Lukesky, the Walker

A Lego Mindstorm NXT 2.0 Robot

Missouri S&T

Gary Steelman

4/30/2011

Contents

[Background 4](#_Toc291894146)

[Project Goals 5](#_Toc291894147)

[Project Goals Accomplished 6](#_Toc291894148)

[Robot Specifications 7](#_Toc291894149)

[Locomotion Mechanism 7](#_Toc291894150)

[Legs 7](#_Toc291894151)

[Motors 8](#_Toc291894152)

[Gears 8](#_Toc291894153)

[Vision Mechanisms 8](#_Toc291894154)

[Color Sensor 8](#_Toc291894155)

[Ultrasonic Sensor 8](#_Toc291894156)

[Visual Output 9](#_Toc291894157)

[Implementation Details 10](#_Toc291894158)

[3D Library 10](#_Toc291894159)

[Features 10](#_Toc291894160)

[Definitions 10](#_Toc291894161)

[Optimizations Specific to NXC 11](#_Toc291894162)

[3D Library Use for Movement Logic 13](#_Toc291894163)

[Lukesky’s Reference Frame 13](#_Toc291894164)

[Translation 13](#_Toc291894165)

[Rotation 13](#_Toc291894166)

[Movement between Two Arbitrary Points 13](#_Toc291894167)

[Technical Challenges 14](#_Toc291894168)

[Conclusion 15](#_Toc291894169)

[Future Work 16](#_Toc291894170)

[Acknowledgements, Sources, and References 17](#_Toc291894171)

[Acknowledgements 17](#_Toc291894172)

[Sources and References 17](#_Toc291894173)

[Appendix 18](#_Toc291894174)

[Pictures of Lukesky 18](#_Toc291894175)

[Code 18](#_Toc291894176)

# 

# Background

# Project Goals

The main goal of this project is to exhibit implementation of the mathematical concepts demonstrated in classroom lecture. These concepts include:

* Euclidian Spatial Descriptions
  + Use of 4x4 matrices called *reference frames* to represent the orientation and location of a 3-Dimensional Coordinate Axis in Euclidian space.
* Coordinate Transformations
  + Pre or post multiplication of reference frames to create chains of reference frames to rotate and translate reference frames.
* Manipulator Kinematics
  + Use of the Denavit-Hartenberg (DH) Convention with reference frames to define the link and joint chains of a system.
* Inverse Kinematics
  + Utilizing trigonometric expressions to read the orientation of a reference frame in real time.
  + Use of Euler Angles to define the transformation angles necessary to obtain a specified robotic wrist orientation from another orientation.
* Jacobians
  + Use of a skew symmetric matrix’s properties to derive the linear and angular velocity of a reference frame in real time.
* Path Planning and Trajectory
  + Use of potential field path planning wherein obstacles are represented as repulsive fields and targets are represented as attractive fields.
  + Use of quintic polynomials to create paths with smooth transitions from one segment to another.
* Robot Vision
  + Application of filters to images to manipulate the image for mathematical use.
* Cams, Gears, and Mechanisms
  + Use of cams, gears, and mechanisms to achieve repetitive motion in a system while minimizing stress on actuators in the system.

To embody these concepts in a project form, the following goals for a robot were created:

1. The robot would have locomotive ability through walking rather than driving.
2. The robot would seek a specified color source then seek another source of the same color.
3. The robot would avoid obstacles in its path.
4. The robot would return to the place it started searching from after a certain amount of time.

Use of all concepts mentioned would necessitate a large project. Due to time constraints only a partial number of the concepts were implemented. For formulae and mathematical expressions embodying the concepts implemented for this project, see the Robot Specifications section.

# Project Goals Accomplished

The following concepts listed in the Project Goals section were implemented:

* Euclidian Spatial Descriptions
  + Lukesky tracks his position and orientation using reference frames. Even though Lukesky only has the ability to move in a 2D plane, the reference frames used are still 3D frames.
* Coordinate Transformations
  + To track his position and orientation, Lukesky multiplies his current position’s reference frame with a transformation frame to reach the destination frame.
* Manipulator Kinematics
  + The DH Convention was used to define Lukesky’s body. The scope of this project only required that one frame be defined; though theoretically it could be useful for one frame to be defined for the end of each of the six legs.
* Inverse Kinematics
  + In order for Lukesky to utilize the reference frame system properly, it is necessary to extract the angles off of the world reference frame easily. This is implemented using trigonometric expressions derived specially for Lukesky. These angles allow us to easily detect the orientation of Lukesky at any time.
* Cams, Gears, and Mechanisms
  + A chain of 2 gears is used for each motor to decrease the output speed and increase the output torque such that the legs operate smoothly and have enough power to move Lukesky.
  + The leg mechanism is generated from a four-bar design so that the repetitive forward motion can be generated from a single motor.

