DOING THINGS WITH THE NAO : LIKE LASERS AND STUFF

THESIS

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

MASTER OF SCIENCE (Electrical Engineering)

at the

NEW YORK UNIVERSITY
POLYTECHNIC SCHOOL OF ENGINEERING

by

Griswald Brooks

July 2015

Approved:

_____
Advisor

_____
Date

_____
Department Head Signature

_____
Date

Copy No. # 1

University ID: N17150539

VITA

Dec, 1983 ..... Born

Jun, 2001 ..... Graduated from Inter-Lakes High School.

Sep, 2001 ..... Enlisted into the US Navy to be trained as an Electronics Technician.

Dec, 2007 ..... Seperated from US Navy with Honorable Discharge.

Jan, 2007 ..... Entered the Lakes Region Community College as a Computer Technologies major.

May, 2010 ..... Received AS Degree from the Lakes Region Community College.

Jan, 2010 ..... Entered the Polytechnic Institute of NYU as a Computer Engineering major.

May, 2013 ..... Received BS Degree cum laude from the Polytechnic Institute of NYU.

Jan, 2014 ..... Entered the NYU Polytechnic School of Engineering as a Electrical Engineering major.

May, 2015 ..... Received MS Degree from the NYU Polytechnic School of Engineering.

Dec, 2065 ..... Deceased due to Robot Apocalypse.

ACKNOWLEDGEMENTS

Thank Khorrami

Thank PK, Brian, and Agraj.

Dedicate to Mom and Dad. I totally wouldn't have even gone to this school if not for my Dad.

ABSTRACT

DOING THINGS WITH THE NAO : LIKE LASERS AND STUFF

by

Griswald Brooks

Advisor: Dr. Farshad Khorrami

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science (Electrical Engineering)

July 2015

This thesis presents the development and control of a small-scale quadruped robot platform with 16 actuated degrees-of-freedom, named "BlueFoot." The BlueFoot platform has been developed for the purpose of studying multi-terrain navigation and gait control in concert with full-body actuation, which may be used for reorienting payloads (*e.g.*, laser distance sensor and vision-sensor peripherals). This thesis will detail the design of the BlueFoot platform and its hardware sub-systems; an in-depth analysis of the system's kinematic model and robot dynamics; core Central-Pattern Generator (CPG) based gaiting algorithms introducing reflexive, feedback-driven mechanisms; and a unique foot placement and Zero-Moment Point (ZMP) posture controller based on a virtual force model and a posture feedback loop utilizing inertial measurements.

In addition, this thesis offers a method for attaining constant orientation of the trunk of a multi-legged (here a quadruped) robot in the presence of disturbances due to feet impact with the ground. This is significant when payloads (such as cameras, optical systems, armaments) are carried by the robot. The trunk is stabilized by the utilization of an on-line learning method to actively correct the open-loop gait generated by a CPG or a limit-cycle method. The learning method is based on a Nonlinear Autoregressive Neural Network with Exogenous inputs (NARX-NN)– a recurrent neural network architecture typically utilized for modeling nonlinear difference systems. A supervised learning approach is used to train the NARX-NN. The efficacy of the proposed approach is shown in detailed simulation studies of a quadruped robot.

Lastly, this thesis will present several algorithms related to navigation control, terrain modeling, and rough-terrain gait planning. In particular, algorithms for surface reconstruction and foothold planning over uneven terrain will be integral components for future developments related to the BlueFoot project. Results from simulations and actual robot trials will be presented to demonstrate the performance of these control strategies.

TABLE OF CONTENTS

VITA **ii**

LIST OF FIGURES **viii**

LIST OF TABLES **ix**

I. Introduction **1**

1.1 Central Pattern Generators for Gait Control 2

1.2 Zero Moment Point Body Placement Control 4

1.3 Trunk Stabilization 6

1.4 Potential-Fields Navigation 8

1.5 3D Surface Reconstruction for Rough Terrain Planning 11

1.6 Overview of Thesis 12

References **14**

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

Introduction

The design of legged robots and associated methods of locomotion control has been an area of interest spanning the past several decades, as shown by [1–5]. Quadruped robotic systems have gained popularity in studies pertaining to variable terrain navigation and full-body stability adaptation. Well known examples of this from the past 15 years are the Tekken [6], Kolt [7], BigDog [8], and HyQ [9] quadrupeds. Many of these systems have been implemented on a larger scale so that they can carry substantial payloads while maintaining adequate system bandwidth for fast gaits and robustness to rough terrains. Few, however, have been implemented on the scale of a hobby-robot platform while still maintaining an aptitude for rough terrain navigation and comparable sensory prowess.

The BlueFoot quadruped is a self-contained, fully-actuated platform with the dexterity to perform stabilization and repositioning maneuvers on variable terrains along the same lines as the LittleDog platform [10]. BlueFoot has been designed with sixteen actuated degrees of freedom to allow for the execution of a wide range of body and leg articulations. This level of dexterity grants the BlueFoot platform the ability to articulate its trunk (main body) over a range of poses, as well as overcome raised or uneven terrain.

