

Machine Intelligence Laboratory
EEL 5934

Josh Fryman

“Antaeon”
Autonomous Robot Report
December 9, 1996

Instructor:
Keith L. Doty
Department of Electrical Engineering

Table of Contents

Abstract	-----	3
Executive Summary	-----	4
Background	-----	5
System Design	-----	7
Leg Design	-----	7
Body Segment Design	-----	12
Device Installation	-----	14
Servomotors	-----	16
Sensory Inputs	-----	17
Materials	-----	18
Algorithm Development	-----	20
Future Directions & Project Evaluation	-----	30
References	-----	31
Acknowledgments	-----	32
Appendix: Source Code	-----	34

Abstract

An organized approach to designing a complex autonomous robot platform is presented. This robot platform is built along the biological basis of the ant. The aim for this platform is to provide a minimum vessel for exploring self-learning with respect to gait, terrain navigation, and group behavior. This platform will be “fully” developed when it has sufficient hardware and software to handle movement in the following conditions: cautious exploration, walking, and obstacle avoidance. These behaviors have been “brute force” programmed to prove that the solution is attainable, and to gain insight into what control algorithms will be needed for self-learning. Like the ant, this platform will consist of three independent segments with two legs each. Each leg has three degrees of freedom in movement and each segment is connected by a two degree of freedom in movement joint. Each segment has its own microprocessor and is designed such that an arbitrary number of segments can be connected together in order to obtain complex robot structures with minimum re-design. The final leg design used is a stacked servo configuration. The final body segment design is a simple cross-shaped frame. The force equation analysis of the final selection in leg design and body segment design are reviewed. This analysis is followed by a logical development of software creation for gait behavior, obstacle avoidance, and ground detection. Four sensor types, using three physically distinct sensors, are implemented to provide real-world feedback. This feedback is used to control the arbitration between programmed behaviors.

Executive Summary

After presenting a stated design goal, the project is scaled back to an achievable level for one semester of work by one student. The workload is reduced from producing an entire robot to producing one segment, well designed and ready for replication. The segment is to have the basic behaviors integrated (walking, object avoidance, ground detection, turning). Future work will be to expand the design into the multiple-segment robot.

A thorough discussion of all attempts to construct this autonomous robot platform is presented. The varied attempts at leg design, their reasons for failure or success, and the material learned from each is discussed. The body segment design is similarly examined and each attempt is discussed within the context of benefits/drawbacks for each method.

The final selection of the leg and body design are discussed with force equation analysis using simple assumptions. The assumptions are designed to cover nominal anticipated operation of the autonomous agent, while giving a great enough flexibility that the agent can attempt unexpected solutions to situations without causing self-destruction.

The pseudo-code for controlling algorithms for gait, turning, and terrain surface calculation are presented with discussion. A brief discussion of sensor types and placement is presented to generate the feedback needed to arbitrate the different behaviors integrated in the segment.

Section 1: *Background*

The primary goal of this robot is to mimic an ant. To mimic an ant is a design goal which can not be obtained readily, so this has been the underlying basis to create a platform from which this goal may be reached. The aim for this platform is to provide a minimum vessel for exploring self-learning with respect to gait, terrain navigation, and group behavior. Using an ant for a basis introduces many interesting design complexities over other walking robots.

Ants have three segmented bodies, where each segment has three degrees of freedom in movement. Each segment appears to have two legs attached to it, each leg having three degrees of freedom in movement. This is being carried into the mechanical design of this autonomous robot platform, where there are three segments, each segment joined by two servomotors for only two degrees of freedom. Six legs are provided, two per segment, with three degrees of freedom in movement per leg.

This platform will be “fully” developed when it has sufficient hardware and software to handle movement in the following conditions:

- cautious exploration: the terrain is uneven or unstable in the immediate path of the robot; an example is horizontal-ladder or inclined-surface traversal;
- walking: stable walking mode, where only one leg is off the ground at a time; and advanced walking mode, where three legs in a triangular pattern are off the ground at a time;
- obstacle avoidance: stepping over, under or around obstructions in the immediate path of the robot.

These design goals are too complex to be attained in a one-semester project with just one student working on the project. These goals were scaled back to the minimum criteria of designing and implemented one segment of this robot ant. The segment chosen was the center segment.

In order to reproduce the gait of an ant in software, we notice first that an ant usually has as many legs on the ground as possible. When walking forward, only one leg tends to advance by lifting off the ground at a time. This is a very stable weight distribution for the ant, and is copied to the robot model.

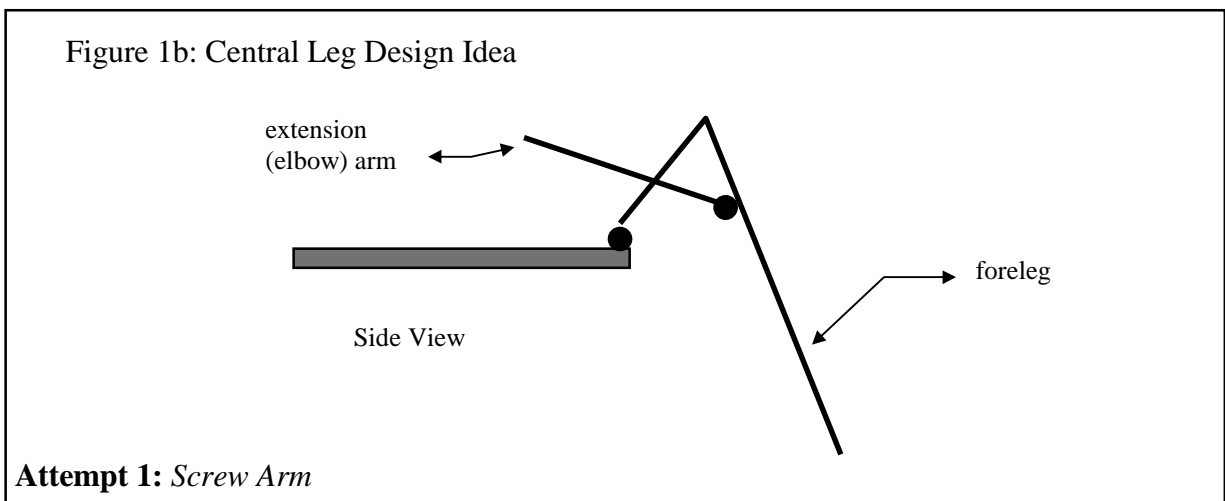
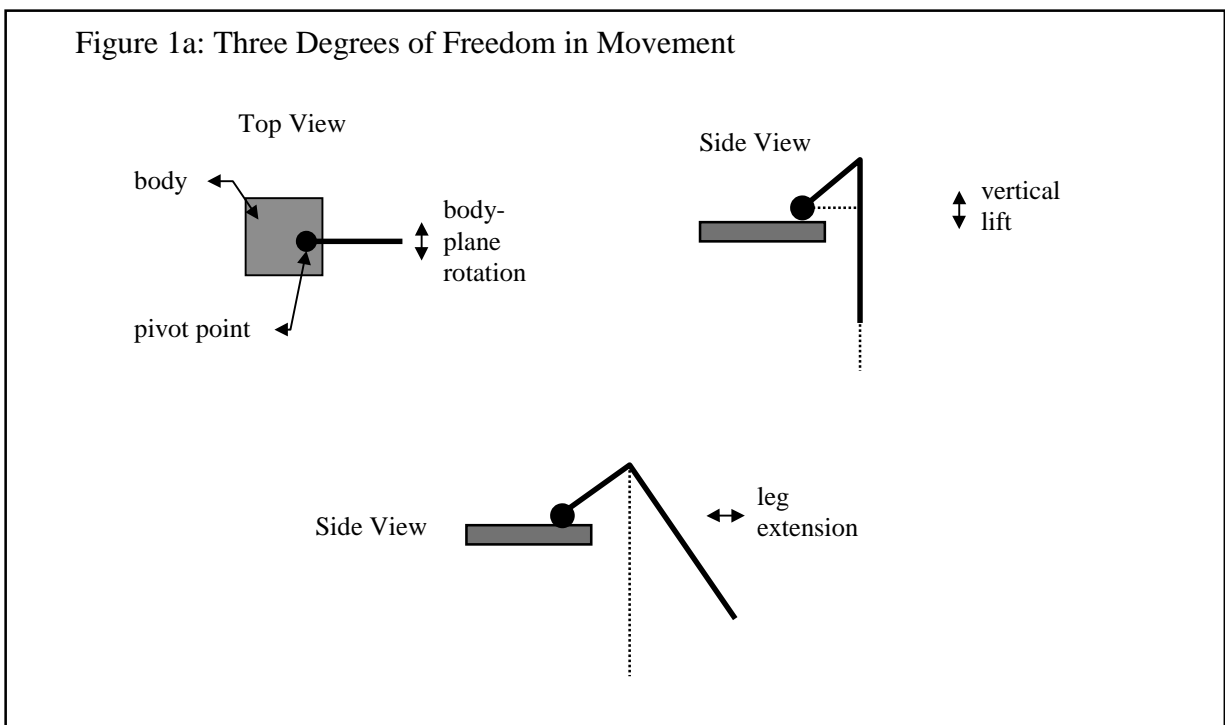
To respond to obstructions in the path of the robot and a non-fixed environment, sensory input devices must be placed strategically about the robot segment. These will be monitored at all times to prevent collisions with obstacles, running off the edge of the world (e.g., a cliff), responding to the inclination of the segment for balance, and to verify each leg is in contact with the ground when appropriate.

