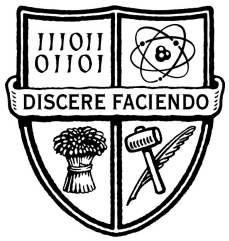

CAL POLY

California Polytechnic University
San Luis Obispo

Masters Thesis

Artificial Neural Network-Based Robotics

Justin Ng

supervised by
Dr. Andrew Danowitz

September 25, 2017

Contents

Acknowledgements **v**

Abstract **vi**

1 Introduction **1**
1.1 Definitions and Assumptions 1
1.2 Kalman Filters 1
1.2.1 Algorithm 1
1.2.2 Design 2

2 Project Plan (draft) **4**
2.1 Problem 4
2.2 Solution 4
2.3 Objectives 4
2.4 Tasks and Timeline 5

3 LaTeX Usage **6**
3.1 Figures and Ref 6
3.2 Math 7
3.3 Citations 7
3.4 Accents 7
3.5 Dashes 7
3.6 Lists 8
3.7 Groups 8
3.8 Code highlighting 8

4 Control Problem **9**

5 Training Algorithm **10**

6 Implementation **11**

7 Results **12**

8 Conclusion **13**

References **14**

Appendices **15**

List of Figures

3.1 Pad thai 6

List of Tables

Acknowledgements

I would like to thank everyone for everything.

Abstract

Note: All project files can be found at `https://www.github.com/okayjustin/roborodentia2017`

Artificial neural networks (ANNs) are highly-capable alternatives to traditional problem solving schemes due to their ability to solve non-linear systems with a non-algorithmic approach. The applications of ANNs range from process control to pattern recognition and, with increasing importance, robotics. A robot is created with multiple sensors and actuators for environment perception, movement and object manipulation while an on-board microcontroller processes the ANNs and provides the control interface. After training, the neural networks use the various sensor inputs to make the robot accomplish a series of tasks. The system demonstrates how effective and applicable ANNs are to robotic control.

Chapter 1

Introduction

1.1 Definitions and Assumptions

The robot is designed to only travel within a rectangular closed area of eight feet in the x direction and five feet in the y direction. The coordinate system is chosen as Cartesian with the origin placed at the bottom-left corner of the field. The position of the robot is always in the xy-plane since it cannot move vertically (z = 0). Therefore, x refers to the robot position along the x-axis and ranges from 0 to 8 feet, and y refers to position along the y-axis and ranges from 0 to 5 feet. Additionally, the robot can only rotate around the z-axis so θ refers to the angle of the robot in the xy-plane. Maintaining standard Cartesian coordinates, $\theta = 0^\circ$ is along the positive x direction while $\theta = 90^\circ$ is along the positive y direction.

1.2 Kalman Filters

The system relies on several different sensors to determine where it is within the environment, a problem commonly referred to as robot localization. The *Kalman Filter* (KF), an optimal state estimator, performs noise filtering and sensor fusion, the process of combining measurements from multiple sensors. The filter operates on the principles of Bayesian inference and uses statistically noisy measurements over time and knowledge of the system to produce a more accurate estimate of an unknown variable than with measurement alone.

1.2.1 Algorithm

The Kalman Filter algorithm and equations are reproduced here from Roger Labbe's excellent interactive online book [4]. The algorithm consists of two stages (not including initialization): prediction and update. During the first stage, the filter uses the current state and a process model (typically a function of time) to estimate the state in the next time step along with its uncertainty. The second stage uses sensor measurements to update the estimation by taking a weighted average based on the ratio of uncertainty between the prediction and measurement.

Initialization

Before the first run of the filter, initialize the estimated state ($\mathbf{x}$, also called the posterior) and estimated state covariance matrix ($\mathbf{P}$).

Predict

During the predict phase, the process model is used to predict the future state (known as the prior) ($\bar{\mathbf{x}}$) after one time step by summing the posterior ($\mathbf{x}$) multiplied by the *state transition function* ($\mathbf{F}$) with the control input model ($\mathbf{B}$) multiplied by the control input ($\mathbf{u}$). The covariance of prior ($\bar{\mathbf{P}}$) is larger than the posterior covariance ($\mathbf{P}$) due to uncertainty in the process model ($\mathbf{Q}$).

$$\begin{aligned}\bar{\mathbf{x}} &= \mathbf{F}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \bar{\mathbf{P}} &= \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q}\end{aligned}$$

Update

Make measurements ($\mathbf{z}$, measurement mean) and determine their accuracy ($\mathbf{R}$, measurement noise covariance). Calculate the residual (or difference) ($\mathbf{y}$) between the measurement and the product of the measurement function ($\mathbf{H}$) and the prior from the previous phase. $\mathbf{H}$ converts the prior from the state space to the measurement space. Calculate the weighting factor ($\mathbf{K}$, Kalman gain), valued between 0 and 1, based on the whether the measurement or prior is more accurate. Set the new posterior, $\mathbf{x}$, to an average of the measurement and prior, weighted by $\mathbf{K}$. Finally, update the posterior's covariance, $\mathbf{P}$, based on the measurement certainty. The algorithm then loops back to the predict phase using the newly-calculated posterior.