BlueFoot is outfitted with several on-board vision sensors, including a LIDAR and camera, which are mounted to its trunk (main body). BlueFoot can articulate (*i.e.*, pitch and yaw) these sensors by posing its trunk using aggregate leg motion controls. BlueFoot also includes a sizable array of other on-board sensors for feedback and control, including joint position, velocity and loading sensors; an inertial measurement unit (IMU); a magnetometer; a GPS unit; humidity and temperature sensors; and foot-contact sensors. Using the computational, sensory and motor capacities at hand, BlueFoot has the ability to utilize similar control mechanisms to those implemented on larger quadruped systems.

The BlueFoot platform inherently demands active control to achieve locomotion

and system stability, making this robot an ample platform for studies related to gait design and motion planning. In particular, BlueFoot's controllers make direct use of the system's kinematic model and involve applications of open-loop gait design and stabilization for the purpose of achieving dynamic locomotion control. In particular, BlueFoot is gaited via a central pattern generator (CPG) based technique which is augmented with a foothold controller along the same lines as [11] and [12]. Active platform stabilization is performed via a zero-moment point (ZMP) based body placement controller which stabilizes the system using planar trunk motions during arbitrary gaiting sequences. Both such controllers make use of virtual-forces to drive system reference commands. These controllers apply BlueFoot's forward kinematic model for the purpose estimating joint and foot positions. Finally, outer-loop control routines are implemented to supply commands and corrections used in system navigation control. Among these controllers are a potential-fields navigation controller, which incorporates image-feature tracking; and 3D point-cloud processing routines for surface reconstruction and foot-placement planning.

1.1 Central Pattern Generators for Gait Control

As previously mentioned, BlueFoot's core gaiting routine relies on the utilization of artificial Central Pattern Generators (CPGs). This control mechanism is inspired from biological neural networks which generate rhythmic motions [13]. Biological CPGs are described in [14] as a form of self-organizing cellular neural network with the role of limited feedback in CPG networks. In fact, a key feature of these networks is that they can act without explicit sensory feedback inputs or directions from a higher-level command unit, such as a brain. Instead, signals emanating from independent motor units (and, sometimes, feedback gathered from sensory neurons) are utilized to trigger or inhibit a sequence of successive, self-coordinated motor operations. These activation sequences create loops which give rise to cyclic motion patterns.

In robotics, biological CPG's are modeled via an artificial counterpart which applies multi-state unit-oscillators to represent neural units. The dynamics of each unit oscillator in an artificial CPG network are designed to influence the dynamics of other network oscillators through tunable coupling parameters. Typically, the coupled oscillator network is implemented on a digital controller, which numerically integrates the dynamics of each neural oscillator. The output states of each unit-oscillator are used to drive selected robot degrees of freedom. Oscillator outputs could also be used for

planning periodic motions in the robot's task space, which are then translated into the joint-space references via an inverse kinematics mapping, as is done in BlueFoot's gait control routine. In implementation, specific motions are achieved through careful tuning of oscillator coupling parameters which incurs particular phase offsets between the individual limit-cycles produced by each unit-oscillator. The ability to coordinate unit-oscillator dynamics is what allows artificial CPGs to be used in the control of higher-level motor tasks, such as walking or crawling.

The selection of a unit-oscillator for use in a CPG network is a fundamental matter in CPG network design. CPG networks can be designed to employ unit-oscillators with varying oscillator dynamics (*i.e.*, varying number of states, tuning parameters, etc.) which exhibit different limit-cycle behaviors. Hopf Oscillators are used in BlueFoot's CPG implementation, as well as in [12, 15, 16]. Other unit-oscillator types which have been applied to CPGs include Van der Pol Oscillators, as detailed in [13], and the Matsuoka Neural Oscillator [17].

Studies dealing specifically with the application of CPG's to multi-legged robot gaiting (*i.e.*, quadrupeds, hexapods and octopodal robots) have been carried out in [18–27]. In particular, [13] states that the attractiveness of CPG's in the control of legged robot locomotion lies in the ability to decouple robot motor control, *i.e.*, walking, from higher-level planning, such as navigation and body-posture control. Additionally, CPG's offer an effective way for smoothly switching between gaiting patterns, such as walking, trotting, or pacing, simply through the modification of a few control parameters. Moreover, the application of artificial CPG's greatly reduces the dimensionality of the gaiting control problem for legged robots.

The specific CPG controller implemented on the BlueFoot robot allows for the generation of coordinated motions which can be modified to yield different overall motion patterns without explicitly directing the motion each joint. Moreover, BlueFoots CPG controller does not use a separate unit-oscillator to control the motion of each joint. Instead, four unit-oscillators are used to control the motion of each *foot*. In doing so, CPG outputs are mapped to stepping trajectories in the robot task-space. The resulting task-space reference commands are mapped into angular joint position references using an inverse kinematics solution for the entire robot. This approach gives way to a hybrid gaiting mechanism which combines the conveniences of CPG-based gaiting with the explicitness of strict foot trajectory planning. In BlueFoot's control scheme, foot-placement is prescribed via a separate planning mechanism which is entirely decoupled

from the CPG gait controller. This controller hybridization allows a CPG-based motion generator to be applied to gaiting over varying terrains.

Another important aspect of BlueFoot's CPG implementation is the incorporation of feedback mechanisms which modify CPG parameters. The use of feedback towards improving gaiting stability in CPG-based applications was inspired by the work of [6,17]. For our purposes, BlueFoot's CPG-based gait generation incorporates inertial feedback signals into its CPG mechanism to modify unit-oscillator amplitudes and modulate unit-oscillator frequencies. Coupling the unit-oscillator dynamics with sensory feedback gives rise to *reflexive* motions which can be tuned to help prevent the system from excessive tipping during gaiting. Reflexive motion incorporation, as it is applied into BlueFoot's CPG gaiting scheme, will be covered in more detail in Section **??**.