The primary goals of solving the software issues can be summarized as follows:

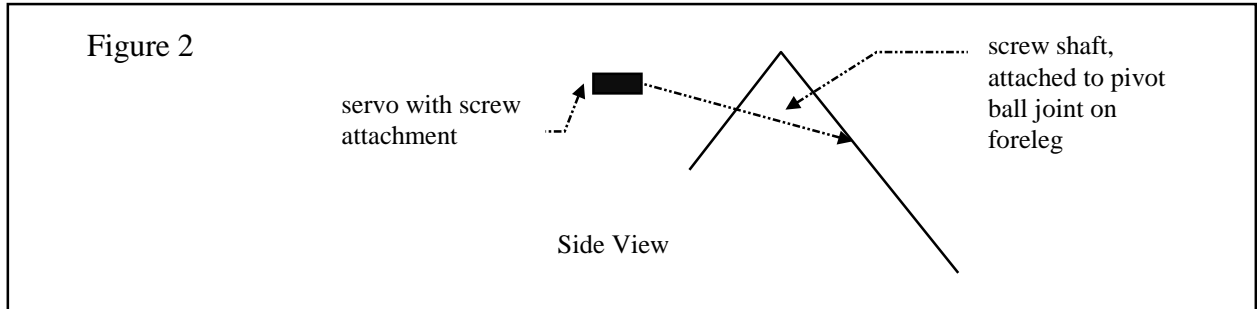
- Simple code (limited space available)
- Quickly executing (time critical motor driver and IR emitter/sensor system)
- Multitasking (capable of responding to outside world)
- Rich motor driver (capable of driving many servos simultaneously)

Leg Design

The primary issue in designing the leg unit is to provide three degrees of freedom in movement, as can be seen in Figure 1a. Obtaining the extension of the leg, along with the lift and body-plane rotation, has led to many interesting design ideas, most of which were failures. Most of the attempts documented here focused on a simple design for extending the foreleg, as can be seen in Figure 1b.



The initial plan was to mount a freely pivoting screw to the foreleg, while attaching a servo capable of spinning arbitrarily in either direction to the opposing end as shown in Figure 2. As the servo turned clockwise, the leg is forced out, and vice versa.

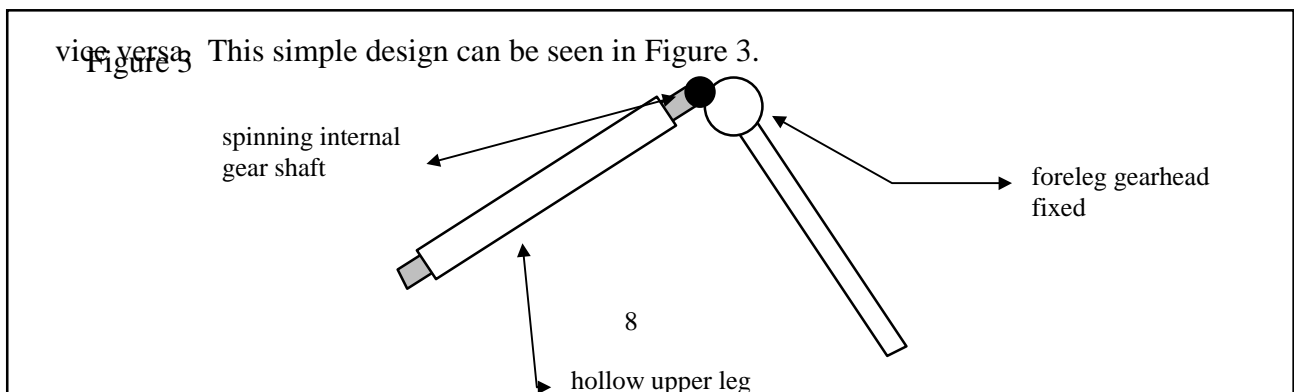


This design was severely hampered by the design of modern servos, which turn 60° arcs in the range of 0.20 - 0.50 seconds. Assuming a thread spacing of 16 threads per inch, to obtain a one-inch extension or retraction would result in 6 revolutions per thread, or 96 revolutions, each revolution taking 0.25 seconds average, or 24 seconds. This is too slow for a robot which has a limited life due to self-containment with batteries. No prototype was built, and another design idea was considered.

Attempt 2: Geared Arm

Removing the cross piece shown in Figure 1b, the initial upper leg diameter was increased to insert a rotating shaft. The ends of this shaft were attached to gear units, where one unit was attached to a servomotor and the second unit was attached to a fixed gear at the joint in leg segments. As the gear spun clockwise, the leg would extend, and

vice versa. This simple design can be seen in Figure 3.



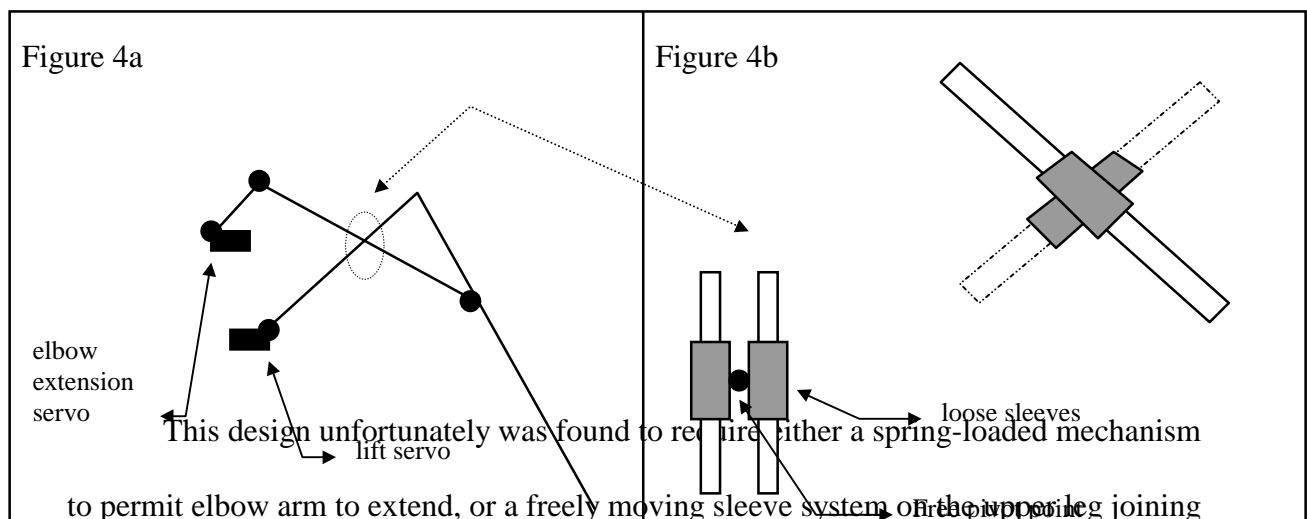


Side View

This had many mechanical problems, starting with the increased weight required by the larger-diameter initial leg segment coupled with the inside shaft. Additional weight was required by the gear boxes. Significant force loads would be placed on the gear at the leg segment joint, requiring both strong gears and a very strong joint connection, thus increasing the weight again. The weight overhead of this design was approximated to be too great a drawback to implement.

