

Jim: An Intelligent Continuous-Motion Frame-walking Robot

Richard S. LaBarca, Gabriel F. Brisson, Peter Sand,
Jonathan W. Hurst, Carlos F. Reverte, Michelle Ungerer
Carnegie Mellon University Robotics Institute Robotics Club

Copyright © 1998 Carnegie Mellon University Robotics Institute.

ABSTRACT

This paper will describe the mechanics, electronics and software of a continuous-motion frame-walking robot named Jim, and the design concepts behind a robust, reusable control infrastructure used in its design. Three challenges will be addressed:

- Design a continuous-motion frame-walker, a frame-walker that keeps most of its mass in motion through its entire walking gait, while maintaining good repeatability
- Design a sensing system that provides navigation aid as well as compensates for any loss of dead reckoning accuracy due to this continuous motion
- Design a control architecture, which provides simulation, planning, and manipulation, and is robust and general enough to port to other robotic systems

INTRODUCTION

A frame-walker, that is, a robot that achieves motion by rotating and translating two sets of independently supporting structures with respect to each other, is generally a reliable design paradigm for walking machines. In the SAE Walking Machine Decathlon[†], given an average development time of less than one year, frame walkers have proved to be ideal competitors since their inherent static stability requires relatively little effort in designing kinematic models and autonomous control algorithms. Statically stable frame-walkers can also easily be controlled manually in case of a computer failure, algorithm flaw or lack of available computer scientists, and naturally lend themselves to a dead reckoning approach to navigation. The frame-walker paradigm is not, however, the best choice in general for a walking robotic platform due to the fact that static

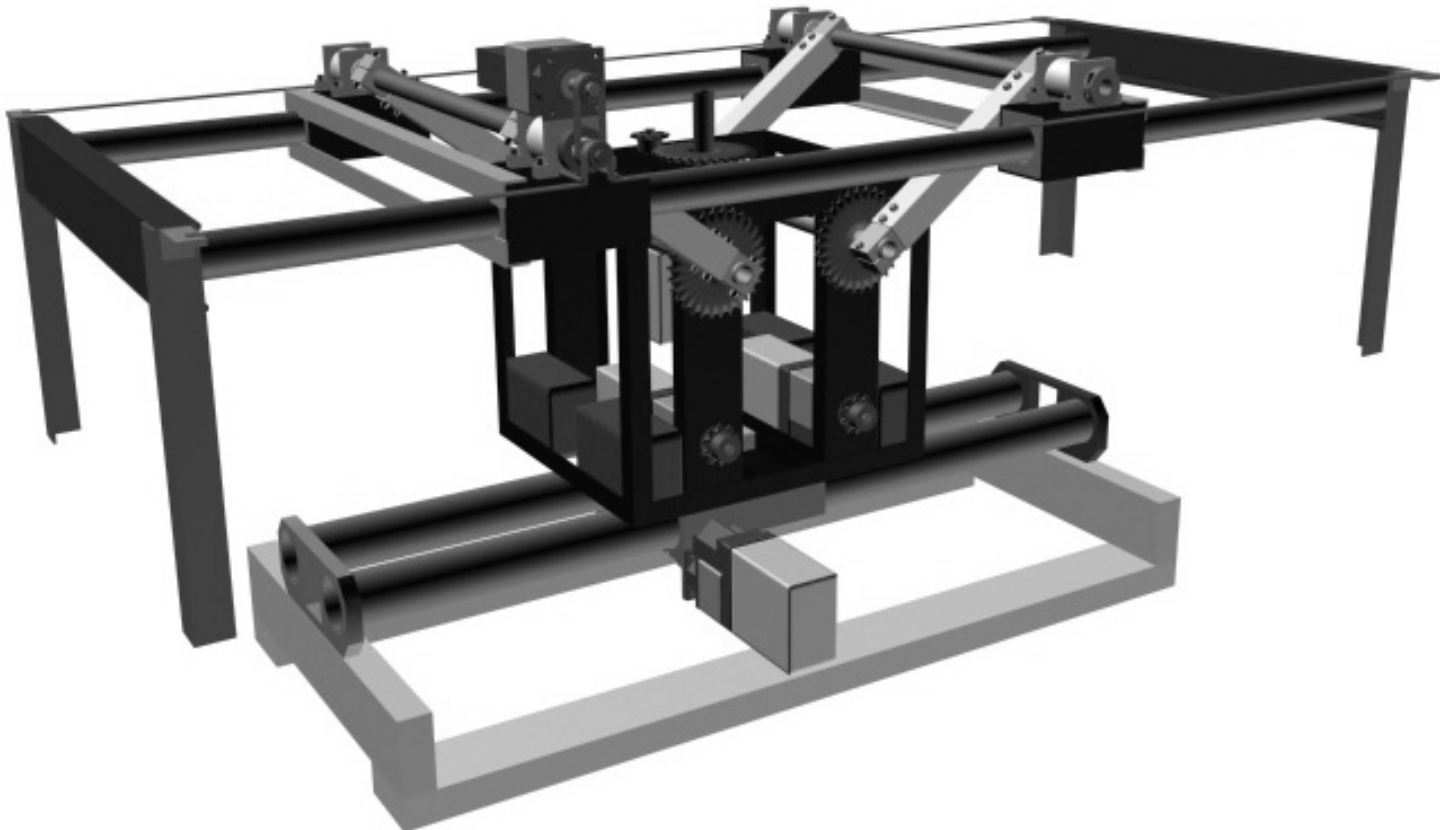


Figure 1: Jim mechanical view

stability and repeatability cannot be guaranteed on arbitrary terrain.