Due to the fact that CPG-based gaits are inherently open-loop motion control routines, a combination of auxiliary mechanisms must be used in concert with the CPG gait controller in order to ensure system stability during gaiting. The incorporation of feedback signals to modify CPG parameters aids in achieving stability although it might be insufficient for stable walking over exclusively uneven terrains. Additionally, this method requires very careful parametric tuning to perform robustly under a larger variety of terrain conditions. Thus, other means of stabilization have been incorporated into BlueFoot's gaiting routine to aid in stability.

BlueFoot's core stabilization routine applies a concept named *artificial synergy synthesis*. Using this technique, gait control is carried out independently of a stabilization control. Namely, body stabilization is performed by a subset of the robot's degrees of freedom while gaiting is carried out by the remaining [28, 29]. In original implementations of this technique, adaptations to trunk motion were utilized to stabilize the overall motion of the robot utilizing a zero moment pint (ZMP)-based approach while gaiting is controlled by a fixed-motion routine. Here, body and foot-placement are both controlled as independent, dynamic routines which supply reference commands in the robot task-space.

1.2 Zero Moment Point Body Placement Control

The zero-moment point (ZMP), which is equivalent to the center of pressure (CoP), is formally defined as a point on the ground beneath a walking system at which the net moment acting upon the trunk (referred to as the *tipping moment*) is zero [30]. The concept of ZMP and its application to legged robotics was originally introduced by [28]

and expanded upon in [31]. Both such studies apply ZMP theory towards the control of biped robots.

Formally, the ZMP can be defined using a formulation for the CoP wherein the moments about τ_x and τ_y, the lateral tipping-moments applied to the robot's body in the world frame, are equal to zero. It will be denoted p_{ZMP}. The solution for $p_{ZMP} \in \mathcal{R}^3$, with respect to a set of N foot contact points $p_{i,e} \in \mathcal{R}^3$ and N associated applied foot-contact forces $f_{i,e} \in \mathcal{R}^3$, arises as a bounded set of solutions to the equation

$$\sum_{i=1}^{N} (p_{i,e} - p_{ZMP}) \times f_{i,e} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ * \end{bmatrix} \qquad ((1.1))$$

where z-coordinate of p_{ZMP}, $[p_{ZMP}]_z$, is strictly zero when the walking surface is flat, as shown in [5]. Here, derivations for flat-surface ZMP motion will be described, but equations associated with this technique can be extended to non-flat terrain motion in a straight-forward way with a conceptually equivalent end result. The following expression is derived from dynamical equations which describe the total angular momentum, $\dot{L}$, about the legged system's center of gravity (COG), p_{COG}:

$$\dot{L} = \sum_{i=1}^{N} p_{i,e} \times f_{i,e} - m_T p_{COG} \times (\ddot{p}_{COG} + \vec{g}) \qquad ((1.2))$$

where m_T is the total mass of the legged system and $\vec{g}$ is the standard gravity vector. Assuming that all points of foot-contacts exist on a flat plane, *i.e.*, $[p_{i,e}]_z = 0 \ \forall i = \{1, ..., N\}$, and all contact force, $f_{i,e}$ are pointing upward, the p_{ZMP} of the system can be written as

$$p_{ZMP} = \frac{\sum_{i=1}^{N} (p_{i,e} \times f_{i,e})}{\left[\sum_{i=1}^{N} f_{i,e}\right]_z} = \frac{p_{COG} \times (\ddot{p}_{COG} + \vec{g}) + \dot{L} m_T^{-1}}{[\ddot{p}_{COG} + \vec{g}]_z} \in \mathcal{C}_{ZMP} \qquad ((1.3))$$

where

$$\mathcal{C}_{ZMP} = \text{conv}\,(p_{1,e}, p_{2,e}..., p_{N,e}) \qquad ((1.4))$$

with conv$(*)$ defining a convex hull generated from a set of input points $(*)$. $\mathcal{C}_{ZMP}$ is used to represent the solution space of p_{ZMP}. Moreover, $\mathcal{C}_{ZMP}$ places a bound on the angular momentum $\dot{L}$ which results from contact variations presented through $p_{i,e}$ and $f_{i,e}$. Setting $\dot{L} = 0$, a condition is defined for zero tipping. For BlueFoot's ZMP controller formulation, it is also assumed that the acceleration of the COG is sufficiently

small, *i.e.*, $\ddot{p}_{COG} \approx 0$ which yields the following, intuitive condition for minimal tipping:

$$\|p_{ZMP} - p_{COG}\| < \epsilon \qquad ((1.5))$$

where ϵ is a scalar bounding constant used to ensure p_{COG} also falls within the bounded set $\mathcal{C}_{ZMP}$. Thus, the general idea of this ZMP-based controller is to compute an approximate ZMP location and place the center of the robot's trunk (described by the translation p_b) such that the platform's COG approaches its associated ZMP for some arbitrary kinematic configuration. This is done in an effort to reduce $\left\|\dot{L}\right\|$, so as to avoid tipping about the contacting feet.