Attempt 3: Elbow Arm

The first successful prototype followed the initial design idea as shown in Figure 1b. This design used an elbow mechanism to directly push the arm forward with a 140° swing of the servo, as shown in Figure 4a. This provided an excellent source of power and fast extension of the foreleg, with the tradeoff being additional torque on the servo.

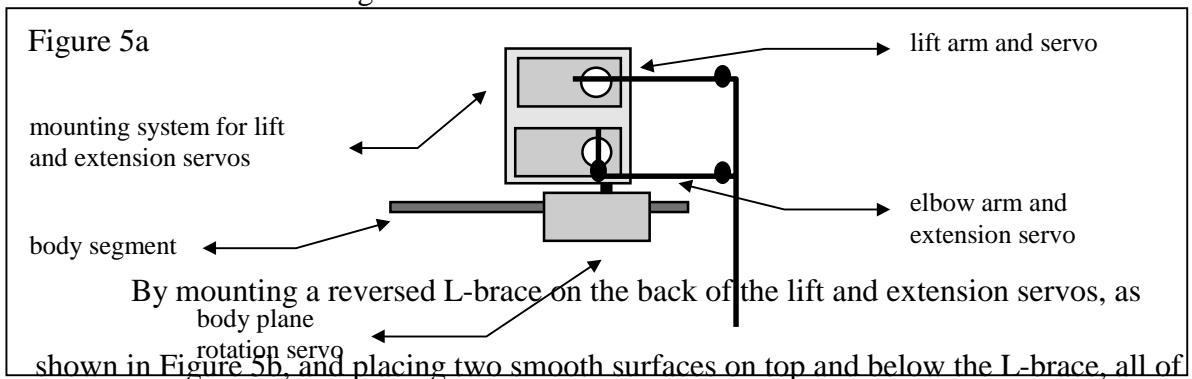


the elbow arm, as shown in more detail in Figure 4b. Additionally, the elbow arm tended

to hit the body platform, mandating mounting the legs on extended projections away from the body. The elbow arm also required a wider body design, allow the arm to travel forward and backward while extending the foreleg without impinging with the opposite side leg's elbow arm. The final drawback to this mechanism was the complexity of the force equations. Using this design as a basis, Dr. Doty was consulted and a minor redesign implemented to remove most of the drawbacks and keep the benefits.

Attempt 4: Stacked Elbow

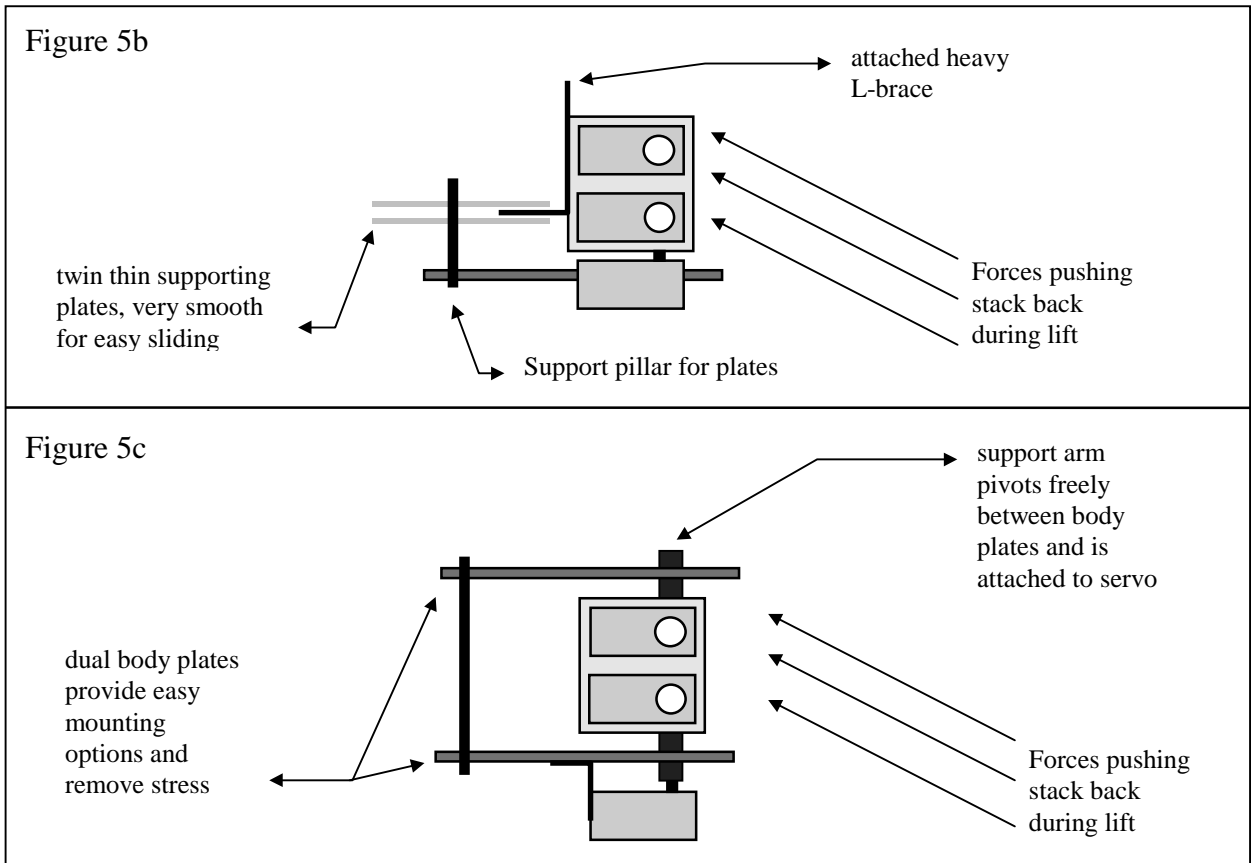
The stacked elbow design mounts the lifting servo and the extension servo on top of each other, these two being mounted directly on top of the body-plane rotation servo, as shown in Figure 5a. This simplified the torque/force equations and made the space usage more efficient. The only significant drawback to this design is the large amount of force created on the body-plane rotation servo head. The force has been judged sufficient to break off the servo head under full load conditions, and burn the servo out under nominal conditions. This has been compensated for with two separate minor modifications to this design.



the force on the servo head is absorbed by the twin sheets. This acts as a stabilizer for the servo stack as well as nullifying the torque. While this solved the problem, it added the

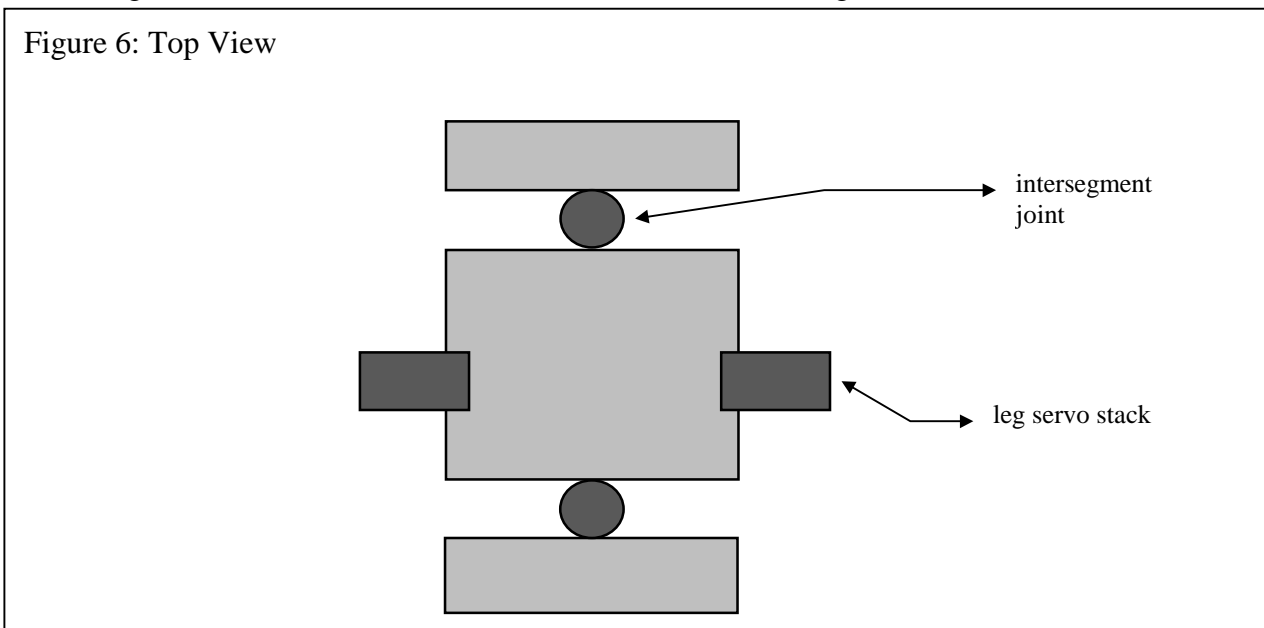
extra weight of two additional (large) sheets of material, as well as the appropriate mounting hardware. This weight overhead led to the second refinement.