The goal in designing Jim was to rely on the simplicity of a frame-walker approach to provide a basic platform to design and test a generally applicable vision-based navigation system and control architecture which can be propagated forward to more complex mechanical designs.

KINEMATICS

As mentioned previously, Jim's kinematics is that of a robot with two independent frames and central body. A detailed mechanical rendering can be seen in Figure 1. It has two walking gaits: a basic motion gait that provides non-continuous motion with very reliable dead reckoning accuracy, and a continuous motion gait which increases speed and decreases abrupt stopping, but limits turning and incurs some dead reckoning inaccuracies due to slipping.

Though Jim is a frame-walker, its kinematic model is not as straightforward as one might expect. One of the major complications is that none of Jim's motion is binary, that is, characterized by absolute positions such as up/down or forward/back. Even the switching of frames requires tracking of a continuous range of motor positions. Therefore, software must be involved to some degree in manual as well as autonomous modes of operation. This involvement is discussed later.

The complete independence of Jim's two frames is what allows it to achieve continuous motion, but this attribute also provides a good amount of mechanical redundancy. If one of the frames fails to translate due to mechanical failure, or becomes entrapped in an obstacle and unable to move, enough actuation remains to operate completely, though non-continuously. This can be understood more thoroughly after a description of Jim's motion sequences has been given.

BASIC MOTION

In its simplest form, forward motion is achieved by frame switching and frame translation, as depicted in Figure 2. Assuming a starting position with both frames

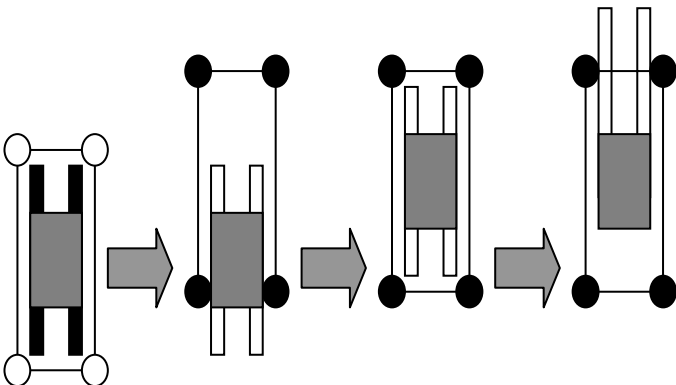


Figure 2: Basic forward motion

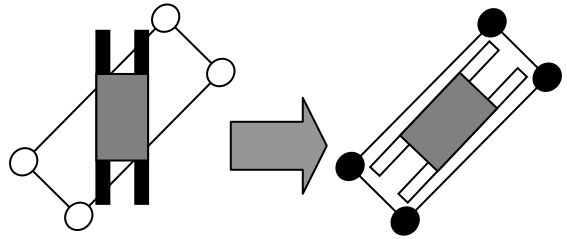


Figure 3: Turning

centered with respect to the body, one frame is chosen to support the rest of the robot, in the case of Figure 2, this is the bottom frame (support structures on the ground are filled black in the figure). With this frame on the ground, the other frame (the free frame) moves forward. The free frame is then placed on the ground and is translated backward, moving the entire robot forward. The frames' roles are then reversed and the process is repeated. This makes up the motion of one walking gait. For reverse movement, the process is the same with the translational frame movements reversed.

Jim is able to turn only its bottom frame, and can therefore only change the orientation of the entire robot when the bottom frame is on the ground. When this is the case, as seen in Figure 3, simply rotating the bottom frame with respect to the body turns the robot. The top frame can be extended in its new position, and forward motion can continue. When the bottom frame is again freed, it can rotate back to a forward position. As can easily be seen, care must be taken not to allow the top frame's legs to collide with the bottom frame as the turning procedure is executed or as the bottom frame is returning to neutral. This is handled by software collision checking, described later.

To switch frames, as well as to tilt itself for hill climbing, Jim can independently or concurrently raise the forward and rear ends of its top frame. This is depicted in Figure 4.

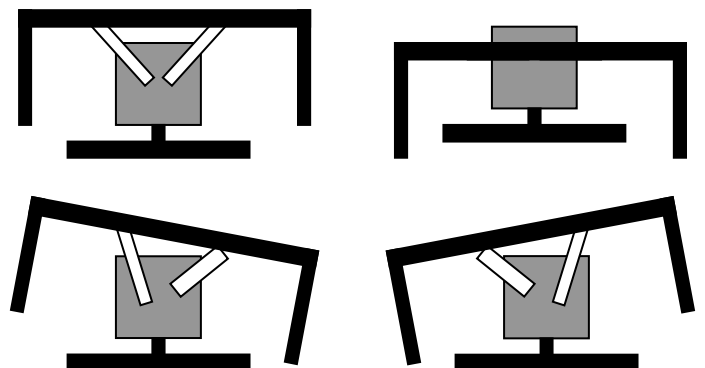


Figure 4: Top frame example configurations

CONTINUOUS MOTION

The same basic movements that accomplish basic motion also make up Jim's continuous motion gaits; however, velocity matching and acceleration issues now have to be factored into the control scheme.