1.3 Trunk Stabilization

In addition to the aforementioned task-space controllers, a learning controller, which features the use of a NARX neural network (NARN-NN), has been studied and evaluated. In essence, this controller learns to approximate disturbance dynamics during periodic gait routines and corrects trunk orientation by administering adaptations to joint position controls. The goal of this NARX-NN based control routine is to achieve a level trunk during locomotion.

A NARX-NN architecture is used in this controller because of its known effectiveness in approximating nonlinear difference systems and making multivariate time-series predictions [32–35]. Moreover, the NARX-NN is a natural fit for a problem of this nature where the dynamics being considered are both periodic and of a high enough complexity where a nonlinear approximation method is warranted. The parallel NARX-NN model, shown in Figure **??**, is comprised of a feed-forward neural network whose input layer accepts a series of time-delayed system state values and network-output histories. The NARX-NN is trained to predict system states in the next time-instant from these inputs. Conveniently, NARX-NN training can be performed using standard error back-propagation because recurrence occurs between network inputs and outputs, and not within the hidden layers [36].

An alternative NARX-NN architecture is the series-parallel NARX-NN, in which network prediction target-values are supplied as inputs, as opposed to true network outputs. This formulation is not truly recurrent, but can be trained in the same way as the parallel NARX-NN. As such, this type of network has different convergence characteristics as compared to the parallel NARX-NN. This NARX-NN showed slower convergence

than the parallel NARX-NN for the trunk-leveling application to be presented and was not used in the final implementation.

In this controller implementation, the NARX-NN is trained to capture the effects of forces, moments and dynamical couplings that act on the trunk so that an appropriate torque inputs to the joints can be computed. These torque inputs are then used to reduce disturbance effects on trunk orientation while performing the gate. This is achieved by considering the inverse dynamics corresponding to joint motion.

NARX-NN training is perform on-line using the standard incremental back-propagation (BP) algorithm with an adapted learning rate, γ^{lr} and momentum term, μ [37,38]. This error BP algorithm [39] is a gradient-descent based method used to train a feed-forward neural network with n-layers and layer-connection matrices $\{W^1, W^2, ..., W^{n-1}\} \in W$. The BP algorithm, as used in this control approach, is as follows:

$$\begin{aligned} \Delta W^i &\leftarrow -\gamma^{lr}\delta^i \left(o^{i-1}\right)^T + \mu\Delta W^i \\ W^i &\leftarrow W^i + \Delta W^i \end{aligned} \qquad ((1.6))$$

where

$$\begin{aligned} \delta^i &= \left(\nabla_y \sigma^i \left(y^i\right)\right) e^i \\ y^i &= W^i o^{i-1} \\ e^i &= \left(W^i\right)^T \delta^{i+1} \ \forall \ i \neq n, \end{aligned}$$

and $\gamma^{lr} \in [0,1]$ defines the learning rate; $\mu \in [0,1]$ defines the learning momentum; $W^i \in \mathcal{R}^{N_O^i \times N_I^i}$, define the weighting matrix between the i^{th} layer (of size N_I^i nodes) and $(i+1)^{th}$ layer (of size $N_O^i = N_I^{i-1}$); ΔW^i, defines the corresponding weight update to W^i; and e^i is the output error for each i^{th} layer. For the output (n^{th}) layer, e^n is equal to the difference between the network output and the network output target, which will be defined later. For all other layers, e^i represents a *back-propagated* error. from the $(i+1)^{th}$ layer.

$\sigma^i(y^i)$ is an element-wise activation function which outputs a vector of activation outputs, $\sigma_j^i(y_j^i)$ for each j^{th}, weighted input, y_j^i, defined as follows:

$$\left\{\sigma^i(y^i) = \left[\sigma_1^i(y_1^i), ..., \sigma_{N_I^i}^i(y_{N_I^i}^i)\right]^T \ : \ \mathcal{R}^{N_I^i} \rightarrow \mathcal{R}^{N_I^i}\right\} \qquad ((1.7))$$

For the trunk-leveling controller described in Section **??**, a symmetric sigmoid activation function is used as a hidden-layer activation function. Formally

$$\sigma_j^i(y_j^i) \equiv \tanh(y_j^i) \in [-1,1]. \qquad ((1.8))$$

Hence, the gradient which arises for the activation mapping at each hidden layer, $\nabla_y \sigma^i(y^i)$, is defined as follows:

$$\nabla_y \sigma^i(y^i) = \begin{bmatrix} 1-(\sigma_1^i(y_1^i))^2 & 0 & \dots & 0 \\ 0 & 1-(\sigma_2^i(y_2^i))^2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & 1-\left(\sigma_{N_I^i}^i(y_{N_I^i}^i)\right)^2 \end{bmatrix} \qquad ((1.9))$$

(1.9) results from the derivative properties of the tanh $(*)$ function.For the output layer, a linear activation function is used to avoid output scaling issues. This activation function is defined simply as:

$$\sigma^O(y^O) = y^O \in \mathcal{R} \qquad ((1.10))$$

with $\nabla_y \sigma^O(y^O)$ defined as:

$$\nabla_y \sigma^O(y^O) = I_{N_I^O \times N_I^O}. \qquad ((1.11))$$

The success of this learning mechanism, as it applies to the trunk-leveling controller to be presented, is predicated on the periodicity of the system dynamics during gaiting. Like any BP-trained neural network, repetition of similar input and output sets is paramount for successful network training and, by extension, prediction accuracy. It is assumed that this specification can be met given the inherently cyclic nature of the dynamics being estimated during gaited locomotion.