By removing the L-brace and replacing it with a vertical pivot rod, as shown in Figure 5c, the torque forces are also nullified. The vertical rod requires that only two body frame pieces are required – rather than three with the L-brace method – to hold the forces off the body-plane rotation servo. This is the method that has been judged to be most efficient in space, weight, and strength. This method is the one that was chosen for the robot platform. (The shoulder servo was moved to the top in implementation.)



Body Segment Design

In designing the body segment, the main criteria was weight, which necessitated the smallest possible size. With two large, heavy leg servo stacks and an inter-segment servo stack, a sufficiently strong frame was also required to keep the segment from tearing itself apart. The basic geometry points of consideration can be seen in Figure 6. Aside from the weight consideration, whatever design is used must not impinge upon the inter-segment rotation motion. A rotation of $\pm 60^\circ$ was the target ideal.



Attempt 1: *Square / Rectangular*

The simplest design to make is that of a square or rectangle. It provides a large area for mounting sensors, battery packs, CPUs, and servo units. This design has two critical failures of the criteria for the body segment: it has large regions of unused space, wasting weight, and it significantly interferes with the inter-segment motion unless the inter-segment servos are extended away from the body. Extending the servos requires

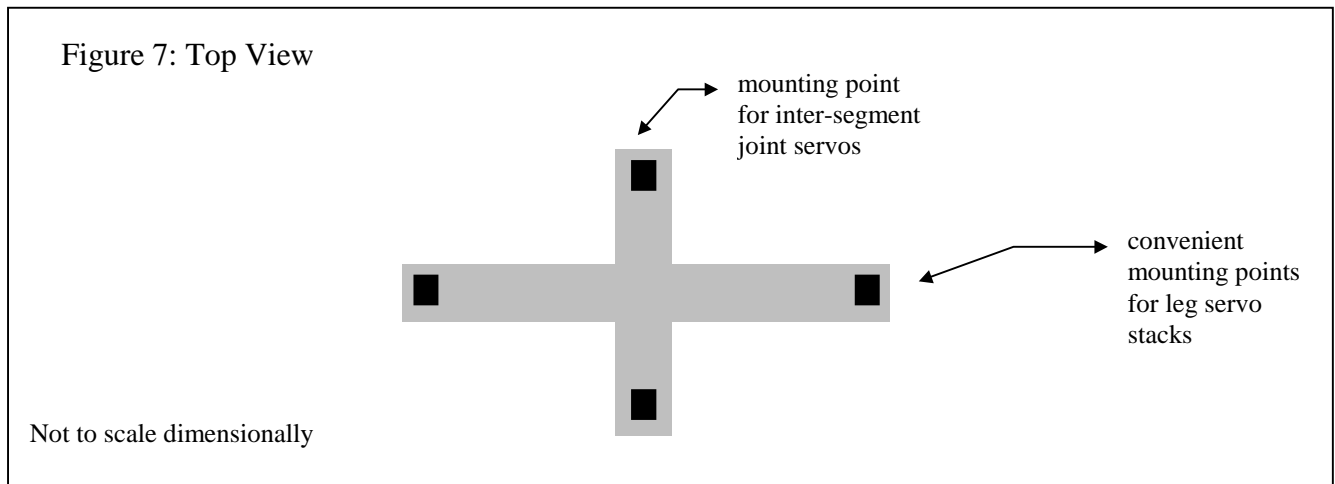
more material and in turn more weight. This idea was then refined to remove the motion problems and some of the excessive unused space.

Attempt 2: Octagon

The octagon is also a fairly simple unit to make. It retains the large surface area for mounting devices, and removes the impinging of inter-segment motion. While this design corrected the problems of the square body, there was still a large amount of space that was not being used effectively, increasing the weight of the segment unnecessarily. After careful consideration, it was decided that there was only one good way to remove the unused space problem.

Attempt 3: Frame Only

A cross-formation frame was designed which provides four mounting points: segment forward, segment backward, left leg, and right leg. Where devices are needed, such as battery units, CPUs, and sensors, they can be either directly mounted to the frame or a small frame extension can be mounted out of lightweight plastic (nylon) or wood (5-ply). This design can be seen in Figure 7.



Section 3: Device Installation

The selection of servomotors was dependent upon quick force approximations. Using the kinematics of the leg with simple assumptions about projected ranges of motions and simple heuristics, the following set of assumptions and calculations were derived:

$$W = \text{Maximum segment weight} = 6\text{lbs.} = 108 \text{ oz.}$$

$$L = \text{Upper and Lower Link Lengths} = 3.5 \text{ in.}$$

$$E = \text{Elbow Link} = 2.5 \text{ in.}$$

Assume: The upper link is usually horizontal

Shoulder:

Assuming only one leg is on the ground, the maximum forces occur when traversing an incline. Assuming a worst-case incline of 30 degrees,

$$\begin{aligned} W' &= W \sin(\text{slope}) \\ &= 54 \text{ oz} \end{aligned}$$

$$\begin{aligned} \text{torque to support weight} &= (L * W') / (\text{Legs on the ground}) \\ &= 54 \text{ oz} - \text{in} \end{aligned}$$

Lift Arm:

The worst case lift is when the leg is fully extended downward, projected outward at an inclination. Using the assumptions that the maximum lateral extension will be 45 degrees in the lift arm link, 30 degrees in the actual leg, the calculations are:

Let M be the moment arm,

$$\begin{aligned} M &= 3.5 \sin(45) + 8.5 \sin(30) \\ &= 6.7 \text{ in} \\ \text{lift torque} &= (M) * (\text{total weight}) / (\# \text{ legs on ground}) \\ &= 723 \text{ oz} \cdot \text{in} = 45 \text{ lb} \cdot \text{in} \end{aligned}$$

If the leg is limited to a mere 30 degree extension, these numbers reduce to,

$$\begin{aligned} M &= 3.5 \sin(30) + 8.5 \sin(30) \\ &= 6 \text{ in} \\ \text{lift torque} &= 648 \text{ oz} \cdot \text{in} = 40.5 \text{ lb} \cdot \text{in} \end{aligned}$$

However, when the leg is held vertical, these forces reduce to,

$$\begin{aligned} M &= 3.5 \sin(45) \\ &= 2.4 \text{ in} \end{aligned}$$

OR

$$\begin{aligned} M &= 3.5 \sin(30) \\ &= 1.75 \\ \text{lift torque} &= 260 \text{ oz} \cdot \text{in} = 16.25 \text{ lb} \cdot \text{in} \text{ (45 degree push down)} \\ &= 189 \text{ oz} \cdot \text{in} = 11.1 \text{ lb} \cdot \text{in} \text{ (30 degree push down)} \end{aligned}$$

Elbow Arm:

$$\begin{aligned} m &= 8.5 \sin(45) \\ &= 6 \text{ in} \\ \text{elbow torque} &= (m) * (\text{total weight}) / (\# \text{ legs on ground}) \\ &= 648 \text{ oz} \cdot \text{in} = 36 \text{ lb} \cdot \text{in} \end{aligned}$$

However, by using the elbow design with the additional arm length of 2.5 in, we reduce this by a factor of (2.5 / 3.5) or 0.71, for a resultant force of,

$$\begin{aligned} \text{elbow torque} &= (1.00 - 0.71) * (648 \text{ oz} \cdot \text{in}) \\ &= 187 \text{ oz} \cdot \text{in} \end{aligned}$$

If the extension of the elbow is limited to a mere 30 degrees, the forces reduce to,

$$\begin{aligned} m &= 8.5 \sin(30) \\ &= 4.25 \end{aligned}$$

$$\begin{aligned} \text{elbow torque} &= (1.00 - 0.71) * (m) * (\text{total weight}) / \\ &\quad (\# \text{ legs on ground}) \\ &= 133 \text{ oz} \cdot \text{in} \end{aligned}$$

In summary, the forces required by the three separate leg servos are: Shoulder = 54 oz - in, Lift Arm = 648 to 723 oz-in, and Elbow Arm = 133 oz - in. These worst case scenarios are further reduced by the real implementation in three ways: usually both legs are on the ground simultaneously, reducing all of these numbers by half; the computed maximum weight of 6 lbs was transferred into ounces with a 12.5% (1/8th) overestimation in weight to (6.75lbs) as a safety factor; and, the actual prototype weight measured at 5.1 lbs fully configured (batteries, CPU, sensors, frame, servos, wires), not 6, further reducing the numbers: Shoulder = 40.8 oz - in, Lift = 490 oz - in (extended), 143 oz - in (vertical), Elbow = 100 oz - in. Again, those reflect only one leg on the ground. When both legs are on the ground, those numbers are divided by two.

