Expedient Integration of a Laser-Based Feature Detector

David Gustafson and **Aaron Chleborad** and **Andrew King**
dag@ksu.edu aaron123@ksu.edu aking@ksu.edu
234 Nichols Hall
Department of Computing and Information Science
Kansas State University

Abstract

For our programming convenience we have implemented an extensible robotic control framework in the Java programming language. This framework manages some of the tedious details in robotics programming (sensor - behavior integration and concurrency issues) as well as enforces a modular programming style that works well when a team is involved with robot software development. We discuss how this framework allows for modular software development. As an example we show how a laser range data based object detection algorithm can be implemented as a framework sensor module and how information provided by that new module can be utilization by the planning and mobility modules.

Introduction

We have developed a robotic control framework in the Java language to expedite robotic control software development in a team of student or academic researchers. Our observations of the development process in this environment indicates that developers have an easier time building robotic software systems when there is a high degree of separation between the low level details of controling the robot itself and the abstract specification of high level behaviors.

Robot control systems like ARIA and Playerstage provide a robust and sensible separation from the low-level details of robotics programming but developing certain types of robotic behaviors, such as complex "event-react" scenarios on those systems can require the developer to build a clever imperative style solution in a language like C. Students end up spending more time fixing concurrency errors and integrating team member's code than testing the capability of their software on the real robot.

We have used this framework to rapidly prototype complex robotic control software such as a system designed to complete in the Semantic Robot Vision Challenge (where the robot must navigate an environment safely and visually identify objects) (Gustafson et al. 2007) and for a robotic IED detection and disposal platform. Both of these projects utilized the MobileRobotics Pioneer P3AT robot platform.

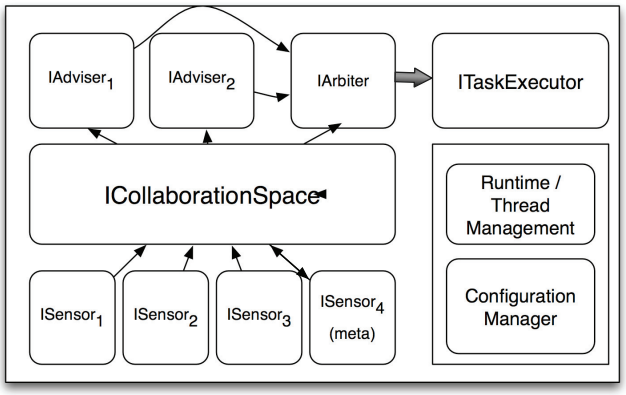

Figure 1: The taxonomy of framework modules and how information about the world and command directives flow between them.

High Level Framework

This robotic control framework was designed and built for a robot software development environment that is comprised of developers with a wide range of backgrounds and programming capability. Many times, whether for a class or research project, the team developing code to drive a robot system will be comprised of five or more software engineering students. Management of the codebase is often made challenging due to pressure to rapidly create a runnable codebase where the robot and its codebase can be tested. Towards the goal of ameliorating code quality challenges in such an environment, this framework: (1) enforces a strict modularity (a running system is a collection of Java `IModule` objects) where module function is constrained by module type (2) internally manages inter-module concurrency and communication, and (3) is implemented in a programming language with some level of type-safety (Java).

As mentioned above the `IModule` are constrained in their function by their type. A operational robot control system built in this framework would normally consist of one `ICollaborationSpace` module, one or more

`ISensor` modules, one or more `IAdviser` modules, and one `IArbiter` modules.

`ICollaborationSpace` modules are responsible for managing the current world knowlege that can be made available to the other modules in the system. Other modules can retrieve or post data to an `ICollaborationSpace` module via key-value queries.

`ISensor` modules define data conduits between sensors (both real and virtual) to the collaboration space. The framework supports two types of `ISensors` active and passive. Active sensors will only post data to the collaboration space if directed by a control module, whereas passive sensors are constantly posting data to the collaboration space. The framework automatically manages the concurrency and threading of a passive `ISensor`

`IAdviser` modules query the collaboration space and then emit a plan or task list based on the state of the collaboration space. Tasks are generated either by composition of sub task types the framework automatically provides or by instantiating a fresh developer programmed task.

`IArbiter` modules receive tasks that `IAdviser` modules have emitted and determine what task will be committed into action. An `IAdviser` has access to the collaboration space also.

The `ITaskExecutor` module executes the Tasks (and possibly subtasks) that the `IArbiter` has committed to action. An `ITaskExecutor` manages the ordering and timing of task execution and completion.

A running system is defined in a configuration file that is read and loaded at runtime. The framework supports loading modules from disk, as well as runtime injection of modules via a network interface if so configured.