A continuous motion movement, referred to as a "run," consists of three phases. These phases are:

- *Acceleration*: ramping the velocities of the frames up to the desired running velocity, rather than jolting the robot to a high speed
- *Running*: keeping a steady center-of-mass velocity and continuous motion
- *Deceleration*: ramping the frame velocities down to zero instead of jolting the robot to a stop

By imposing these three phases to a run sequence, slippage, stress and damage caused by sudden velocity changes can be avoided.

A run sequence can be of arbitrary velocity (up to the limits of the motors) and of arbitrary numbers of whole or partial strides. When a run sequence is initiated, a higher-level algorithm computes an optimal trapezoidal velocity curve to fit the distance and velocity desired. This curve is dynamically sampled for each step as the run sequence is performed.

While running, the frames must not stop their translational motion in order to switch frames. To achieve this, a variation on the simple motion gait is used. This gait is depicted in Figure 5. As the grounded frame translates backward, moving the entire robot forward, the free frame translates into its forward-most position at twice the current velocity of the grounded frame. This ensures that the free frame is in position to be switched when the grounded frame reaches the end of its stride. The free frame is then translated backward to match the velocity of the grounded frame and almost immediately after this, the two frames are swapped. The newly freed frame then translates forward to get into position for the next frame switch, and the process is repeated.

During a run, the bottom frame is allowed to turn, but at angles less than or equal to seven degrees from center. This is to avoid a collision between the top and bottom frames.

MECHANICAL DESIGN

Jim's design stems from the challenge of building a robot that incorporates as many parts currently in the Robotics Club's inventory as possible. This approach was taken in order to reduce its mechanical cost so that financial resources could be focused on reusable control electronics.

The basic mechanical structure of Jim consists of a chassis and two independently translating frames. The lower frame has the additional ability of rotating with respect to the rest of the robot, and the upper frame is able to tilt in a positive or negative pitch along the length (Z-axis) of its body. The upper frame can also move vertically, without tilt.

BODY

Jim's body accounts for approximately sixty percent of its overall weight, due mostly to the motors and batteries it holds. It was designed to serve these purposes:

- Hold and allow rotation of three shafts (bottom rotation shaft, and upper arm rotation shafts)
- Hold three motors to drive those shafts
- Securely hold four 2.5" x 6" x 3.75" batteries
- Securely hold and provide access to all electronic logic circuits, connectors, and switches
- Securely hold a 11" x 2.75" x 2" inch notebook computer with padding
- Securely hold five 1" x 5" x 3" inch amplifiers

While meeting these requirements, the body frame also needed to remain relatively light. In order to accomplish this, 1" x 1/8" angle aluminum was welded together to form the body frame, and 3" x 1/8" channel aluminum was used to hold the motors and shaft bearings. The final frame configuration is shown in Figure 6.

The amplifiers, logic circuits, connectors and switches are mounted on Plexiglas panels that are shaped to fit in the frame's spaces. These panels can easily be unbolted to gain access to electronics, and the panel holding the HC11 embedded computer is mounted on removable hinges for quick, repeated access.

For the notebook computer, care was taken to design an encasement to shield it from any moving parts as well as protect it from shocks, while providing

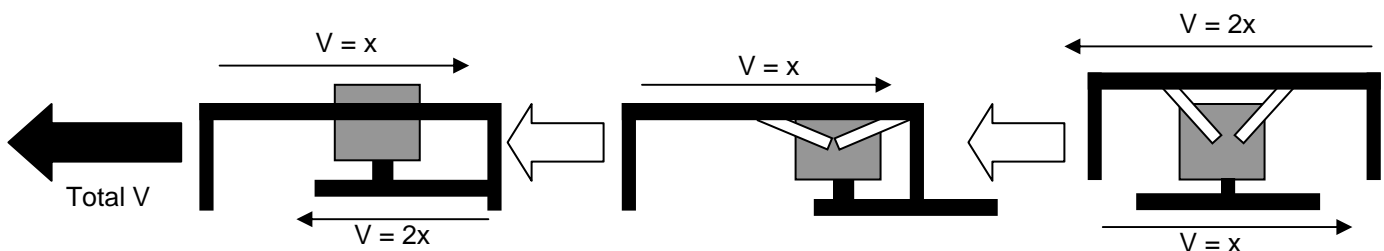


Figure 5: Continuous motion gait

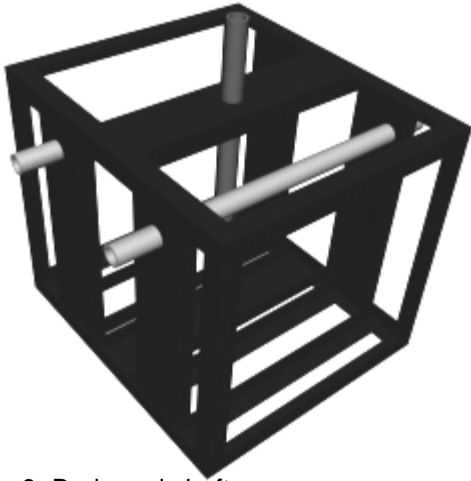


Figure 6: Body and shafts

accessibility for programming and reconfiguration. This was done using a protective aluminum sheath, into which the laptop can be placed from above. The laptop itself is protected with a neoprene casing, secured with Velcro.