Servomotors

The servos obtained for this robot were Hitec/RCD HS-700BB units. These generate 133 oz-in torque at 4.8V DC, or 161 oz-in torque at 6.0V DC. These were chosen primarily for cost considerations, even though they do not provide sufficient torque for the lift joint. A torsion spring is required to generate the additional force needed at the lift arm. Hitec/RCD generously offered to provide these servos at \$20.24 each, rather than the usual \$37.50 through a special corporate sponsorship program. (For details, contact Mike Mayberry at Hitec/RCD -- <http://www.hitecrd.com>.)

These servos measure 2.3 x 1.1 x 2.0 inches, weighing a mere 3.68 oz (102 g). They are water tight, have a ball-bearing assembly, and use metal rings for gear protection against stripping. Unfortunately, these units draw high current and can cause the CPU to reset when several servos operate simultaneously.

The servos are driven in a Pulse Width Modulation system. The three-wire connector on each servo is color coded: red for positive power supply (4.5 - 6V DC), black for ground, and yellow for the PWM control signal. Using the MC68HC11E2FN microcontroller, with the RTI system enabled for 32.77ms interrupts, driving an output pin to +5VDC from 0VDC for a minimum count of 40 (where the count is in a for loop, such as: “for (I=0; I<count; I++) { PORTC = 0xFF; }”) corresponds to the minimum rotation position of the servomotor, and a maximum count of 120 corresponds to the maximum rotation of the servomotor (slightly over 180 degrees of rotation). The six servos, and anticipated two additional servos for inter-segment link control, were driven off the Port C output pins of the 6811.

Sensory Inputs

Three types of sensory input devices are attached to each segment: momentary switches at the bottom of the legs (ground contact), bump sensors on the edges of the legs (object collision with the leg), and IR emitter/sensor pairs (approaching object, terrain computation). After the “Sharp Sensor Hack” was applied to the Sharp GP1U58Y IR detection units, as developed by Ariel Bentolila and refined by Dr. Keith L. Doty, the Sharp sensors could generate an analog distance reading instead of a digital On/Off signal. The analog signal is accurate to approximately 100 to 300mm, with about 6mm

accuracy. The segment designed in this report sits above the ground in normal operation approximately 6" (~150mm), so this resolution was sufficient. The bottom four IR detectors were connected to the MC68HC11E2FN's A/D (Port E) input pins 4-7. The top four IR detectors were connected to the A/D input pins 0-3. By toggling the control bits, the appropriate bank of IR detectors was sampled during operation. The IR emitters were driven using the code developed by Ivan Zapata from Port B, modulated at 40KHz. The top four IR emitters were connected to Port B pins 0-3, and the bottom 4 IR emitters were connected to Port B pins 4-7. The contact sensors (one per side) and the bump sensors (two per side) were designed as digital inputs, connected to Port A on the 68'11.

Materials

After the quick force calculations, it was apparent that significant amounts of force would be applied to the structure of the robot. With various machine shops quoting between \$1200 and \$2500 to manufacture *one* segment of this robot, the author decided to build the unit by hand, borrowing tools, garages, and various friends (see "Acknowledgments"). The first fully function prototype (Prototype #7) was made of ¼" extruded perspex plexiglass. This material is extremely strong, yet flexible. It is almost impossible to break under high force loads. At \$1.50 per square foot, and each segment using about 1.5 square feet of material, this was a very cheap solution. This material can be best obtained from your local plastics company. If you can obtain cast ¼" plexi, use it instead -- it is easier to machine. The following are words of advice for working with this material:

Use the glue the plastics manufacturer recommends. It is a monomer/solvent combination that first dissolves, and then re-bonds, the pieces of plexi back together. It is incredibly strong material.

You can not undo what you have done with glue. Once you glue something, it is permanent (unless it is a very small surface area). You have to throw the material away and start over. (The author used about 15 square feet of plexi to generate 1.5 square feet of usable prototype. Having built the first one, the others can be made with about 5 square feet each. Anticipate making a lot of mistakes.)

To cut this material, use a table saw with a carbide tipped blade -- the more teeth, the better the cut. If you can get a 10" table saw blade with at least 80 teeth, you will have smooth and even cuts. Fewer teeth means the plexi will melt and shatter.

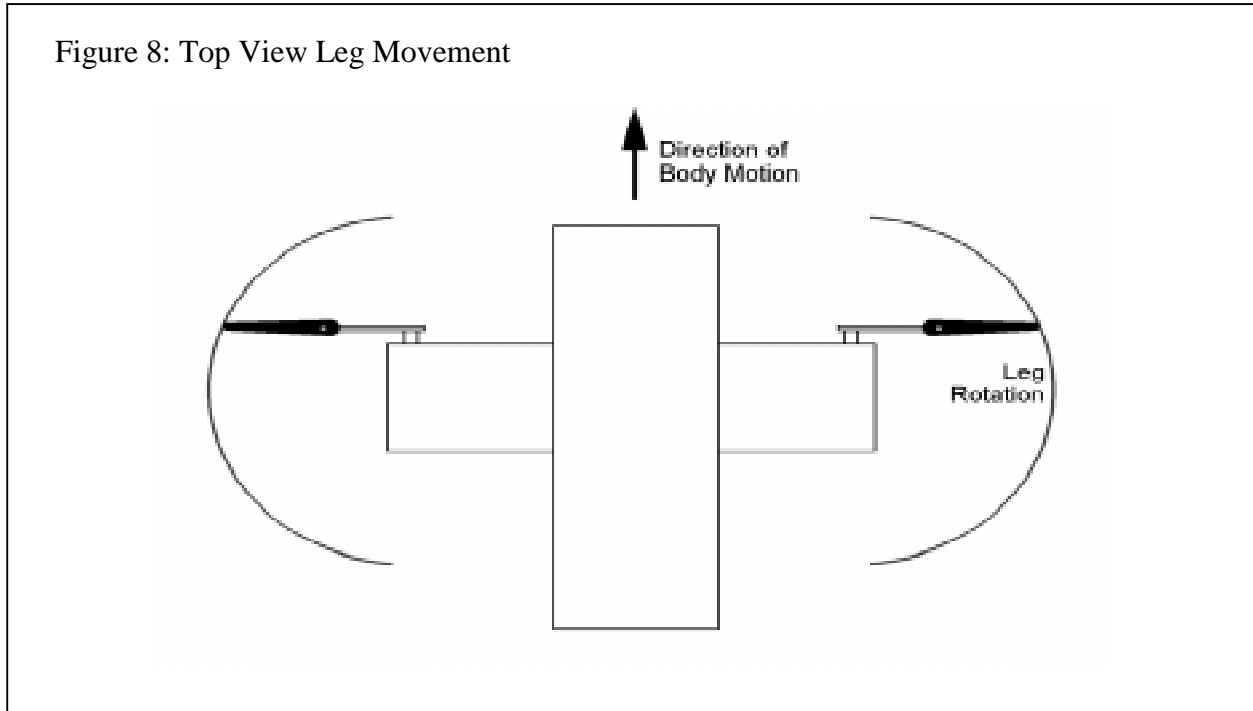
To drill this material, you must have very clean, very sharp bits. Use "metal" bits. Do not use the "plexi" bits -- they don't work. A drill press is very useful. You must drill very slowly, or else the plexi melts around the bit, and either shatters the plexi, shatters the drill bit (if less than 1/4" diameter), or melts the plexi to useless deformity.