$$\begin{aligned}\mathbf{y} &= \mathbf{z} - \mathbf{H}\bar{\mathbf{x}} \\ \mathbf{K} &= \bar{\mathbf{P}}\mathbf{H}^T(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^T + \mathbf{R})^{-1} \\ \mathbf{x} &= \bar{\mathbf{x}} + \mathbf{K}\mathbf{y} \\ \mathbf{P} &= (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{P}}\end{aligned}$$

1.2.2 Design

The desired state variable $\mathbf{x}$ is chosen as the linear position, velocity, and acceleration in the x and y directions as well as the angular position, velocity, and acceleration about the z-axis:

$$\begin{aligned}\mathbf{x} &= [x\ y\ \theta]^T \\ \dot{\mathbf{x}} &= [\dot{x}\ \dot{y}\ \dot{\theta}]^T \\ \ddot{\mathbf{x}} &= [\ddot{x}\ \ddot{y}\ \ddot{\theta}]^T\end{aligned}$$

The process model for position and velocity:

$$\begin{cases} \bar{x} = x + \dot{x}\Delta t + \ddot{x}(\Delta t)^2 \\ \bar{\dot{x}} = \dot{x} + \ddot{x}\Delta t \\ \bar{\ddot{x}} = \ddot{x} \end{cases}$$

Which can be written in the form:

$$\begin{bmatrix} \bar{x} \\ \bar{\dot{x}} \\ \bar{\ddot{x}} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & (\Delta t)^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$$
$$\bar{\mathbf{x}} = \mathbf{F}\mathbf{x}$$

The measurement vector is chosen as:

$$\mathbf{z} = \begin{bmatrix} z_x & z_{\ddot{x}} & z_y & z_{\ddot{y}} & z_\theta & z_{\dot{\theta}} \end{bmatrix}^T$$

The measurement noise matrix is shown below. The off-diagonals are 0 because the noise between sensors is assumed to be uncorrelated.

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{\ddot{x}}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_y^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\ddot{y}}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\dot{\theta}}^2 \end{bmatrix}$$

Chapter 2

Project Plan (draft)

2.1 Problem

Create a competition robot for Roborodentia and control it using neural networks.

2.2 Solution

Design and manufacture a robot. Write firmware with a neural network implementation and train it to perform competition tasks.

2.3 Objectives

- Select robot design criteria.
- Create several solutions and prototype a few proof-of-concepts.
- Design electrical systems:
  - Power systems: Batteries, regulation, distribution.
  - Microcontroller: sufficient processing power and IO.
  - Sensors: dependent on control strategy. What information needed?
  - Actuators
- Model robot in SolidWorks with attention to manufacturing.
- Order parts and manufacture robot.
- Bring up electrical system and perform functional checks.
- Design neural network and training plan.
- Develop firmware: neural network, FSM design, communications, debug code.

- Train network and tune.
- Revise mechanical/electrical/firmware and repeat.

2.4 Tasks and Timeline

Chapter 3

LaTeX Usage

3.1 Figures and Ref

This is where I introduce stuff. See Figure 3.1.

Figure 3.1: Pad thai

3.2 Math

$$f(x) = x^2 \tag{3.1}$$

This is an equation that we don't want to number:

$$f(x) = x^2$$

Here's an in-line equation: $f(x) = x^2$.
Here's an aligned equation: Aligns at the &.

$$\begin{aligned} 1 + 2 &= 3 \\ 1 &= 3 - 2 \\ f(x) &= x^2 \\ g(x) &= \frac{1}{x} \\ F(x) &= \int_b^a \frac{1}{3}x^3 \end{aligned}$$

$$\frac{1}{\sqrt{x}}$$

$$\begin{bmatrix} a & \lambda \\ c & d \end{bmatrix}$$

3.3 Citations

Random citation [1] embeddeed in text.
Random citation [2] embeddeed in text.
Random citation [1] embeddeed in text. [3] [4]

3.4 Accents

Première $\dot{\mathrm{x}}$

3.5 Dashes

The space is 3-dimensional.
Read pages 3–4.
I saw them—there were 3 men alive
The temperature dropped to −3 degrees.

3.6 Lists

- First Level
  - Second Level
    - Third Level
      - Fourth Level

1. First level item
2. First level item
   (a) Second level item
   (b) Second level item
      i. Third level item
      ii. Third level item
         A. Fourth level item
         B. Fourth level item

3.7 Groups

will produce a paragraph that is four (this
is an easy mistake to make).

3.8 Code highlighting

```
class MyClass(Yourclass):
def __init__(self, my, yours):
bla = '5 1 2 3 4'
print bla
```

Chapter 4

Control Problem

Chapter 5

Training Algorithm

Chapter 6

Implementation

Chapter 7

Results

Chapter 8

Conclusion

References

[1] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Control Systems Magazine*, 1990. [Online]. Available: `https://web.stanford.edu/class/ee373b/NNselflearningcontrolsystems.pdf`.

[2] J. Doe, *The Book without Title*. Dummy Publisher, 2100.

[3] R. Negenborn, "Robot localization and kalman filters," PhD thesis, 2003. [Online]. Available: `http://www.negenborn.net/kal_loc/thesis.pdf`.

[4] R. R. Labbe, *Kalman and bayesian filters in python*, 2017. [Online]. Available: `https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python`.

Appendices