1.4 Potential-Fields Navigation

The method selected for navigating the BlueFoot platform over flatland is a potential-fields control approach. This approach is described in [40] for the purpose of controlling robotic manipulators, and analyzed in-depth in [41]. In particular [41] presents shortcomings of this approach, which will be covered in-brief later in this section. Despite its pitfalls, potential-fields navigation methods offers a relatively simple and intuitive approach to robot navigation and fits well into mobile robotic tasks which involve "wandering"-type navigation over flat-regions. In particular, this approach is applied to situations where the robot has yet to acquire any knowledge of its surroundings. In the approach to be presented, potential-fields navigation is coupled with a camera-based feature tracking, which is used to guide the robot towards features of interest within the unknown environment.

The potential-fields approach is used in mobile robot navigation by moving the robot according to a guiding virtual force-vector, F_{nav} [41,42]. This vector is comprised of a sum of virtual repulsive forces, F^- (typically generated from range-sensor data), and virtual attractive forces, F^+, which pull the robot towards known goals. Moreover, F_{nav} formally defined as:

$$F_{nav} = F^+ + F^-. \qquad ((1.12))$$

The general form for the force components F^+ and F^- (represented as F_c in equation (1.13)) is as follows:

$$\begin{aligned} d_k &= p_{POI,k} - p_{robot} \\ F_c &= \alpha_F \sum_k \left(f(\|d_k\|) \frac{d_k}{\|d_k\|} \right) \end{aligned} \qquad ((1.13))$$

where $p_{POI,k}$ and p_{robot} represent the position of each k^{th} point-of-interest (POI) and the position of the robot platform; $f(*)$ is a potential function which returns a scalar potential factor with respect to a scalar argument $(*)$; and α_F is a scaling parameter which is positive when the POIs considered represent goals and negative when POIs represent obstacles to avoid. The potential function, $f(*)$, and scaling factor, α_F, are designable for particular applications. For BlueFoot's navigation scheme, attractive and repulsive forces are generated using a consolidated, piecewise forcing function which is used to guide through an environment where a goal is not specified before hand.

According to [41] the main pitfall with potential-fields navigation is susceptivity to local minima within the global force-field. At a local minimum, the F^+ and F^- are of nearly equal magnitude, causing the magnitude of the total guiding force vector, F_c, to be close to zero. In practice, reaching a point at which robot will be completely stationary is unlikely, as sensor readings used to observe environmental obstructions are corrupted by noise. In turn, this noise induces random perturbations in the robot's motion. In fact, perturbations due to sensor noise could actually aid in relieving a situation in which a robot is stuck in a local minima. However, the gradient of the force-field around a minimum point could be very steep over a large area around the aforementioned singularity. These type of potential-sink regions could cause the robot to exhibit limit-cycle behavior as it periodically overshoots and re-attracts to the location of the force-minimum.

This can be overcome, in part, by adding an artificial *inertia* (essentially a tunable gain parameter) when updating the robot's navigation reference signals. Given a set

of robot navigation command parameters, v^r and ω^r, and potential reference outputs, v_L^r and ω_L^r (which are generated from F_{nav}), navigation updates with an added update-inertia, $B^r \in \mathcal{R}^{2\times 2}$, exhibit the following controller dynamics:

$$\begin{bmatrix} \dot{v}^r \\ \dot{\omega}^r \end{bmatrix} = B^r \left(\begin{bmatrix} v_L^r \\ \omega_L^r \end{bmatrix} - \begin{bmatrix} v^r \\ \omega^r \end{bmatrix} \right) \qquad ((1.14))$$

where B^r is a strictly positive definite, diagonal "inertia" matrix. This control formulation is equivalent to an outer-loop P-control scheme. Using this update scheme may cause the robot platform to sufficiently overshoot minimum points such that it leaves the local attraction field. Care must be taken in the selection of B, however, so that system still exhibits stable behavior during navigation control. A more sophisticated approach to escaping local minima is mentioned in [43], which involves a "stuck" detection algorithm. The idea behind such an algorithm involves a deduction about whether or not the robot is captured in a local minima based on samples of the robots motion state (*i.e.*, position and velocity). Once the robot has determined that is it stuck, it executes a small random-walk as a means of escape.

The local minima problem is addressed through the use of auxiliary (possibly dynamic) target points, as done in [42] via a hybrid navigation potential-fields/visual-servoing scheme. Instead of incorporating goal points directly into the potential-fields scheme, goals (in this case, image features) are tracked using an entirely separate navigation scheme. A separate set of navigation reference commands, v_C^r and ω_C^r, are mixed with commands generated via the potential-fields algorithm. The amount of influence either command scheme has over the final navigation command parameters, v^r and ω^r, depends on relative measure of "closeness" to the object being tracked, which the robot determines from an image processing routine. The specifics of the visual-servoing controller will be described in **??**.

Hence, the potential-fields portion of this control scheme is used only for the purpose of avoiding potential obstructions, sensed via LIDAR range data. Image-features are used in a visual-servoing routine which guides the robot toward features-of-interest which fall within the robots camera gaze. This approach offers the ability to manually guide the robot during navigation, by either a human overseer, or a partner robot (which could wear trackable markers), as it performs an independent obstacle avoidance routine. The advantage of this approach lives in its simplicity, as it relies only on immediate environmental samples and, thus, has relatively minimal implementation demands. As a mechanism for partially-guided wandering-type (random) navigation within an unknown

region, this approach is certainly adequate as a will be shown via empirical results from real-world trials.

