, V. Lakamraju, I. Koren, C.M. Krishna, "Application-level Fault Tolerance as a Complement to System-level Fault Tolerance", The Journal of Supercomputing, 16:53-68, 2000.

[6] R. Ferraro, "Remote Exploration and Experimentation Project Plan", Technical Report, Jet Propulsion Lab, NASA, July 2000.

[7] H.S. Stockman, et al., "The Next Generation Space Telescope: Visiting a Time When Galaxies Were Young", The NGST Study Team, 1997.

[8] Peter Stockman, "Beyond HST : The Next Generation Space Telescope", presented in the 33rd Space Congress, 1997.

[9] J.L. Barth and J.C. Isaacs, "The radiation environment for the Next Generation Space Telescope", December 1999.

[10] B.J. Rauscher, J.C. Isaacs and K. Long, "Cosmic Ray Management on NGST 1: The effect of Cosmic Rays on Near Infrared Imaging Exposure Times", Space Telescope Science Institute (STSci), 2000.

[11] H.S. Stockman, D. Fixsen, et al., "Cosmic Ray Rejection and Image Processing Aboard the Next Generation Space Telescope", 1998.

[12] D.J. Fixsen, R.J. Hanisch, et al., "Cosmic Ray Rejection and Data Compression for NGST", ASP Conf. Ser., Vol. 216, Astronomical Data Analysis Software and Systems IX, eds. N. Manset, C. Veillet, D. Crabtree (San Francisco: ASP), 539, 2000.

[13] M. Im, H.S. Stockman, "Science with the NGST", ASP Conference Series, 133, 263, E.P. Smith, A. Koratkar eds., 1998.

[14] NASA/Science Office of Standards and Technology, NOST 100-2.0, Code 633.2, "Definition of the Flexible Image Transport System (FITS)", 1999.
<http://archive.stsci.edu/fits/fits_standard/>

[15] W. Hardle and W. Steiger, "Optimal Median Smoothing", Universite Catholique de Louvain, Center for Operational Research and Econometrics, 1970.

[16] B. James, O. Norton, et al., "The Natural Space Environment : Effects on Spacecraft", 1994.

[17] E. Ciocca, "Application Level Fault Tolerance and Detection", Master's thesis, University of Massachusetts Amherst, 2002.

[18] The official OTIS websites:
<www-ree.jpl.nasa.gov/fy00_reports/OTIS/OTIS.html>
<hpcc.arc.nasa.gov/reports/annrep98/REE/arganrpt.html>