LOWER FRAME

The vertical shaft shown in Figure 6 serves as the connection between the lower frame and the body and rotates in conjunction with the lower frame. It has a 1" diameter and a 1/8" wall thickness and is threaded on the lower end. Situations could arise in which the lower frame is forced to bear horizontal loads, so a lazy susan (a wide, flat bearing) was placed between it and the body. Since this bearing must be in compression to bear a load, and since the lower frame will experience vertical forces in both directions, a thrust bearing opposing the lazy susan was used. A large nut was then tightened on the bottom to put compressive loads on both the lazy susan and the thrust bearing at all times. A motor drives this shaft from a chain linkage inside the body.

The bottom of the vertical shaft is slotted and a custom clamp assembly is used to provide a slip-free connection to the bottom frame bushing. This bushing,

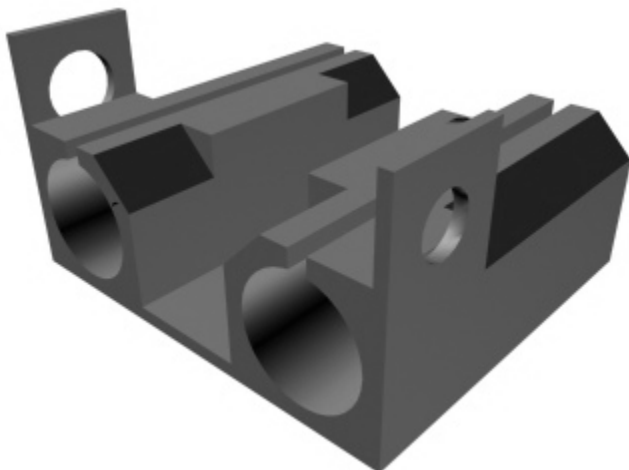


Figure 7: Bottom frame bushing

depicted in Figure 7, is composed of anodized (hard coat) aluminum, and is lined with Teflon on the inside to provide slippage. The bottom frame tubes pass through this bushing, with a toothed rail, running along the length of the tubes fitting into slots in the bushing. The skis (the part of the lower frame that actually touches the ground) are attached to these tubes at each end using an aluminum plate and cross beam. To drive the lower frame assembly, a motor is mounted on the bushing and drives a shaft with two gears that mesh with the toothed rails.

UPPER FRAME

To create an upper frame that is able to raise and lower without interference from the body, its framework needed to be set wide apart, unlike the lower frame. Materials and motor positions had to be chosen carefully to avoid unbalanced positions and excessive weight on the lifting motors.

The upper frame is attached to the body via four arms made of 1.5", 1/8" wall aluminum box stock. Two of these arms (one for the forward set and one for the back) are driven via chain linked sprockets at a 3:1 ratio to motors mounted in the body. Each set of arms can be raised or lowered independently as to tilt the upper frame to a positive or negative pitch, as well as to raise and lower the upper frame entirely.

The arms are mounted via rotational joints to the upper frame bushings. These joints rotate freely by placing Delrin between the arms and the bushings for slippage. The bushings themselves also use Delrin to provide slippage for the upper frame tubes. Each bushing has two, 1.5" long Delrin tubes, mounted inside 3" x 3", 1/8" wall aluminum box stock, spaced 4" apart to prevent binding. The upper frame tubes are made of aluminum, and are anodized to provide smoothness and durability.

On the forward set of upper frame bushings, a shaft runs across the width of the top frame, supported by bearings. This is the main upper frame translational

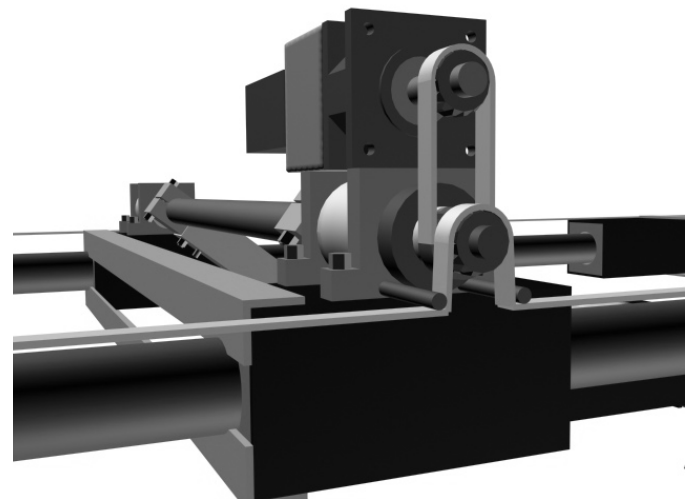


Figure 8: Shaft, bushing, drive axle and chain path

drive shaft, which is driven by a motor attached to one of the forward bushings. By mounting the motor at the attachment point of the arm to the top frame, its weight remains directly supported by the body, and does not vary the weight distribution as the top frame translates.

Two sprockets are mounted on the drive shaft and are aligned with the top translational tubes. These sprockets drive two cable chains that run the length of the tubes and are fixed at the ends. Figure 8 shows the chain's path through four idler sprockets on one of the bushings. This provides the force to move the top frame evenly through the bushings.