The framework includes default implementations of the "core" system modules, such as an `ICollaborationSpace`, `IArbiter`, and `ITaskExecutor`. An example configuration file is as follows.

```
ICollaborationSpace=edu.robotics.DCS
ITaskExecutor=edu.robotics.DfltExecutor
IArbiter=edu.robotics.DfltArbiter
IAdviser=edu.robotics.RandomWalkAdviser
ISensor=edu.robotics.P3ATSonar
ISensor=edu.robotics.P3ATBump
ISensor=edu.robotics.UKYOLaser
```

Laser based feature detector

The data provided by the SICK LMS 200 laser after a complete scan has high angular resolution by default. The default settings give a scanning angle of $180°$ allowing a reading for every $1°$ at a range of 30m for a comprehensive range image detailing the features in the environment on the laser scan plane. For finer sensor readings, the angular resolution can be increased up to $.25°$ with a loss in scanning angle which is reduced to $100°$. A good compromise allows an increase of the angular resolution to $.50°$ while keeping the scanning angle at $180°$, giving a total of 360 range readings every 26 ms (SIC 2002). Range images are typically specified in the polar coordinate system $\{(\rho_i, \theta_i) \mid i = 1..N_R\}$, with$(\rho_i, \theta_i)$ being the polar coordinates of the i^{th} range reading with ρ denoting the range in millimeters and θ the angle of the reading in radians, with the origin at the location of the sensor.

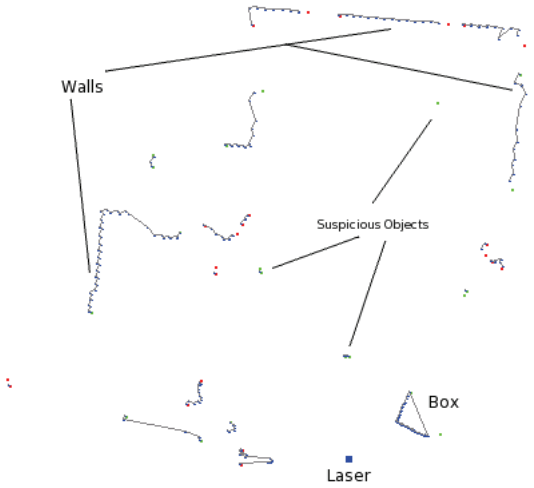

Figure 2: A typical range image of the environment. Three points of interest are detected along with other items in the environment such as walls, a box and clutter. Red and green dots signify breakpoints in the laser data.

The first step in the identification process requires a definition of each polar coordinate in terms of a function f. Because we are working with digital data that is finite, the maximum possible range change is also finite. The shortest distance over which the change can occur is between individual range readings. This provides a one-dimensional function which can be utilized during data processing. The process of finding breakpoints and rupture points involves identifying range readings that change abruptly from their surrounding range points in the laser scan data by exceeding a threshold value. A typical method of finding these abrupt changes involves using the second-order derivative of the digital function, i.e. the laser scan data, to isolate them. To understand how to arrive at the second-order derivative it is beneficial to understand how to calculate the first-order derivative. The basic definition of the first-order derivative of a one-dimensional function $f(x)$ is the forward-difference (Gonzalez and Woods 2007)(Basson and Engelbrecht 1999)

$$\frac{\partial f}{\partial x} = \frac{\Delta f}{\Delta x} \approx \frac{f(x+1) - f(x)}{(x+1) - x} \tag{1}$$

To find the second-order derivative, the first-order forward-difference is extended to two forward differences(Gonzalez and Woods 2007)(Basson and Engelbrecht 1999)(Morse 2000)

$$\frac{\partial^2 f}{\partial x^2} = \frac{\Delta f}{\Delta x} - \frac{\Delta f}{\Delta (x-1)} \tag{2}$$

$$\approx \frac{f(x+1) - f(x)}{(x+1) - x} - \frac{f(x) - f(x-1)}{x - (x-1)} \tag{3}$$

$$= f(x+1) + f(x-1) - 2f(x) \quad (4)$$

One should note that the use of the partial derivative does not affect the result of what is trying to be accomplished as $\frac{\partial f}{\partial x} = \frac{df}{dx}$ when there is only one variable in the function. Partial derivatives that are used here remain consistent with image processing literature. The second-order derivative was chosen as the method to extract breakpoints since it is non-zero at the start and end of all changes in range. Segments of data are formed by pairing breakpoints in succession starting from the beginning of the breakpoint list. The segmentation is meant to classify the data into groups which could possibly associate them with structures in the environment (Nunez et al. 2006). It was stated previously that in order to classify breakpoints and rupture points, they must differ significantly from their surrounding data points by a certain threshold. (Borges and Aldon 2004) describe a technique that can be used to adjust the threshold dynamically thereby reducing the likelihood of adding inappropriate breakpoints. In their algorithm, two consecutive range readings belong to different segments if