To place a screw in this material, drill the hole the exact diameter of the screw. It will melt slightly and fill back in to provide a tight lock. Before inserting the screw, apply the solvent/monomer glue thoroughly inside the pre-drilled hole and let it sit for 10 - 20 seconds before screwing together. You will inadvertently break many screws if you don't follow this process exactly. Even with these steps, you will strip ~1/2 of the screw heads.

Once you screw something together, you can not unscrew it and then screw it back together again. Screws, like glue, are one-shot items with plexiglass.

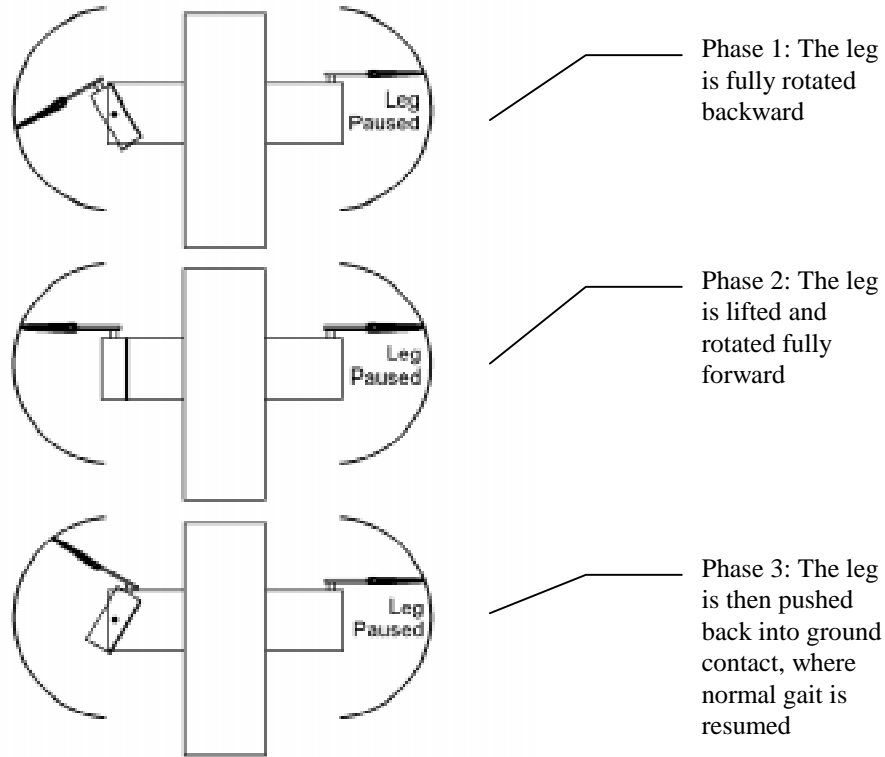
Section 4: Algorithm Development

In order to evaluate a programming solution to the gait of a walking robot, it is helpful to first examine gait visually.



As can readily be seen, the twin leg servo stacks will rotate freely. While in contact with the ground, the servo stacks rotate backward to push the body forward. Once the maximum backward rotation is reached in a leg, the opposite leg is frozen; the fully extended leg is lifted off the ground, rotated back to the front, and pushed back into contact with the ground. The normal walking procedure then resumes. This can be seen in Figure 9.

Figure 9: Top view of leg rotation



Gait

It is observed that the walking algorithm for a legged robot is simultaneously simple and complex. The servo actuators are driven in a constant pattern, assuming no obstacles, which is easy to reproduce in a simple looping algorithm. Yet it is also complex, for firm contact with the ground must be checked at all times, and any encountered obstacles must be avoided. A first-pass walking algorithm can be demonstrated by the following C pseudo-code:

sub-process Walk:

```
while (WALKING_FORWARD) {
    if (!LeftLegAllBack && !RightLegAllBack) {
        rotate_leg( LEFT_LEG, degree_increment );
        rotate_leg( RIGHT_LEG, degree_increment );
    }
    else
    if (LeftLegAllBack && !RightLegAllBack) {
        lift_leg( LEFT_LEG, LIFT_UP_FULLLY );
    }
}
```

```

        rotate_leg( LEFT_LEG, LEFT_ZERO_POSITION );
        rotate_leg( RIGHT_LEG, degree_increment );
        lift_leg( LEFT_LEG, PUSH_DOWN_TO_CONTACT );
    }
    else
    if (RightLegAllBack && !LeftLegAllBack) {
        lift_leg( RIGHT_LEG, LIFT_UP_FULLY );
        rotate_leg( RIGHT_LEG, RIGHT_ZERO_POSITION );
        rotate_leg( LEFT_LEG, degree_increment );
        lift_leg( RIGHT_LEG, PUSH_DOWN_TO_CONTACT );
    }
    else
    /* Both legs are fully extended to the rear */
    {
        lift_leg( LEFT_LEG, LIFT_UP_FULLY );
        rotate_leg( LEFT_LEG, LEFT_ZERO_POSITION );
        lift_leg( LEFT_LEG, PUSH_DOWN_TO_CONTACT );
        lift_leg( RIGHT_LEG, LIFT_UP_FULLY );
        rotate_leg( RIGHT_LEG, RIGHT_ZERO_POSITION );
        lift_leg( RIGHT_LEG, PUSH_DOWN_TO_CONTACT );
    }
}

```

This walking algorithm assumes a separate arbitration process which determines the status of each attempted action. The arbiter first determines if the “Walk” process should operate at all (there may be obstacles to navigate), and then maintains the state of the walking system by controlling the algorithm in two separate ways. First, by disabling the boolean “WALKING_FORWARD” bit in a boolean bit-vector, the walking subsystem will complete the current iteration of the loop and then halt. This can be useful when decided on a new direction of travel, or when it is necessary to investigate an “interesting” object. Second, the arbiter can freeze the walking subsystem by refusing to return control when a leg subroutine is called. For example, if a rotate_leg(..) fails because of an obstacle, the arbiter can activate a new process “ObstacleAvoid” or “GroundSeek” instead of allowing control to return to the walking subsystem. This is useful for immediate action situations where it is imperative the robot stops all current behavior.

The key idea to this walking algorithm is to have all legs moving backwards as much as possible. While each leg is in contact with the ground, the “shoulder” joint

servos rotate backward propelling the robot body forward. When the maximum backward rotation is reached, the leg is lifted and rotated to the maximum forward rotation. It is then pushed back to the ground to reestablish a load-bearing contact. It is important that only one leg be lifted at a time for stability. As each leg is lifted, significant torque forces (as high as 30 lbs-in) are generated at the segment joints and lifting multiple legs can increase these forces by a factor of four with each additional leg lifted.

In such a multisegmented system as Antaeon, keeping all but one leg on the ground at all times is a stable gait mode. The weight of all combined segments is distributed approximately evenly through those legs in contact with the ground. However, to assist the robot in keeping a wide distribution of legs, the range of motion of the legs is divided by the number of legs in the system to obtain a collection of unique starting points. This assures that by forward walking alone, no two legs will ever need to be raised simultaneously. Each leg has a unique starting angle in the rotation cycle that avoids having any two legs over time ending in the same position.

While this is a good initial condition, we still code the subprocess “Walk” to handle the case of two legs needing to be moved forward at the same time. To reduce “brittle” engineering, it is imperative to recognize at the outset that as obstacles are navigated and slippage with the ground occurs, it is likely over time (as time goes to infinity) that two legs will end up in this condition.

An important optimization to this basic algorithm is to rotate the leg opposite of the lifting leg while the lifting leg is in the air. That is, as one leg lifts off the ground, the opposing leg performs its specified rotation forward. This avoids a change in the

movement vector that would occur if the leg were immediately placed on the ground. Since the lifted leg never pushed the body forward, yet the opposing leg does, it forces a linear drift in direction to the side of the lifted leg. Rotating the opposing leg such that its forward push is ended by the time the lifted leg comes back into contact with the ground avoids this problem.