ELECTRONICS

Jim's electronic system was designed with reliability, safety, and scalability as primary objectives. A basic block diagram of the system is pictured in Figure 9.

POWER

The power system is driven by four, twelve-volt lead batteries, wired to produce 2.5 amps at 24 volts. These batteries can be directly cut off via the emergency stop (E-stop) switch. These batteries provide power to all motors, logic, and the embedded computer, but not to the laptop, which has its own internal battery (see

below). With this system, all motion can be cut while the laptop remains functional to avoid disk corruption.

LAPTOP

Jim's higher level algorithms, vision, and control interfaces are all functions of the onboard laptop. With the demands of all of these functions, the laptop is required to have a good amount of processing power. Therefore, the system used is an I586, 200 MHz processor with 48 megabytes of RAM (in order to avoid the need for disk cacheing). It is self-powered, using a 35WHr 8-cell NiMH battery, which averages approximately one hour of life under normal Jim operating environments. Its operating system and development environment is described below.

The laptop communicates with the HC11 via a serial link and to a Quickcam² through a parallel link in bi-directional mode. The Quickcam draws five volts of power through the PS/2 mouse/keyboard port. To communicate via user interfaces or command-line interfaces to the operator, the laptop's keyboard and LCD monitor can be used, as well as a telnet/X connection through Ethernet via a PCMCIA Ethernet card.

EMBEDDED COMPUTER

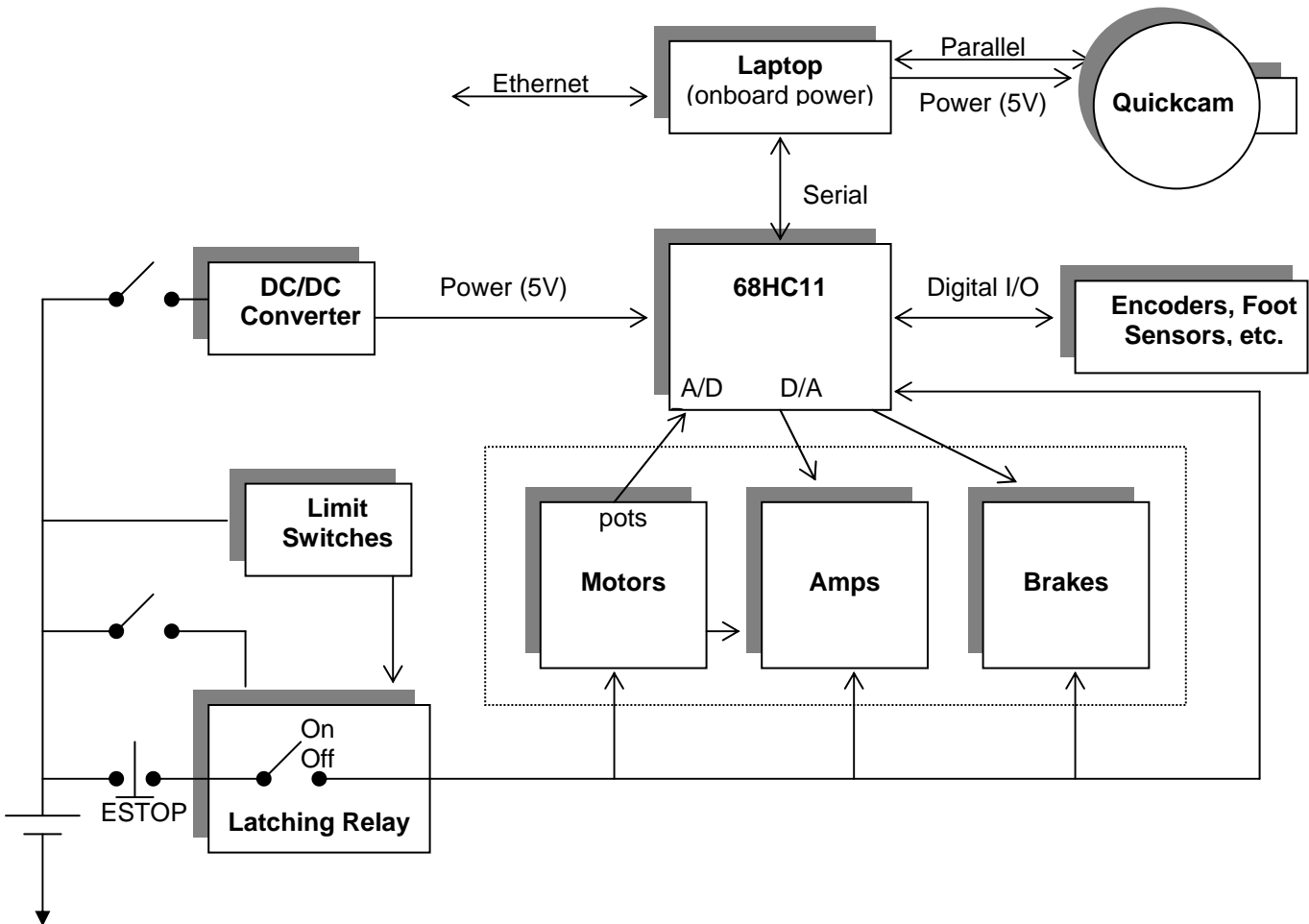


Figure 9: Electrical block diagram

The MC68HC11F1 microprocessor is mounted on a Cohesive Aesthetics GCB11 motherboard and runs at 16MHz with 32Kb RAM and 8Kb ROM. It is used to handle all low-level motion control and sensor data gathering (other than vision data) and receives instructions, sends back data, and downloads its program code exclusively through its serial link. It is provided clean power through a DC/DC converter and can be shut off with a switch mounted on the body or by cutting power via the E-stop.

