# Improving Run-Time Bug Detection in Aviation Software Using Program Slicing

Hu Huang, Samuel Z. Guyer, Jason H. Rife
Tufts University

Tufts UNIVERSITY | School of Engineering

## Abstract

We present an improvement to run-time bug detection by using program slicing instead of a heuristic to identify relevant program variables to monitor. Our results show that selecting variables using the program slice produces higher sensitivity compared to selecting variables heuristically. Moreover, at least in the experiments conducted, program slicing resulted in more consistent detection, with lower variance in sensitivity as compared to the heuristic approach.

## Introduction

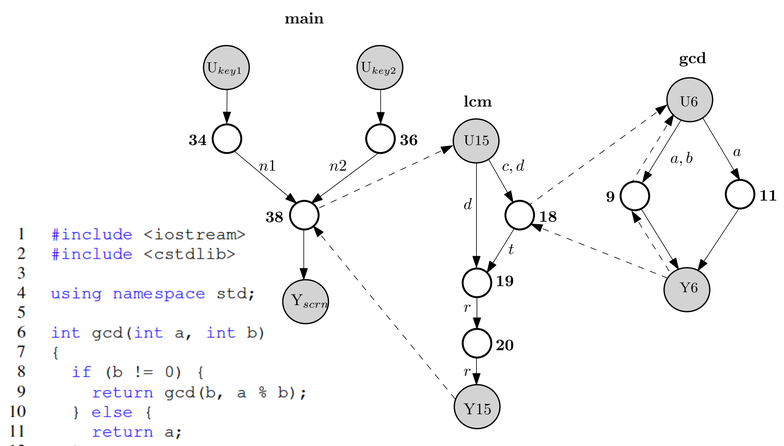
- We address the question how to choose monitoring variables to detect bugs in software by leveraging program slicing, a well-known static analysis technique originally proposed by Weiser[1].
- Our hypothesis is that variables chosen using program slicing result in better bug detector performance than variables chosen through a reasonable heuristic.
- We construct bug detectors using a standard machine learning algorithm, the One-Class SVM. We use the One-Class SVM to conduct two computational experiments in order to test our hypothesis. The experiments, which consider a simplified but representative flight control code, address two questions.

## Research Questions

**Question 1:** *For the same level of specificity, do instrumentation variables chosen using a program slice, as opposed to a heuristic, result in higher monitor sensitivity for bugs injected at different locations in the model code?*

**Question 2:** *Is the benefit of selecting instrumentation variables using program slicing enhanced as bug magnitude gets smaller, indicating greater sensitivity when detection is hardest?*

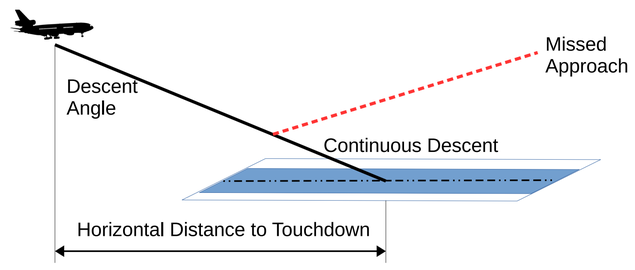
## Background




```
1  #include <iostream>
2  #include <cstdlib>
3
4  using namespace std;
5
6  int gcd(int a, int b)
7  {
8    if (b != 0) {
9      return gcd(b, a % b);
10   } else {
11     return a;
12   }
13 }
14
15 int lcm(int c, int d)
16 {
17   int r, t;
18   t = abs(c) / gcd(c, d);
19   r = t * abs(d);
20   return r;
21 }
22
23 /*
24  * Computing the least commmon multiple
25  */
26 int main()
27 {
28   int n1;
29   int n2;
30   cout << "*****************" << endl;
31   cout << "Computing the LCM" << endl;
32   cout << "*****************" << endl;
33   cout << "Enter the first integer: ";
34   cin >> n1;
35   cout << "Enter the second integer: ";
36   cin >> n2;
37   cout << "LCM of " << n1 << " and "
38        << n2 <<" is: " << lcm(n1,n2) << endl;
39 }
```