For more information on the exact implementation of each of these concepts, see the Implementation Details section.

The following goals were achieved using the concepts implemented:

1. The robot would have locomotive ability through walking rather than driving.
   * The leg mechanism gives Lukesky the ability to walk.
2. The robot would return to the place it started searching from after a certain amount of time.
   * The use of reference frames with coordinate transformations and forward and inverse kinematics allows Lukesky to return to the initial point at any time.

# Robot Comopnents

## Locomotion Mechanism

### NXTAPOD_tmb.jpgLegs

Lukesky is an adaptation of Daniele Benedettelli’s NXTAPOD. [1]

“[The NXTAPOD is a] simple hexapod robot that can walk and reverse-turn using **just one motor**. It can detect obstacles using the ultrasonic sensor.”

The NXTAPOD’s locomotive abilities are based on a four-bar mechanism and produce forward motion when the legs are driven exactly one-half motor rotation out of phase. Its locomotive abilities are similar to how four-legged animals walk. By driving forward legs in opposite positions, forward locomotion is produced. See Figure 2 below for details.

Figure - Daniele Benedetelli’s NXTAPOD. [1]

1 4  
2 5  
3 6

*Initial Position After 1 Step After 2 Steps After 3 Steps***Figure 2 - Leg positions through a few steps of the NXTAPOD.**

The red circles indicate which of the NXTAPOD’s six legs are in contact with the ground for each step. The legs on the left hand side are numbered 1, 2, 3 and legs on the right hand side are numbered 2, 4, 6. To step, the legs that will contact the ground in the next step are rotated forward or backward and when placed on the ground, lift the legs that previously on the ground. This cycle is repeated and thus forward or backward locomotion is achieved. Thus, the stepping equation is:

Where is the set of all six legs and is the set of legs in contact with the ground this step. At any given time only three of the six legs are in contact with the ground, as shown in Figure 2.

### Motors

The NXTAPOD has only forward locomotion and rightward rotation ability. Put simply, the NXTAPOD walks forward until it encounters an object, then utilizing a special mechanism, rotates to its right and continues to walk forward. This range of motion is inadequate to achieve the Project Goals. Thus the single motor driven design of the NXTAPOD was insufficient. The NXTAPOD design was modified to include the twin-motor system:

Where *Motor B* drives the left side legs and *Motor A* drives the right side legs. The motors can be driven entirely *independently* from each other. To accommodate the twin-motor system, Lukesky is wider and heavier than the NXTAPOD.

### Gears

For smooth locomotion, it was necessary to significantly alter the rotation speed of the NXT 2.0 motors. The motors at 100% of their possible maximum speed moved far too rapidly for the NXTAPOD to follow the Stepping Equation without large amounts of error. The servo-motors were also incapable of producing a satisfactory amount of torque for locomotion. For this reason, the stepping mechanism includes two different gears for each motor:

*Gear 1* is placed along the axle through each motor and rotates in at a 1:1 angular velocity with the motor. *Gear 2* is placed in contact with *Gear 1*, but as part of a gear chain. Thus the gear ratios for the output are:

This generates more torque and slower angular velocity for the motors, which was necessary for smooth locomotion.

## Vision Mechanisms

### Color Sensor

It was intended for the light color sensor to be equipped onto Lukesky, however, due to time constraints, no behavioral algorithm was developed to use the input from this sensor. The sensor was thus omitted from the final design.

### Ultrasonic Sensor

As with the NXTAPOD, Lukesky is equipped with the NXT 2.0 Ultrasonic Sensor. This sensor provides Lukesky with the ability to detect objects directly in front of him in real time.

## Visual Output

Lukesky is capable of outputting his current position and orientation to the LCD screen on the NXT 2.0 Brick. This output represents the *reference frame* for his current position and orientation.