Customizations to the GCB11 motherboard have provided Jim with many I/O ports, allowing room for

Type	Avail.	Used	Function
A/D	8	4	Potentiometers
D/A	8	5	Amplifier Control
D/D	16	12	Opt. Encoders/Switches
Power Out	4	3	Brakes
Logic In	4	0	Accessories

Table 1: GCB11 I/O

expansion and redundancy. The specific ports and their uses are enumerated in Table 1.

MOTORS AND AMPLIFIERS

Jim's motors are brushless DC motors, which run through gearboxes that produce a high (approximately 1:50) gear reduction. They are driven by Hall Effect amplifiers, which can perform current or velocity control loops. They are braked by default, and are released by supplying power to the brakes.

The positions of all but the upper frame translation motor are tracked by potentiometers. An optical encoder attached to the gearbox tracks the upper frame position. Since keeping the motors non-braked requires a half an amp of power, all motors that are free for a larger percentage of running time have had their brakes removed. The final motor configuration is shown in Table 2.

Function	Brake	Position
Upper Translate	N	Optical Encoder
Bottom Translate	N	Potentiometer
Arm 1	Y	Potentiometer
Arm 2	Y	Potentiometer
Rotation	Y	Potentiometer

TABLE 2: Motor configurations

These motors were obtained from an aborted NASA project. From this project's data on motor performance, Jim's total motor power draw is approximately 1.5 amps at 24V.

Limit switches are used to electrically prevent joints from overstepping their physical limits; most importantly, the arms. These switches are run through relays, which

trigger an immediate power system shutdown upon activation.

THE JIM STICK

The Jim stick is the main manual control device for Jim. It is based on the modular control architecture discussed throughout this paper, in that it uses the same serial interface as the laptop to talk with the HC11. The stick's tubular design is more comfortable and natural to hold than a normal button box, and the button positions are positioned in a trigger formation.

The Jim Stick translates combinations of movements of its joystick and pushes of its buttons into motion commands for Jim's embedded computer using an MC68HC811E2 microprocessor mounted on a Mini Board³ V2.0 inside the base of the stick. Due to this, its behavior is customizable, and can be configured to control Jim in a variety of capacities.

The microprocessor on the Jim Stick has software that can easily be reconfigured to control other robotic devices over a serial link. It can also be configured to serve as a bridge between analog control devices, such as the flightstick, mentioned previously, and a digital serial link. This "intelligent controller" is another example of how the individual components of Jim are generally applicable and are able to be propagated to future generations of robots.

SOFTWARE

In designing the software for Jim, the main goal was to stay as general as possible so that much of the code could be used in subsequent years. As a result, the software used in Jim is clearly modular, using layers of abstraction in order to keep Jim-specific code easily identifiable and making the basic infrastructure very portable to other robotic systems.

DEVELOPMENT ENVIRONMENT

In developing the layers which make up Jim's abstracted software model, care had to be taken to chose an environment that would provide the ability to work on, integrate, and test each layer conveniently and rapidly. Our most limiting restriction was our 68HC11-microprocessor compiler, which is able to compile C, but only runs in DOS. Unfortunately, DOS is not the ideal environment for writing the complex, graphical control and simulation software that Jim required. Therefore, the requirements for an ideal environment were:

- Editing and compilation for a powerful, standard programming language
- Scalability, so that software can be developed on the desktop and embedded in the robot with little to no modification
- A graphical environment, as well as development libraries for 2D and 3D graphics and user interfaces

- Access to serial, parallel, and Ethernet communication
- Available Quickcam drivers in order to develop vision system
- The ability to easily run the DOS-based Intron⁴ Motorola 68HC11 C compiler

With these requirements in mind, and due to personal bias, the non-HC11 software was developed in C++ under the Linux operating system, using X11R6 as a graphical environment. For simple 2D graphics, a canvas library was written using XLib and could be linked with any module that required simple graphics to be displayed. For 3D graphics, Mesa 2.2⁵, a freeware, non-hardware dependant port of OpenGL was used. Mesa's texture mapping ability was used to write a user interface for the vision module so that it could easily be run under screens with different visual depths.

To develop for the HC11, code was compiled by Intron running under DOSEmu⁶, a DOS emulator for Linux, and was downloaded to the HC11 via a Linux terminal program or through the serial link integrated into Jim's control software, described later. For Jim to be operational, this code must be downloaded, even if control will be done using the Jim Stick. However, when the low-level software has been proven to be reliable, a ROM can be burned and booted upon power-up.

3D KINEMATIC SIMULATOR

Due to the fact that Jim's kinematic model, described previously, required well thought out control algorithms, the development of these algorithms needed to begin months earlier than the mechanics would be ready.