1.5 3D Surface Reconstruction for Rough Terrain Planning

The previously introduced navigation scheme is utilized for navigation over flat-land, exclusively. As such, it is generally insufficient for navigation and planning over rough terrain. Navigation and footstep planning over rough terrain require the acquisition of 3D surface data from the robots immediate environment, generally with high feature detail. This thesis will provide the preliminaries for rough-terrain navigation by way of several surface reconstruction/representation methods from point-cloud data. First, a method for composing 3D point clouds from successive 2D LIDAR scans will be described. Then, algorithms for generating 3D height-maps will be described and a method for cost assignment based on generated height-maps will be presented for use in rough terrain foot-placement planning. Finally, an implementation for surface reconstruction from raw point-cloud data will be described.

Work related to height-map generation and associated cost-assignment (used in generating discrete cost-maps) from 3D point cloud data is formulated from an intuitive evaluation of the problem of rough-terrain planning. This work should be viewed as the initial steps towards formal rough terrain planning which will later be solidified with existing research. On the other hand, implementations related to 3D surface reconstruction from 3D point clouds rely heavily on algorithms originally outlined in [44] and formally implemented in the OpenPCL library [45].

Surface estimation from 3D point cloud data is approached by way of successive plane-fitting (normal estimation) via Principle Component Analysis (PCA) on a moving subset of points [44, 46]. Specifically, a normal vector is associated with each point, $\bar{x}_i$, within the point cloud, $\bar{\mathcal{S}}$, by fitting a plane to a subset of neighboring points which fall within a unit-ball (of radius d_s) around $\bar{x}_i$. Here, $\bar{\mathcal{S}}$ represents a raw 3D point cloud. This point cloud is pre-conditioned to generate a sparser point cloud, $\hat{\mathcal{S}}$, which estimates the collection of points $\bar{\mathcal{S}}$. Preconditioning involves a voxel-grid filter, which is used to down-sample the original point cloud; and a moving-least-squares (MLS) filter, which is used to regularize points and remove outliers. These preconditioning routines aid in the normal estimation process by decreasing computational burden of the estimation algorithm. This is achieved by reducing the density of the space which must be searched

for nearest neighbor points about each $\bar{x}_i$. Preconditioning also improves the relative smoothness of the reconstructed surface [44].

To estimate a local plane (via PCA) about the point $\bar{x}_i$, we define a moving-neighborhood, $\mathcal{B}_{\bar{x}_i}$, as follows:

$$\{\bar{x}_j \in \mathcal{B}_{\bar{x}_i} \ : \ \|\bar{x}_j - \bar{x}_i\| < d_s \ , \ \forall \ i \neq j \ \} \ , \ \mathcal{B}_{\bar{x}_i} \subset \bar{S}. \qquad ((1.15))$$

Additionally, the covariance matrix, $C_{\bar{x}_i} \in \mathcal{R}^{3\times 3}$, of the subset $\mathcal{B}_{\bar{x}_i}$ is defined as:

$$C_{\bar{x}_i} = \frac{1}{S(\mathcal{B}_{\bar{x}_i})} \sum_{\bar{x}_j \in \mathcal{B}_{\bar{x}_i}} (\bar{x}_j - \bar{x}_{c,i})(\bar{x}_j - \bar{x}_{c,i})^T \qquad ((1.16))$$

where the centroid, $\bar{x}_{c,i} \in \mathcal{R}^3$, of the subspace $\mathcal{B}_{\bar{x}_i}$ is defined as:

$$\bar{x}_{c,i} = \frac{1}{S(\mathcal{B}_{\bar{x}_i})} \sum_{\bar{x}_j \in \mathcal{B}_{\bar{x}_i}} \bar{x}_j \qquad ((1.17))$$

and $S(\mathcal{B}_{\bar{x}_i})$ defines the number of points within $\mathcal{B}_{\bar{x}_i}$. Note that $C_{\bar{x}_i}$ is a positive semi-definite matrix with all eigenvalues, $\lambda_{x_i,j} \geq$ for $j \in \{1,2,3\}$ such that $\lambda_{x_i,1} \leq \lambda_{x_i,2} \leq \lambda_{x_i,3}$. The unit-eigenvectors $v_{x_i,j}$ for $j \in \{1,2,3\}$ represent the principal components of the local subspace $\mathcal{B}_{\bar{x}_i}$, and are defined by:

$$\left\{v_{x_i,j} \ \mathcal{R}^3 \ : \ C_{\bar{x}_i} v_{x_i,j} = \lambda_{x_i,j} v_{x_i,j} \ \forall \ j \in \{1,2,3\}\right\}. \qquad ((1.18))$$

The eigenvector $v_{x_i,1}$ with corresponding smallest eigenvalue $\lambda_{x_i,1}$ represents a unit-normal which emanates from a 2D manifold of $\mathcal{B}_{\bar{x}_i}$ whose origin is located at x_i. This manifold represents a planar least-squares best-fit for the information contained in $\mathcal{B}_{\bar{x}_i}$. To fortify this definition, we can also consider that the minimum eigenvalue $\lambda_{x_i,1}$ indicates that $v_{x_i,1}$ is a direction in $\mathcal{B}_{\bar{x}_i}$ with the lowest variational *energy*. Hence, the sum of the square-errors among all points in $\mathcal{B}_{\bar{x}_i}$ is minimized in the $v_{x_i,1}$ direction.