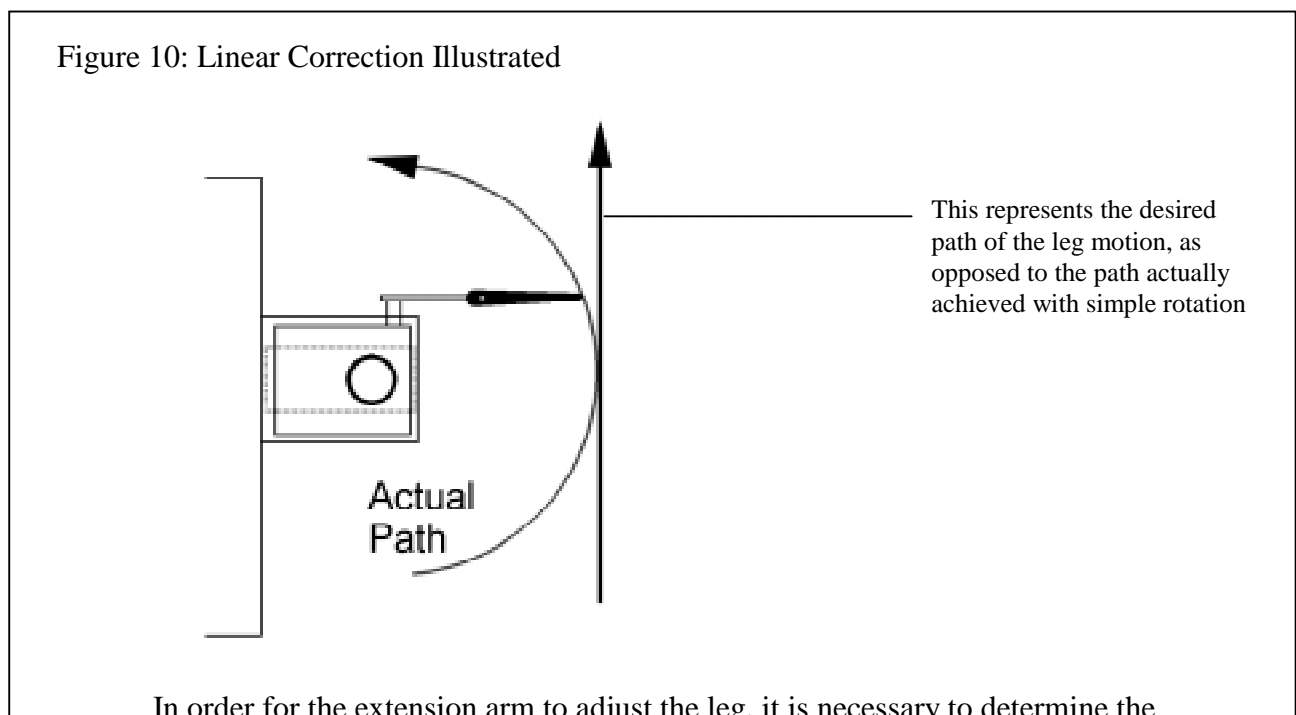
An obvious disadvantage of this system appears when the code is actually used. Since the rotation/lift subroutines are “blocking” (they don’t return control until the routine is finished executing), each rotation occurs sequentially in time. This causes each leg to subtly push against the other, much as the case of rotation while leg lifting, inducing repeated changes in the direction vector which cause the robot to “waddle.” As the left leg rotates backward, the motion moves to the right; as the right leg rotates backward, the motion moves to the left.

By modifying the rotation subsystem to be an interface to a more complex, automatic process, this can be avoided. The calls to “rotate_leg” and “lift_leg” are replaced with calls which setup bitmask patterns that code what action a leg is to take on the next motor-drive cycle. They therefore return control immediately, without waiting for the leg to perform a specified action. A new process, subprocess “MotorDriver”, launches every 32.77ms and totally consumes the CPU for approximately 5ms driving the servomotors to their specified locations. A new routine is added to handle special timing cases, “WaitForMotorCompletion()” which is blocking until the leg action finishes. This is inserted around the leg lift code to avoid the case of the bitmask patterns overwriting each other before the rear leg can be lifted, rotated forward, and then placed back into contact with the ground.

Linear Travel Adjustment

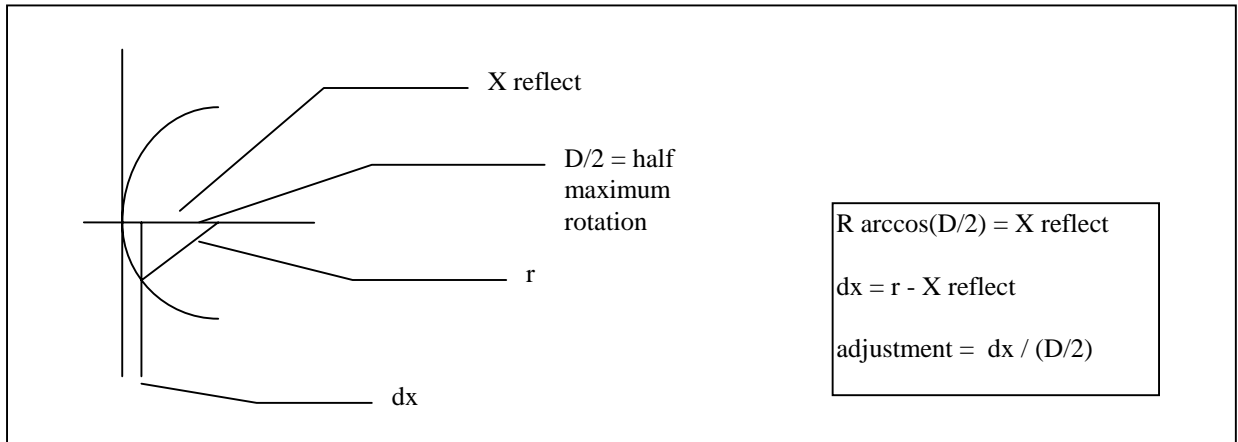
A less obvious flaw in the subprocess “Walk” is due to the geometry of Antaeon’s design. Since each segment’s leg control systems are mounted on vertical pivot points, with an extension arm running from the servos to the actual leg, all motion is circular. When a leg rotates backwards, the motion it describes on the ground is an arc, not a straight-line path.

This results in a wobble while walking. Each leg is pushing or pulling against the other legs, and the robot body is constantly undergoing many minute changes in direction. This can only be compensated by manipulation of the elbow servo joint to extend and retract the leg as the leg rotates about the body. This problem can be seen in Figure 10.



ratio of adjustment per degree of rotation. (Note that degrees of rotation are really measured by cycles in pulse-width-modulation of the servomotor control line.) This is

then used to modify the subprocess Walk into the version derived after the following calculations:



The results of this simple geometric analysis, combined with the previously discussed modifications, leads to the following C pseudo-code:

sub-process Walk:

```

while (WALKING_FORWARD) {
    if (!LeftLegAllBack && !RightLegAllBack) {
        rotate_leg( LEFT_LEG, degree_increment );
        extend_leg( LEFT_LEG, extend_adjustment );
        rotate_leg( RIGHT_LEG, degree_increment );
        extend_leg( RIGHT_LEG, extend_adjustment );
    }
    else
    if (LeftLegAllBack && !RightLegAllBack) {
        lift_leg( LEFT_LEG, LIFT_UP_FULLLY );
        rotate_leg( LEFT_LEG, LEFT_ZERO_POSITION );
        rotate_leg( RIGHT_LEG, degree_increment );
        extend_leg( RIGHT_LEG, extend_adjustment );
        extend_leg( LEFT_LEG, extend_adjustment );
        WaitForMotorCompletion();
        lift_leg( LEFT_LEG, PUSH_DOWN_TO_CONTACT );
    }
    else
    if (RightLegAllBack && !LeftLegAllBack) {
        lift_leg( RIGHT_LEG, LIFT_UP_FULLLY );
        rotate_leg( RIGHT_LEG, RIGHT_ZERO_POSITION );
        rotate_leg( LEFT_LEG, degree_increment );
        extend_leg( LEFT_LEG, extend_adjustment );
        extend_leg( RIGHT_LEG, extend_adjustment );
        WaitForMotorCompletion();
        lift_leg( RIGHT_LEG, PUSH_DOWN_TO_CONTACT );
    }
}

```

```

    }
    else
    /* Both legs are fully extended to the rear */
    {
        lift_leg( LEFT_LEG, LIFT_UP_FULLLY );
        rotate_leg( LEFT_LEG, LEFT_ZERO_POSITION );
        extend_leg( LEFT_LEG, extend_adjustment );
        WaitForMotorCompletion();
        lift_leg( LEFT_LEG, PUSH_DOWN_TO_CONTACT );
        lift_leg( RIGHT_LEG, LIFT_UP_FULLLY );
        rotate_leg( RIGHT_LEG, RIGHT_ZERO_POSITION );
        extend_leg( LEFT_LEG, extend_adjustment );
        WaitForMotorCompletion();
        lift_leg( RIGHT_LEG, PUSH_DOWN_TO_CONTACT );
    }
}

```

Turning

When the robot encounters an obstacle and needs to avoid collision or entrapment, it is necessary for the robot to turn away from the “danger.” Turning can be regarded as a slightly modified case of walking – instead of walking forward with each leg traveling the same distance, one leg travels less.

