CHEAP OUTDOORS AUTONOMOUS NAVIGATION WITH ROS

by

Noah Johnson

A Thesis

Submitted to the Division of Natural Sciences New College of Florida in partial fulfillment of the requirements for the degree Bachelor of Arts under the sponsorship of Professor Gary Kalmanovich

Sarasota, Florida May 2017

Acknowledgments

This is the acknowledgements section. You should replace this with your own acknowledgements.

Contents

Acknowledgments ii

Contents iii

Abstract iv

1 Theory **1**

1.1 Probability Background 1

1.2 Bayes Filter 2

1.2.1 Scenario 2

1.2.2 Derivation 3

1.2.3 Example 5

1.3 Kalman Filter 5

Bibliography **6**

Cheap Outdoors Autonomous Navigation with ROS

by

Noah Johnson

Submitted to the Division of Natural Sciences
on May 23, 2017, in partial fulfillment of the
requirements for the degrees of
Bachelor of Arts in Computer Science
and
Bachelor of Arts in Applied Mathematics

Abstract

In this thesis, I designed and implemented a compiler which performs optimizations that reduce the number of low-level floating point operations necessary for a specific task; this involves the optimization of chains of floating point operations as well as the implementation of a "fixed" point data type that allows some floating point operations to simulated with integer arithmetic. The source language of the compiler is a subset of C, and the destination language is assembly language for a micro-floating point CPU. An instruction-level simulator of the CPU was written to allow testing of the code. A series of test pieces of codes was compiled, both with and without optimization, to determine how effective these optimizations were.

Professor Gary Kalmanovich ....

May 23, 2017

Chapter 1

Theory

1.1 Probability Background

Discrete random variables have a finite output space of possible values that may be observed. Let X be a random variable, then we define the probability that we observe value x from X as $p(X = x) \equiv p(x)$. Since x is arbitrary, this defines a probability distribution. For every random variable X, we have

$$\sum_{x \in X} p(x) = 1 \tag{1.1}$$

Given two more random variables Y and Z, we'll define the joint distribution $p(X = x \text{ and } Y = y \text{ and } Z = z) \equiv p(x, y, z)$, and the conditional probability $p(X = x \text{ given that } Y = y \text{ and } Z = z) \equiv p(x \mid y, z)$. The conditional probability is defined to be

$$p(x \mid y, z) = \frac{p(x, y, z)}{p(y, z)} \tag{1.2}$$

The *Law of Total Probability* states that $p(x) = \sum_{y \in Y} p(x, y)$. Extending this law to use a third random variable Z, and incorporating the definition of conditional probability, we end up with the following equation:

$$p(x \mid z) = \sum_{y \in Y} p(x, y, z) = \sum_{y \in Y} p(y, z)p(x \mid y, z) \tag{1.3}$$

Lastly, we can use equation 1.2 to derive a version of Bayes' Theorem.

$$p(x \mid y, z) = \frac{p(x, y, z)}{p(y, z)} = \frac{p(y, x, z)}{p(x, z)} * \frac{p(x, z)}{p(y, z)} = \frac{p(y \mid x, z)p(x, z)}{p(y \mid z)} \tag{1.4}$$

In the future this will prove to be a useful tool to compute a posterior probability distribution $p(x \mid y)$ from the inverse conditional probability $p(y \mid x)$ and the prior probability distribution $p(x)$.

1.2 Bayes Filter

1.2.1 Scenario

Consider the general case of a robot which uses sensors to gather information about its environment. These sensors provide readings at discrete time steps $t = 0, 1, 2, ...$. Some amount of noise is associated with each of these readings. At each time step t, the robot may execute commands to affect its environment, and wishes to know its current state.

Let's encode the robot's current state at time t in the vector x_t. Similarly, z_t will represent a sensor measurement at time t, and u_t will represent the commands issued by the robot at time t. For each of these vectors we will use the notation $z_{1:t} = z_1, z_2, ..., z_t$.