Given a preprocessed point cloud $\hat{\mathcal{S}}$ (generated from a raw point cloud $\bar{\mathcal{S}}$), the full normal-estimation algorithm is then implemented using the PCA formulation defined from (1.15)-(1.18) in Algorithm 1, to follow. Here, $\vec{n}_i \in \hat{N}_{\mathcal{S}}$ represents a normal emanating from each i^{th} point, $\hat{x}_i \in \hat{\mathcal{S}}$.

1.6 Overview of Thesis

This thesis will first detail the major hardware components; design considerations; and construction of the BlueFoot platform in Chapter **??**. Next, BlueFoot's software

Algorithm 1 Surface-normal estimation from a preprocessed 3D point cloud.

init: $\hat{\mathcal{S}}, \hat{N}_{\mathcal{S}} = \phi$

for all $\hat{x}_i \in \hat{\mathcal{S}}$ **do**

$\mathcal{B}_{\bar{x}_i} = \text{getNearestNeighbors}\left(\hat{\mathcal{S}}, \hat{x}_i, d_s\right)$

$\vec{n}_i = \text{performPCA}\left(\mathcal{B}_{\bar{x}_i}\right)$

$\hat{N}_{\mathcal{S}} \leftarrow \hat{N}_{\mathcal{S}} \bigcup \vec{n}_i$

end for

and processing architecture, applied in the control the BlueFoot platform, will be described in Chapter **??**. The kinematic and dynamical models of the BlueFoot system will be described in Chapter **??**, followed by descriptions of the controllers which are presently implemented to gait and stabilize the platform in Chapter **??**. Chapter **??** will cover navigation algorithms applied to the BlueFoot platform, including preliminary approaches for rough-terrain planning. Finally, Chapter **??** will contain concluding statements about the system's design and control, including remarks about possible future directions of study related to the BlueFoot platform and legged robotics as extensions of the existing work.

REFERENCES

[1] R.B. McGhee and A.A. Frank. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3(0):331 – 351, 1968.

[2] J.K. Hodgins and M. Raibert. Adjusting step length for rough terrain locomotion. volume 7, pages 289–298, Jun 1991.

[3] R. Altendorfer, N. Moore, H. Komsuoglu, M. Buehler, Jr. Brown, H.B., D. McMordie, U. Saranli, R. Full, and D.E. Koditschek. RHex: A Biologically Inspired Hexapod Runner. volume 11, pages 207–213. Kluwer Academic Publishers, 2001.

[4] J.Z. Kolter, M.P. Rodgers, and A.Y. Ng. A control architecture for quadruped locomotion over rough terrain. In *IEEE International Conference on Robotics and Automation*, pages 811–818, May 2008.

[5] Tedrake R. Kuindersma S. Wieber, P.-B. Modeling and control of legged robots. In Siciliano and Khatib, editors, *Springer Handbook of Robotics, 2nd Ed*, Lecture Notes in Control and Information Sciences, chapter 48. Springer Berlin Heidelberg, 2015.

[6] Y. Fukuoka, H. Kimura, Y. Hada, and K. Takase. Adaptive dynamic walking of a quadruped robot 'Tekken' on irregular terrain using a neural system model. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 2037–2042 vol.2, Sept 2003.

[7] Joaquin Estremera and Kenneth J. Waldron. Leg Thrust Control for Stabilization of Dynamic Gaits in a Quadruped Robot. In Teresa Zieliska and Cezary Zieliski, editors, *Romansy 16*, volume 487 of *CISM Courses and Lectures*, pages 213–220. Springer Vienna, 2006.

[8] Marc Raibert. BigDog, the Rough-Terrain Quadruped Robot. In Myung J. Chung, editor, *Proceedings of the 17th IFAC World Congress, 2008*, volume 17.

[9] C. Semini. *HyQ Design and Development of a Hydraulically Actuated Quadruped Robot*. PhD thesis, Apr 2010.

[10] J.R. Rebula, P.D. Neuhaus, B.V. Bonnlander, M.J. Johnson, and J.E. Pratt. A Controller for the Littledog quadruped walking on Rough Terrain. In *IEEE International Conference on Robotics and Automation*, pages 1467–1473, April 2007.

[11] M. Ajallooeian, S. Pouya, A Sproewitz, and A.J. Ijspeert. Central Pattern Generators augmented with virtual model control for quadruped rough terrain locomotion. In *IEEE International Conference on Robotics and Automation*, pages 3321–3328, May 2013.

[12] Simon Rutishauser, A Sprowitz, L. Righetti, and A.J. Ijspeert. Passive compliant quadruped robot using Central Pattern Generators for locomotion control. In *2nd IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 710–715, Oct 2008.

[13] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural networks*, 21(4), 2008.

[14] P. Arena. The central pattern generator: a paradigm for artificial locomotion. *Soft Computing*, 4(4):251–266, 2000.

[15] L. Righetti and A.J. Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 1585–1590, May 2006.