If the right leg moves 4”, but the left leg moves only 2”, simple geometry tells us the robot will veer 45 degrees to the left. Similarly, moving the right leg a subset of the left leg induces a turn to the right. Therefore, if we consider walking to be movement between the values of (for the left leg) “LEFT_ZERO_POSITION” and “LEFT_MAXIMUM_POSITION”, then we can turn to the left by storing these original values in temporary variables, and reassigning these to lesser values. The urgency of the turn (object is imminent, object is far away, object is intermediate) can govern the degree of the turn by controlling the limits used. As the two limits come together (from ZERO to MAXIMUM and from MAXIMUM to ZERO), the turn becomes sharper.

Computing the Plane

The only other complex algorithm development was determining the planar vector of the surface under the segment. This information will be used when the full robot – all three segments – is completed. In order to maintain stability between segments, it is important that each segment understand the terrain underneath itself and try to stay parallel to that terrain. This reduces risks of unbalancing and helps minimize the amount of forces transferred between the segment joints.

Only three IR emitter/sensor pairs are required for this calculation, but using four provides us with additional information. Not only can we monitor the surface underneath the segment, the four pairs – one in each extreme of the bottom segment body plate – can also provide very accurate information about the distance to the ground to detect imminent ground collisions, or for other (presently unforeseen) expansion.

The four IR pairs are quickly divided into two groups – the front-back pair and the left-right pair. By computing the slope of the ground in each pair, the segment can quickly adjust the appropriate leg or inter-segment connection to balance. The pseudo-code is as follows:

```
sub-process ComputePlane:
front = analog(BOTTOM_FRONT);
back = analog(BOTTOM_BACK);
left = analog(BOTTOM_LEFT);
right = analog(BOTTOM_RIGHT);
fb_slope = ((front-back)/back * 100);
lr_slope = ((left-right)/right * 100);
```

Computing the slope as the “`abs(front-back)/back`” places the slope in relative comparison to the distance of the robot above the terrain. As the robot gets closer to the

surface of the ground on either side of the pair, the slope becomes larger indicating tilt.

When the slope is negative, it indicates the tilt is to the latter direction of the pair (back or right); when the slope is positive, the inclination is to the forward direction of the pair (front or left). The final multiplication is simply to put the slope in integer terms rather than perform CPU costly floating point calculations.

Section 5: *Future Directions & Project Evaluation*

To complete this project, the remaining two segments need to be built. Prior to this, torsion springs need to be found which counter appropriately the lift forces needed for this design. With the torsion springs installed on the center segment, and the design verified, the three segments need to be connected together.

Control software for the inter-segment joint will be extremely complicated. A nonlinear system is suggested, with overall system stability as the goal. A finished SPI network needs to be installed between each segment, so that messages can be relayed and the robot can reach a consensus for how to act, rather than being driven by a central “brain”.

The gait needs to be modified to allow arbitrary extension of the lower leg while walking, rather than the current “fixed” model where the leg travels a known path with known angles. The advantage is when traveling porous (pot-holes, horizontal ladder) surfaces, the leg can still be used to propel the body forward if the leg extends farther out from the body.

The class in general was an excellent tutorial on how to think and build real-world non-brittle projects. Unfortunately, the scope of the project the author chose to attempt was far too large to be done in a single semester. There were too many things that had never been done before that had to be solved. While this hampered gaining true momentum (always building another prototype, trying to reduce weight, or size, or moment arms) in creating a fully functional robot, the author was pleased with what was accomplished. Several hundred man-hours of labor (in excess of 300 machining and assembling parts alone) went into this project. When this is complete, I hope to be able to release technical diagrams in AutoCAD such that anyone can build this robot, to learn from it -- and with it. Until then, the work must go on.

Section 6: *References*

- [1] Joseph L. Jones and Anita M. Flynn. *Mobile Robots: Inspiration to Implementation*
K. Peters 1993
- [2] Pattie Maes. *Designing Autonomous Agents*
MIT / Elsevier 1990
- [3] Fred Martin. *The 6.270 Robot Builder's Guide*
Epistemology and Learning Group, MIT 1992
- [4] M68HC11 Reference Manual
Motorola 1991
- [5] MC68HC11E9 Technical Data
Motorola 1991
- [6] High-Speed CMOS Data
Motorola 1993

Acknowledgments

Much appreciation in this project needs to be conveyed from myself to the people who help me through the obstacles and kick around design ideas. Without these friends, none of this would be anywhere near as good as it is becoming:

David and Sara Perez, whom I am a roommate of. Dave is a self-labeled ‘frustrated engineer’ because he chose to not become an engineer while attending UF, but loves to tinker and explore the mechanics of the world around us. While in the middle of a design implementation, Dave frequently would stop and demand to be shown how the unit works and then a lengthy argument of pros/cons would ensue. Sara has my undying gratitude for tolerating the mess of parts and space I go through, in addition to weird hours with lots of machine noises, while constructing this robot. Plus, she’ll feed me when I don’t have time to feed myself. They can be reached via “perezd@cps.acast.nova.edu”.

Pete Taylor, a good friend and student at UF who can’t decide if he wants his second degree to be in ME, CE, EE, CEN, or perhaps Forever-Student-Of-Everything. His insights into simple rearrangements of components to reduce forces and space have solved many a 2a.m. problem. His hours are almost as weird as mine. Pete can be reached via “pete@cop.health.ufl.edu”.

Chuck and Laurie Higgins, who let me come over to their house and destroy the garage several weekends running. Chuck is an “I can build anything” kind of guy, and has collected almost a complete machine shop in his garage over the years. Most of his material he salvaged from fires and rebuilt. Without his excellent set of tools and periodic offerings of insight, this project would be a pale imitation of what it is. Laurie patiently put up with my causing her husband to become obsessed with plexiglass, robots, and servomotors.

Livewire, Inc., specifically my employers *Mel Paulick*, *Chuck Higgins*, and *Ted Grauch*. Aside from keeping me gainfully (and legally) employed, the work I do for them has paid for this project so far. I am hoping it will continue to pay for this project as I continue working on it. These three gentlemen have been extremely tolerant of my wild working schedule and when I drag my robot parts in to the office to test things on “the scope” or use my office system to write robot software. They even let me use the whiteboards to draw schematics or sketch crazy ideas.

Mark Dinsmore and Chelsea Jones, two very good friends who forced me to take a break now and then when things were a little too intense. Mark is always up for a game and is much better at graphics programs than I am for making diagrams and figures. Chelsea just forces Mark to force me to relax. Unfortunately, they moved to Austin, and now it’s all work and no play.

Professor Keith L. Doty, an instructor and guide who firmly believes that I can do what I try to. He doesn't believe in the word "no" and is a veritable fount of knowledge, when you know to ask the question – or know that there is a question waiting to be asked. With any luck, I won't disappoint him in this project. Dr. Doty can be reached via "doty@mil.ufl.edu".

Thing, the quadruped robot created at the University of Massachusetts. Thing is similar in design to my final result, and some elements of the leg design were borrowed from Thing to make space usage more efficient. The creator of Thing also worked out all the kinematics of the stacked elbow system, which I was able to use with slight modifications to calculate the forces applied to the servos. Thing can be found via the URL: "<http://www.piglet.umass.edu:4321/thing/thing.html>".

And, all of the TA's and classmates of the Intelligent Machine Design Laboratory.

If I had had the time and the money, this project would have been unstoppable. If only congress would make the weekday 30 hours and the week 8 days...

Appendix: Source Code

This is stored in a separate file on the associated floppy disk. It is quite extensive to print out, and will be changed before the demonstration. The early version of the source code is in the file “appx_c.doc”. A new “appx_c.doc” will be provided on demonstration day.