Therefore, a kinematic testing environment was written, and a "Virtual Jim" was constructed within it. Figure 10 depicts this simulator in use. The 3D representation is manipulated through software control algorithms, which, when perfected, can be ported with little to no modification to control the actual robot. The simulator software can also be linked with the actual control software to display in real-time the 3D configuration of the robot while it is being operated.

This testing environment is general enough to be used on future robots and easily extended to incorporate new geometric primitives for a number of structural representations.

HIGH LEVEL PLANNER AND CONTROL INTERFACE

The high level Jim control software is the main interface mechanism used for autonomous operation. This interface, depicted in Figure 11, provides the user with the ability to test planned routines (or events) by graphically viewing the results before Jim physically executes them.

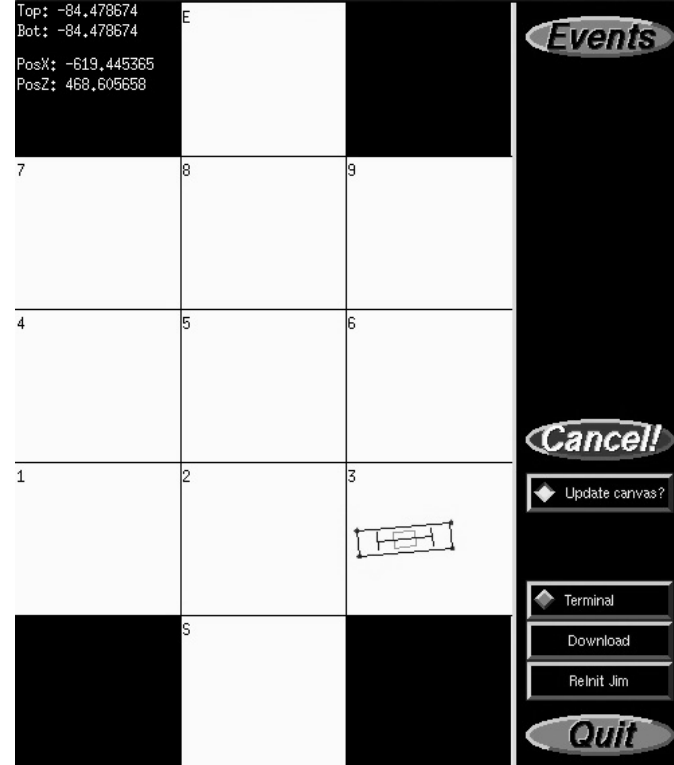


Figure 10: Jim control panel

The Jim control panel, along with providing routine testing abilities, allows a command-line interface to the embedded HC11 to be used for debugging low level software. Upon execution, it downloads the default behavior code to the HC11 and executes it, but can also download separate code for testing or modified behavior.

The control panel can be compiled in two modes of operation: with or without the serial link. This allows the user to test routines offline, with the robot not even present. This software is very general and can be ported very easily to future robots.

HIGH TO LOW LEVEL CONTROL LINK

Jim's control hierarchy is very straightforward. The link between high level control methods and Jim's onboard computer is through a 9600bps serial connection. This connection has a defined, common protocol and, as mentioned above, can be accessed using any programmable device capable of serial communication.

Due to the fact that continuous motion could not reliably be achieved under simple electrical control, and that even in non-continuous motion, Jim is a complex mechanism to operate, the embedded computer plays a role in both autonomous as well as manual modes of operation. To provide an unlimited number of options for commanding the robot, the serial control interface is strictly defined and is unchanged between manual and autonomous control modes. Therefore, the laptop and any manual control device can be swapped without any software reconfiguration.

VISION

Jim's vision system serves as its primary sensor and is designed to perform the following tasks:

- Ball finding and position determination
- Cone finding and position determination
- Dead reckoning correction and navigation

As described above, all vision processes occur on the laptop, as do decisions based upon the vision analysis. However, using the feedback provided by the HC11, the vision system can be tightly correlated with the robot as a whole.

Essentially, the dead reckoning correction performed by the vision system is designed to determine the probability P of being in a position (x, z, ϕ) from known information: a dead reckoning model and a vision model. From the dead reckoning model we can describe a function $D(x, z, \phi, x_0, z_0, \phi_0)$ that is the probability of the robot being at a position (x, z, ϕ) given the dead reckoning predicted position (x_0, z_0, ϕ_0) . From the vision model, which includes the model of the environment, the model of the camera, and the position of the camera relative to the robot, we can describe a function $V(x, z, \phi, i)$ that is the probability of the robot being in position (x, z, ϕ) given a camera image i .

The actual probability function P is clearly dependent on many factors other than D and V , but, given our knowledge constraints, we can assume it is essentially a positively increasing function of these two functions. Additionally, we know that D is rather localized and that values of V can be large in various locations throughout the (x, z, ϕ) space (it is possible to observe similar things at different locations). Thus, we choose the heuristic of first selecting a subset of (x, z, ϕ) space in which D is large, then, within that subset, finding the global maximum value of V . This works well in practice because D is a much more computationally