# Implementation Methodology

## 3D Library

### Features

To accurately track Lukesky’s position in space, a reusable 3D reference frame library was created using the NXC programming language [2] and the BricxCC IDE [3]. The NXC programming language is a language that aims to emulate C for the Lego Mindstorm NXT 2.0. The NXC source code for the library can be found in the Code - 3D Library section of the Appendix. The most useful features are listed here:

* Create a 3D reference frame.
* Translate a 3D reference frame.
  + Relative to Lukesky’s current position.
  + Relative to the constant world frame.
* Rotate a 3D reference frame.
  + About any axis, X, Y, or Z.
  + Relative to Lukesky’s current orientation.
  + Relative to the constant world frame.
* Set a reference frame to the equivalent of the world frame rotated by some degrees about the X, Y, or Z axis.
* Multiply two reference frames.
* Calculate the distance between two reference frames.
* Output a reference frame to the LCD screen of the NXT 2.0 Brick.

### Definitions

A 3D reference frame, A, is defined as:

Where the first column, represented as , is the X axis of the reference frame defined as a vector with three components, each in the direction of the world X, Y, and Z axes. The second column represents the reference frame’s Y axis in its three components and the third column represents the reference frame’s Z axis in its three components. The fourth column represents the location (x, y, z) relative to the world reference frame:

As can be seen, the world reference frame essentially represents the *origin* and the direction of each of the X, Y, and Z axes for the world.

To accomplish a rotation or a translation, the library essentially multiplies a reference frame by the appropriate rotation or translation frame.

The translation reference frames are:

; ;

Where means a translation along the reference frame’s X-axis by a number a. The translations for the Y-axis and Z-axis are defined analogously.

The rotation reference frames are:

; ;

Where means a rotation about the reference frame’s X-axis by a *radian* value α and represents the cosine of the angle alpha and represents the sin of the angle alpha.

### Optimizations Specific to NXC

In general, a 3D reference frame would be a 4x4 matrix. However, for the scope of this project the fourth row of the reference frame is essentially useless as it is *always* . To reduce memory consumption by sixteen bytes per frame and reduce the number of operations per translation or rotation, the last row of the reference frame was omitted. This change is visualized as thus:

Additionally, to improve library performance for the NXC language, multiple optimizations were implemented that would not necessarily be more efficient for a standard C implementation:

* For each operation that required accessing an index of a reference frame, ie A[0], the elements were accessed only once and their values stored in a local temporary variable, ie A0.
* For each trigonometric operation, ie, , the value was stored in a local temporary variable, ie, stheta. Since the rotation matrices utilize the negation of trigonometric functions, ie, , having the value of stored locally allowed omission of another call to the sin function just by storing -stheta.
* Storing the values for the trigonometric functions not only allowed less calls to calculate trigonometric values, but also allowed use of the NXC function ArrayBuild(). This function is the fastest way to create or store multiple values in an array.
* Simple functions, like initialization or copying of a reference frame were defined using the #define preprocessor macro to avoid additional function calls on the stack.
* Each time a reference frame was set equal to another reference frame, ie, A=B, the assignment was made exactly as A=B. Assignment in this fashion generates a single MOV assembly call. This style of assignment is different from standard C. In standard C a memcpy() call would have to be used.

These optimizations allow a rotation or a translation operation in approximately four milliseconds.

## 3D Library Use for Movement Logic

### Lukesky’s Reference Frame

Lukesky’s movement mechanics restrict movement to a 2-Dimensional plane. However, the 3-Dimensional library can still be utilized easily. One frame was assigned to represent Lukesky’s position and orientation in 3D space.

* The frame is positioned directly in the center of Lukesky.
* The positive Z-axis, , is oriented pointing forward; the negative Z-axis, , is oriented pointing backward.
* The positive X-axis, , is oriented pointing to Lukesky’s left; the negative X-axis, , is oriented pointing to the right.
* The positive Y-axis, , is oriented pointing above Lukesky; the negative Y-axis, , is oriented pointing below Lukesky, directly at the ground.

The reference frame can be seen overlaid on Lukesky in the Figure below.



It can be seen that movement takes place in the plane defined by both the X and Z axes and that locomotion takes place along the Z-axis. Turns in the XZ plane are accomplished by rotating about the Y-axis.

## Movement between Two Arbitrary Points

## Translation

## Rotation

# Technical Challenges

# Conclusion

# Future Work

# Acknowledgements, Sources, and References

## Acknowledgements

Most of these optimizations come at the suggestion of John Hansen (bricxcc@comcast.net) the creator of NXC. His assistance in understanding and using NXC was invaluable during this project.

Thanks to Rebecca Carrender for help figuring out part of the facing algorithm.

## Sources and References

1. <http://robotics.benedettelli.com/NXTAPOD.htm>
2. <http://bricxcc.sourceforge.net/nbc/>
3. <http://bricxcc.sourceforge.net/>

# Appendix

## Pictures of Lukesky

## Code