

Going Straight: a Lego-based Minirobot Drive System

E. Cervera

Robotics and AI Lab,
Jaume-I University, Campus Riu Sec, E-12071 Castelló, Spain

E-mail: `ecervera@icc.uji.es`

Abstract. Most minirobot platforms have the ability to perform translational and rotational motions in a decoupled way. This is an important feature, which allows the robot to go to any desired position and / or orientation. However, in practice, real robot motions are far from accurate, and control strategies using either internal or external sensors must be implemented, e.g. to achieve a straight trajectory. Such a trajectory is very helpful in structured environments like rooms and corridors. We present a mechanical design, with a moderate complexity, which is capable of making the robot go straight while still allowing it to turn in place. The resulting robot is a differential drive design, where motors do not drive directly the wheels: one of them makes the robot move forward or backward, and the other motor makes it turn in place. Of course, both motors can be run simultaneously. Experiments demonstrate the feasibility of the design, and the quality of the achieved trajectories. This platform considerably alleviates the difficulties of robot control, allowing the user to concentrate in higher level tasks.

1 Introduction

Though some tasks do not rely on a precise trajectory, e.g. a Sumo combat, the ability of the robot to go straight can be very helpful in a structured environment composed of rooms, corridors, and doors. Though sensor feedback cannot be neglected, if the underlying machinery is capable of maintaining a programmed trajectory, the problem of control is alleviated, specially in small robots, where available computing power is limited.

Most small educational robots use a differential drive mechanism, a perfect solution if both wheels rotate *exactly* at the same velocity. In the real world, however, to make such robots go straight can be a real pain, consuming a lot of time and effort in order to implement an appropriate feedback control loop.

In this paper we present our experiences with autonomous mobile minirobots. Through several refinement steps, we have come out with a small robot platform which is able to move along a straight trajectory, while keeping the ability to turn in place.

The rest of the paper is organized as follows: Section 2 describes the elements of our robot platform; next, Section 3 briefly resumes the experiences carried out with such robots. The

problem of going straight is presented in Section 4, together with the most common drive configurations for minirobots. Section 5 proposes different solutions to the problem, mostly based on feedback loops, and our new approach based on a special mechanical design. Experimental results demonstrate the feasibility and quality of this new approach. Finally, Section 6 presents some conclusions and future work.

2 The Robot

The elements of our autonomous mobile minirobot platforms are presented. The overall robot is rather inexpensive, highly configurable, and it has proven to be very robust. Moreover, it provides enough computing power for some rather interesting AI algorithms (e.g. Q-learning, or a basic subsumption architecture).

2.1 Electromechanics

Robot structure is made completely of Lego parts. Our most common design has diametrically opposed drive wheels and two free-wheeling castor. Each wheel is driven by its own motor, thus allowing the robot to go forward, backward, or turn in place. This is a classic differential drive configuration, which is thoroughly described in Section 4, together with other alternative designs.

Besides the Lego parts, only miscellaneous electronic components were used, e.g. bumpers, IR proximity sensors, IR range finders, photo-cells, etc.

2.2 Hardware

Autonomous robot control is achieved by a Handy Board [4]: it is a 68HC11-based controller board designed for experimental mobile robotics work. MIT has licensed the Handy Board at no charge for educational, research, and industrial use.

Besides the Motorola MC68HC11 processor, the Handy Board includes 32 K of battery-backed static RAM, four outputs for DC motors, a connector system that allows active sensors to be individually plugged into the board, an LCD screen, and an integrated, rechargeable battery pack.

The board can be upgraded with an Expansion Board which plugs on top of it, providing additional features, like additional analog sensor inputs, active Lego sensor inputs, digital outputs, servo motor control signals, and a connector mount for Polaroid 6500 ultrasonic ranging system.

2.3 Software

A wide range of options are available for developing software on the Handy Board, including free assembly language tools provided by Motorola, and commercial C compilers.

Additionally, the Handy Board is compatible with *Interactive C*, the programming environment created for the MIT Lego Robot Design project. Interactive C (IC) is a multi-tasking, C language based compiler that includes a user command line for dynamic expression compilation and evaluation.

3 Laboratory Experiences

The presented experiences have been carried out in the MSc Program in Computer Science at Jaume-I University. First, two experiences in the Robotics course are presented: kinematics and sensor-based control. Next three experiences belong to the AI course: reinforcement learning, subsumption, and agents.

3.1 Mobile Robot Kinematics

A rotation sensor was attached to each wheel. The goal was to construct the direct and inverse kinematics of the mobile robot. The model itself is rather simple [5], but the low precision of sensors together with the poor control of motors makes the problem rather difficult.