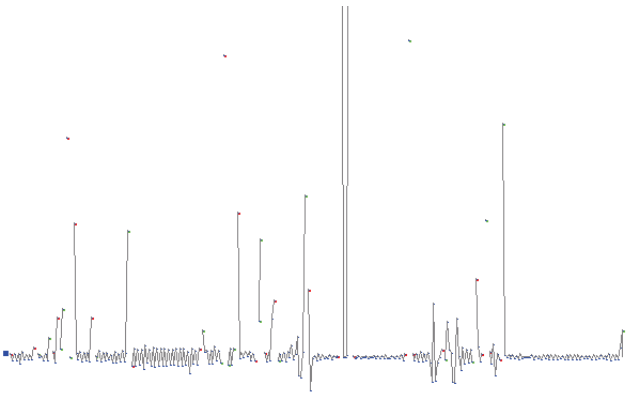

Figure 3: 2^{nd} derivative image of Figure 2. Note green and red dots represent breakpoints. The adaptive breakpoint algorithm is used to determine these dynamically. Larger spikes represent large sudden changes in the laser data. The large spike continuing off the image edge represents the suspicious object near the laser in Figure 2. Conversely, the flat parts of the data represent the walls and areas of little change from point to point.

$$\|(\rho,\theta)_i - (\rho,\theta)_{i-1}\| > \rho_{i-1} \cdot \frac{\sin\Delta\theta}{\sin(\lambda - \Delta\theta)} + 3\sigma_r \quad (5)$$

where $\Delta\theta$ is the angular resolution, λ corresponds to the worst case of incidence angle of the laser scan ray with respect to a line for which the scan points are still reliable. The statistical error in the range readings is represented by σ_r. Values for $\Delta\theta$, λ and σ_r are .50, 10 and .005 respectively and were obtained from a mixture of (Nunez et al. 2006), (SIC 2002) and this experiment's specific value for angular resolution.

If a breakpoint is detected, both $(\rho,\theta)_i$ and $(\rho,\theta)_{i-1}$ added to the breakpoint list. The reader should note that

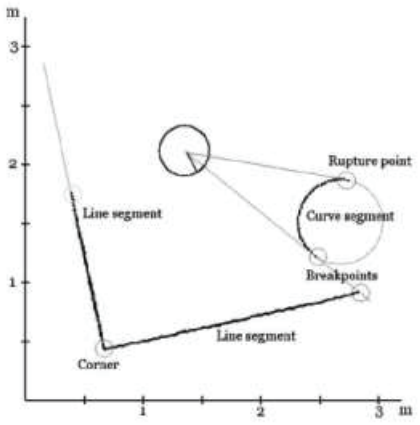

Figure 4: Segment illustration as defined in (Nunez et al. 2006)

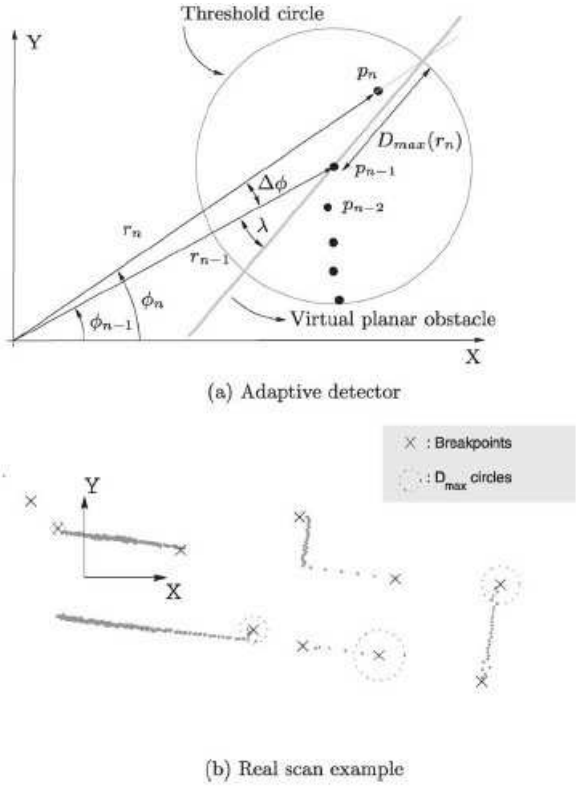

Figure 5: Adaptive breakpoint detector as defined in (Borges and Aldon 2004). This determines how far a point must be from a neighboring point so as to classify it as a breakpoint. The green and red dots in Figures 2 and 3 were detected using this technique.