The robot only has access to data in the form of z_t and u_t. Thus it cannot ever have perfect knowledge of its state x_t. It will have to make do by storing a probability distribution assigning a probability to every possible realization of x_t. This posterior probability distribution will represent the robot's belief in its current state, and should be conditioned on all available data. Thus we'll define the robot's belief distribution to be:

$$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \tag{1.5}$$

1.2.2 Derivation

We can use equation 1.4 to rewrite $bel(x_t)$:

$$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) = \frac{p(z_t \mid x_t, z_{1:t-1}, u_{1:t})p(x_t \mid z_{1:t-1}, u_{1:t})}{p(z_t \mid z_{1:t-1}, u_{1:t})} \tag{1.6}$$

In order to simplify $p(z_t \mid x_t, z_{1:t-1}, u_{1:t})$, we'll have to make an important assumption. We'll assume that the state x_t satisfies the Markov property, that is, x_t perfectly encapsulates all prior information. Thus if x_t is known, then $z_{1:t}$ and $u_{1:t}$ are redundant. This assumption lets us remove consideration of past sensor measurements and commands, and to rewrite the belief distribution as:

$$bel(x_t) = \frac{p(z_t \mid x_t)p(x_t \mid z_{1:t-1}, u_{1:t})}{p(z_t \mid z_{1:t-1}, u_{1:t})} \tag{1.7}$$

Notice that $p(z_t \mid z_{1:t-1}, u_{1:t})$ is a constant with respect to x_t. Thus it makes sense to let $\eta = (p(z_t \mid z_{1:t-1}, u_{1:t}))^{-1}$ and rewrite the belief distribution as:

$$bel(x_t) = \eta p(z_t \mid x_t)p(x_t \mid z_{1:t-1}, u_{1:t}) \tag{1.8}$$

Now we are left with two distributions of interest. Looking closely one may notice that $p(x_t \mid z_{1:t-1}, u_{1:t})$ is simply our original belief distribution, equation 1.5, but not conditioned on the most recent sensor measurement, z_t. Let us refer to this distribution as $\overline{bel}(x_t)$, and break it down further using equation 1.3 and our Markov assumption:

$$
\begin{aligned}
&\overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \sum_{x_{t-1}} p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \\
&= \sum_{x_{t-1}} p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \\
&= \sum_{x_{t-1}} p(x_t \mid x_{t-1}, u_t) bel(x_{t-1})
\end{aligned}
\tag{1.9}
$$

We have arrived at a recursive definition of $bel(x_t)$ with respect to $bel(x_{t-1})$! As long as $p(x_t \mid x_{t-1}, u_t)$ and $p(z_t \mid x_t)$ are known, we can recursively calculate $bel(x_t)$.

$p(x_t \mid x_{t-1}, u_t)$ defines a stochastic model for the robot's state, defining how the robot's state will evolve over time based upon what commands it issues. This probability distribution is known as the *state transition probability.* []

$p(z_t \mid x_t)$ also defines a stochastic model, modeling the sensor measurements z_t as noisy projections of the robot's environment. This distribution will be referred to as the *measurement probability.* []

Once we have models for both the *state transition probability* and *measurement probability*, we can finally construct the algorithm known as Bayes' Filter:

$$
\begin{aligned}
&\text{Given } (bel(x_{t-1}), u_t, z_t): \\
&\quad \text{For all possible states } x_t^* \text{ compute:} \\
&\qquad \overline{bel}(x_t^*) = \sum_{x_{t-1}^* \in x_{t-1}} p(x_t^* \mid x_{t-1}^*, u_t) bel(x_{t-1}^*) \\
&\qquad bel(x_t^*) = \eta p(z_t \mid x_t^*) \overline{bel}(x_t^*) \\
&\quad \text{end loop} \\
&\quad \text{Set } \sum_{x_t^* \in x_t} bel(x_t^*) = 1, \text{ and solve for } \eta \\
&\quad \text{Use } \eta \text{ to compute and return } bel(x_t)
\end{aligned}
\tag{1.10}
$$

1.2.3 Example

1.3 Kalman Filter

Bibliography