[16] Vitor Matos and Cristina P. Santos. Omnidirectional locomotion in a quadruped robot: A CPG-based approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3392–3397, Oct 2010.

[17] G. Endo, J. Morimoto, J. Nakanishi, and G. Cheng. An empirical exploration of a neural oscillator for biped locomotion control. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE Int. Conf. on*, volume 3, pages 3036–3042 Vol.3, April 2004.

[18] Paolo Arena. A mechatronic lamprey controlled by analog circuits. In *MED'01 9th Mediterranean Conference on Control and Automation*. IEEE, June 2001.

[19] Bernhard Klaassen, Ralf Linnemann, Dirk Spenneberg, and Frank Kirchner. Biomimetic walking robot scorpion: Control and modeling. *Robotics and Autonomous Systems*, 41(23):69 – 76, 2002. Ninth International Symposium on Intelligent Robotic Systems.

[20] P. Arena, L. Fortuna, M. Frasca, and G. Sicurella. An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion. *Systems, Man,*

and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 34(4):1823–1837, Aug 2004.

[21] Shinkichi Inagaki, Hideo Yuasa, and Tamio Arai. {CPG} model for autonomous decentralized multi-legged robot systemgeneration and transition of oscillation patterns and dynamics of oscillators. *Robotics and Autonomous Systems*, 44(34):171 – 179, 2003. Best papers presented at IAS-7.

[22] Shinkichi Inagaki, Hideo Yuasa, Takanori Suzuki, and Tamio Arai. Wave {CPG} model for autonomous decentralized multi-legged robot: Gait generation and walking speed control. *Robotics and Autonomous Systems*, 54(2):118 – 126, 2006. Intelligent Autonomous Systems 8th Conference on Intelligent Autonomous Systems (IAS-8).

[23] A. Billard and A.J. Ijspeert. Biologically inspired neural controllers for motor control in a quadruped robot. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 6, pages 637–641 vol.6, 2000.

[24] Giorgio Brambilla, Jonas Buchli, and AukeJan Ijspeert. Adaptive four legged locomotion control based on nonlinear dynamical systems. In Stefano Nolfi, Gianluca Baldassarre, Raffaele Calabretta, JohnC.T. Hallam, Davide Marocco, Jean-Arcady Meyer, Orazio Miglino, and Domenico Parisi, editors, *From Animals to Animats 9*, volume 4095 of *Lecture Notes in Computer Science*, pages 138–149. Springer Berlin Heidelberg, 2006.

[25] J. Buchli, F. Iida, and A.J. Ijspeert. Finding resonance: Adaptive frequency oscillators for dynamic legged locomotion. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3903–3909, Oct 2006.

[26] K. Tsujita, K. Tsuchiya, and A. Onat. Adaptive gait pattern control of a quadruped locomotion robot. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 4, pages 2318–2325 vol.4, 2001.

[27] K. Tsujita, H. Toui, and K. Tsuchiya. Dynamic turning control of a quadruped robot using oscillator network. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 3, pages 2613–2618 Vol.3, April 2004.

[28] Miomir K. Vukobratovi. Contribution to the study of anthropomorphic systems. *Kybernetika*, 08(5):(404)–418, 1972.

[29] J.-I. Yamaguchi, A. Takanishi, and I. Kato. Development of a biped walking robot compensating for three-axis moment by trunk motion. In *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 1, pages 561–566 vol.1, Jul 1993.

[30] P. Sardain and G. Bessonnet. Forces acting on a biped robot. center of pressure-zero moment point. *Trans. Sys. Man Cyber. Part A*, 34(5):630–637, September 2004.

[31] A. Goswami. Foot rotation indicator (fri) point: a new gait planning tool to evaluate postural stability of biped robots. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 47–52 vol.1, 1999.

[32] Tsungnan Lin, B.G. Horne, P. Tino, and C.L. Giles. Learning long-term dependencies in NARX recurrent neural networks. *Neural Networks, IEEE Transactions on*, 7(6):1329–1338, Nov 1996.

[33] Grant P. M. Chen S., Billings S. A. Non-linear system identification using neural networks. In *Int. Journal of Control*, volume 51 of *Lecture Notes in Control and Information Sciences*, pages 1191–1214. Taylor and Francis, 1990.

[34] Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies, 1996.

[35] S. A Billings. *Nonlinear system identification : NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley and Sons Ltd, 2013.

[36] Oliver Nelles. Neural networks with internal dynamics. In *Nonlinear System Identification*, pages 645–651. Springer Berlin Heidelberg, 2001.

[37] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[38] David E. Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation. chapter Backpropagation: The Basic Theory, pages 1–34. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

[39] Raúl Rojas. The backpropagation algorithm. In *Neural Networks: A Systematic Introduction*, chapter 7. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[40] N. Hogan. Impedance control: An approach to manipulation. In *American Control Conference, 1984*, pages 304–313, June 1984.

[41] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1398–1404 vol.2, Apr 1991.

[42] F. Arambula Coso and M.A. Padilla Castaeda. Autonomous robot navigation using adaptive potential fields. *Mathematical and Computer Modelling*, 40(910):1141 – 1156, 2004.

[43] P. Krishnamurthy and F. Khorrami. Godzila: A low-resource algorithm for path planning in unknown environments. *Journal of Intelligent and Robotic Systems*, 48(3):357–373, 2007.

[44] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[45] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[46] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.