$\|(\rho,\theta)_i - (\rho,\theta)_{i-1}\|$ corresponds to the first derivative of the digital laser data function. As stated earlier, the implementation in this report uses the second-order order derivative as noted in (4) but the results are similar due to the fact that (Borges and Aldon 2004) record both the i^{th} and $i^{th}-1$ in a single iteration of the processing loop while this implementation records only the $i^{th}-1$ per iteration. However, the second derivative yields a non-zero result at both the start and end of changes in range which is essentially equivalent to both points captured in (Borges and Aldon 2004) implementation utilizing the first order derivative.

Once the list of breakpoints has been compiled, it can used in a variety of ways ranging from assisting in localization tasks to object detection efforts.

Laser Object Detector as Framework Modules

In this section we discuss how we have built framework modules that allow a robot to perform as an IED (improvised explosive device) detection and disposal robot. IED objects in the environment are cardboard cyclinders of varying size and location. These framework modules allow the robot to detect the IED objects with a laser range scanner (such as a SICK LMS or UKYO) and then approach the IED.

object detector

The object detector module, `ObjectSensor` is an passive virtual `ISensor`. This means that the object detector is being driven by the framework thread manager to query the collaboration space for some knowlege, proccess this knowlege, then post some new knowlege back into the collaboration space. This sensor queries the collaboration space asking for the latest `LaserRangeScanData` (A data type provided by the framework) which contains a vector of 180 integer values representing the range in milimeters where the vector index is the scan direction. In this instance the `ObjectSensor` is continually retrieve the latest scan posted by the laser range finder module, applies the range based object detection algorithm, and then posts a collection of world vectors (`MapVector` object) back to the collaboration space.

behavior module

The behavior module, `ObjectAdviser` is a relatively simple `IAdviser` which periodically checks the collaboration space for world object vectors. If object vectors are retrieved from an area of the map the robot has not yet visited (according to the collaboration space) then the `ObjectAdvisor` will emit a composition of the built in task types, one that tells the robot to drive to a coordinate, the other to generate a collaboration space post (a `MapLandmark` object) that indicates the robot has visited this object.

Conclusion

This framework has allowed us to rapidly prototype robot control software for several different scenarios. The framework has increased the ease of integrating code from different student developers. The framework configuration for the Semantic Robot Vision Challenge consisted of 5 non-trivial `ISensor` modules, and 3 different `IAdivser` modules. The `IAdviser` modules can be enabled or disabled depending on how well they each perform in the SRVC environment (which is usually much different than the home testing environment.) 5 different student developers were involved in the implementation of these 8 modules, and very no effort was expended debugging adverse interactions between different developer's code. This has allowed us to both concentrate on challenging robot problems in parallel with allowing novice developers to quickly contribute to the project.

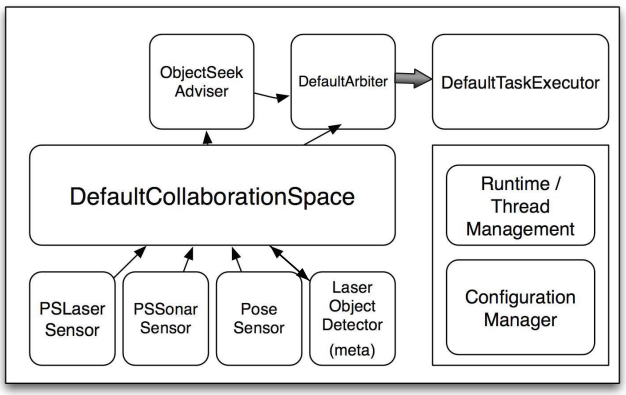

Figure 6: Depiction of the modules running in a robot control system configured to detect objects via the laser detector and approach detected objects.

References

Basson, E., and Engelbrecht, A. 1999. Approximation of a function and its derivatives in feedforward neural networks.

Borges, G. A., and Aldon, M.-J. 2004. Line extraction in 2d range images for mobile robotics. *J. Intell. Robotics Syst.* 40(3):267–297.

Gonzalez, R. C., and Woods, R. E. 2007. *Digital Image Processing (3rd Edition)*. Prentice Hall.

Gustafson, D.; Marlen, M.; Chavez, A.; and King, A. 2007. Ksu willie in the aaai 2007 semantic vision challenge.

Morse, B. S. 2000. Lecture 13: Edge detection. lecture.

Nunez, P.; Vazquez-Martin, R.; del Toro, J.; Bandera, A.; and Sandoval, F. 2006. Feature extraction from laser scan data based on curvature estimation for mobile robotics. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* 1167–1172.

SICK AG. 2002. *LMS 200 / LMS 211 / LMS 220 / LMS 221 / LMS 291 Laser Measurement Systems*.