## Methodology

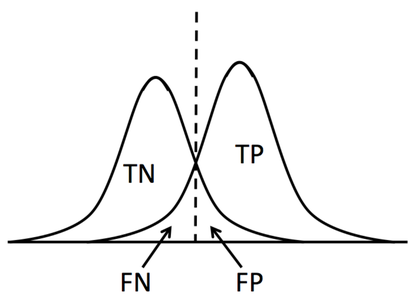
### Operational Scenario:




### Evaluation Metric:

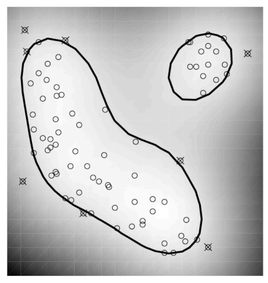
$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$




### Anomaly Detection Model:

One-Class SVM



$$\text{Kernel} = \text{RBF}$$
$$\gamma = 0.001$$
$$\nu = 0.2$$

## Results

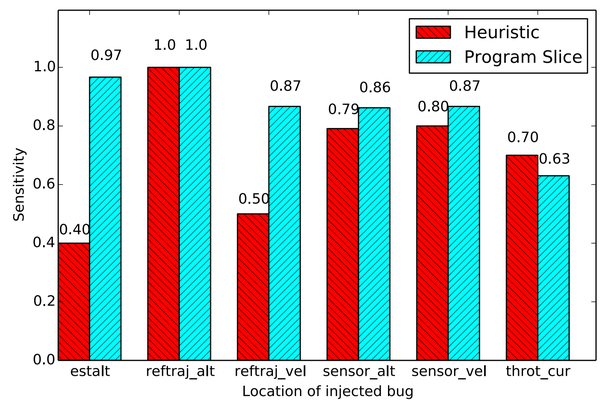



**Figure 3.** Sensitivities of both heuristic and program slice detectors for the six local variables where bugs were injected (higher is better). While the program slice performs better across four of the six variables, it does not always perform better. The program slice detector actually performs slightly worse for `throt_cur` and ties for `reftraj_alt`. For all the bug injection sites, the specificity was set to 0.8.

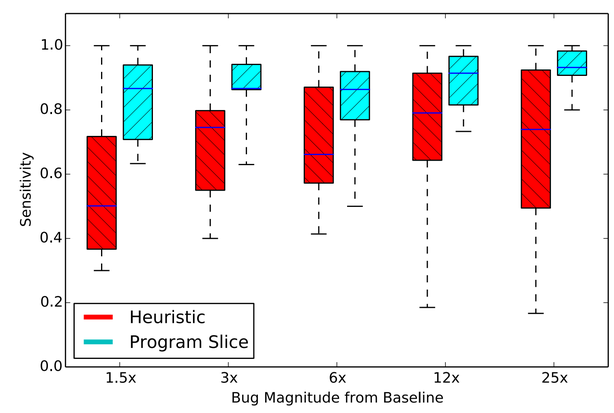



**Figure 4.** Box plots for Experiment 2, where higher on the vertical axis corresponds to better sensitivity. With larger bug magnitudes, both the heuristic and the program slice detectors have relatively high sensitivities. At lower bug magnitudes, the program slice detectors perform significantly better.

## Future Work

We are currently preparing a manuscript to be submitted to TCPS journal that expands on the theoretical notions of the program slice for a CPS.

In addition, we are currently building compiler infrastructure to automate program slicing and will be applying this technique on real-world code bases such as Ardupilot.

## Acknowledgement

This research was supported by National Science Foundation under grant CNS-1329341

## Contact

Hu Huang
Tufts University
Email: hu.huang@tufts.edu

## References

1. Mark Weiser. 1984. Program Slicing. IEEE Transactions on Software Engineering 4, SE-10 (1984), 352–357.
2.