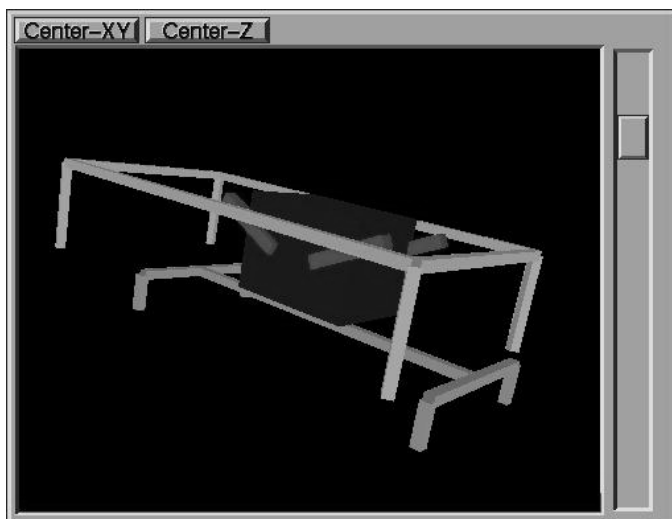


Figure 10: 3D kinematic simulator



Figure 12: Vision calibration UI

efficient function than V .

For ball and cone finding, the vision algorithms are quite simple, employing a color histogram algorithm enhanced with shape determination heuristics. To compute actual distances from image coordinates, the vision model's camera position is used, as well as information about the position of the xz plane (floor) given to the system during calibration. A field-of-view calculation is done using this information and, as long as the camera position stays fixed relative to the robot body, all calculations will be accurate indefinitely.

The calibration routine for the camera is necessary for both color calibration as well as position information. To simplify this and make it as user-friendly as possible, a calibration control UI is used which outputs the calibration information to a file. This file is then read in when the Jim control software is executed and is valid until the lighting environment's color spectrum changes or until the camera is physically disconnected from the robot. This UI is shown in Figure 12.

ACKNOWLEDGMENTS

Without the support and assistance of these people, the Jim project would not have been possible:

- William "Red" Whittaker
- John Wiss
- Ryan Miller
- James "Oz" Osborne
- CMU Center for Medical Robotics and Computer Assisted Surgery
- Matt Mason
- Ralph Hollis
- Ben Brown
- Eric Rollins
- Colin Piegras

- Patrick Nelson
- Jetware Heavy Duty Sipper/Stirrer Straws

CONTACTS

The Carnegie Mellon University Robotics Club has been actively building robots and sensing systems since 1984. It is an officially recognized CMU organization, but operates under the supervision, guidance and funding of the CMU Robotics Institute. The presidents, Gabe Brisson and Rich LaBarca, met at a freshman welcoming party at a podiatrist's house in New Jersey and decided to make robots together at CMU. Four years later, with the help of three talented freshmen and one sophomore, they gave birth to a hefty young lad named Jim.

REFERENCES

1. Annual SAE competition: <http://www.sae.org/STUDENTS/walking.htm>
2. Copyright Connectix Corporation <http://www.connectix.com/html/hardware.html>
3. Copyright Fred Martin, MIT Media Lab
4. Copyright Introl Corporation
5. <http://www.informatik.unisiegen.de/softdocs/opengl/Mesa/Mesa.html>
6. <http://www.suse.com/~dosemu/appendix>

APPENDIX

BUDGET

Description	Supplier	Price	Notes
<i>Mechanical</i>			
Body Aluminum	CMU Meche Shop	\$50	Donated
2 Arm Bearings	Applied Industrial Technologies	\$150	
Arm Shafts	CMU Meche Shop	\$10	
Rotation Shaft	CMU Meche Shop	\$7	
Sprockets	Applied Industrial Technologies	\$50	
All Nuts & Bolts	RoboClub Stock	\$8	Owned
Lower Bushing	NASA	\$300	Donated
Lower Tubes	NASA	\$80	Donated
L-Frame Aluminum	NASA	\$7	Donated
L-Frame Wood	Home Depot	\$10	
U-Frame Tubes	Clinton Aluminum	\$75	
Bushing Metal	Clinton Aluminum	\$20	
U-Frame Legs			
U-Frame Delrin	Small Parts Inc.	\$21	

U-Frame Chain	Applied Industrial Tech.	\$26	Owned
U-Frame Bearings	Small Parts Inc.	\$20	
<i>Electrical</i>			
5 Motors	NASA	\$7500	Stolen
Switches & Pots	CMU Physics Stockroom	\$30	
A/D converters	Digikey	\$24	
7 12V Batteries	Ryan Miller	\$150	Donated
DC/DC Converter	Vicor	\$50	Owned
Misc. Electronics	Various	\$40	
<i>Computing</i>			
Laptop Computer	Gateway 2000	\$2000	
Minboard	Fred Martin	\$50	Owned
GCB11	Coactive Aesthetics	\$100	Bought Used

Total \$10,778

PROJECT MEMBERS

Name	Yr	Specialty	Major	Hours
Gabe Brisson	S	P,M,E,H,L,S	ECE	469
Jonathan Hurst	F	M	ME	300
Rich LaBarca	S	P,H,L,S,V,D	CS	469
Peter Sand	O	E,H,L,V,D	CS	300
Charlie Reverte	F	M	ECE	260
Michelle Ungerer	F	M,H	ME	222

Total: 2010

Year:

S: Senior
O: Sophomore
F: Freshman

Specialty:

P: Project Leaders
M: Mechanics
E: Electronics
H: High Level Software
L: Low Level Software
S: Simulation
V: Vision
D: Documentation

Major:

CS: Computer Science
ECE: Electrical and Computer Engineering
ME: Mechanical Engineering