Nonetheless, several interesting results were achieved, e.g. trajectory control made possible to program the robot to go straight.

3.2 Sensor-driven Control

The goal of this experience was to program the robot for a Sumo tournament. The robot had to be controlled mainly from its sensor inputs, so a state finite automaton was designed, where transitions were triggered by sensor conditions.

Robots had line sensors for detecting the ring boundary, bumpers, and an IR range finder for detection of the opponent.

3.3 Reinforcement Learning

Reinforcement learning [8] is a computational approach to learning whereby an agent tries to maximize the total amount of reward it receives when interacting with the environment.

Robots had to learn a line following behavior. A rather simple task was chosen in order to see the learning progress in a small amount of time. Instead of *hard-coding* a line following algorithms, students programmed a reinforcement learning algorithm which actually learnt to control the robot by its own experience, following different lines marked on the floor.

3.4 Subsumption Architecture

Brooks' subsumption architecture [1] consists of behaviors, i.e. layers of control systems that all run in parallel whenever appropriate sensors fire. Parallel behaviors can be easily implemented in IC on the Handy Board, using its parallel processing primitives.

Different sensors (bumpers, proximity, light) were mounted on the robot, and students had to design and implement a layer of control systems. The resulting robot had an *emergent* behavior: it wandered around the laboratory, avoiding obstacles and either following or escaping from the light.

3.5 Agent-based Architecture

According to the *intelligent agent* view [7], the problem of AI is to describe and build agents that receive percepts from the environment and perform actions. Robotics is not defined independently: this approach emphasizes the task environment characteristics in determining the appropriate agent design.

The goal task was inspired by the Trinity College Fire Fighting Contest: robots had to find and extinguish a fire in an office-like environment.

4 The Problem of Going Straight

Though some tasks do not rely on a precise trajectory, e.g. a Sumo combat, the ability of the robot to go straight can be very helpful in a structured environment composed of rooms, corridors, and doors.

In this section we present a review of the most common drive designs for wheeled robots, their advantages and drawbacks.

For a wheeled robot, the designer may choose among several different arrangements of driven and steerable wheels. Among these arrangements, thoroughly described by Jones et al. in [3], are differential drive, synchro drive, tricycle drive, and car drive (also known as *Ackerman steering*).

4.1 Differential Drive

From both programming and construction standpoints, differential drive can be one of the least complicated locomotion systems. This scheme consists of two wheels on a common axis, each wheel driven independently. Such an arrangement gives the robot the ability to drive straight, to turn in place, and to move in an arc.

Figure 1 depicts the right motor and wheel of a simple differential drive mechanism constructed with Lego parts. It should be noted that this Lego motor has internal reduction, thus there is practically no need of more gears.

However, how to make the robot go straight is far from direct. Even when the same voltage is applied to the two motors, they will turn at different speeds and the robot will veer to one side or the other. To make the robot go straight, we must ensure that the wheels turn at the same velocity.

4.2 Synchro Drive

With such a mechanism, all wheels both steer and drive. They are linked in such a way that all point in the same direction at all times. In order to change direction, the robot simultaneously rotates all wheels about a vertical axis. The synchro scheme overcomes many of the problems of differential, tricycle, and car-type drives *at a cost of greater mechanical complexity*.

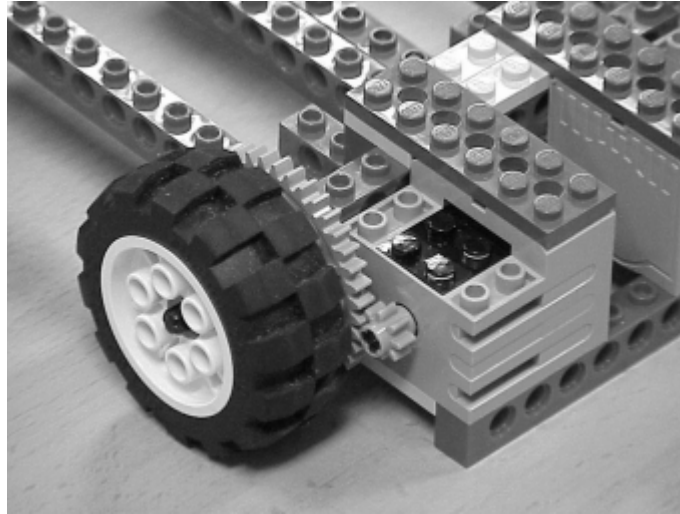


Figure 1. Simple differential drive, right wheel.

4.3 Car and Tricycle Drives

Car-type drive (Ackerman steering), with its four points of suspension, provides good stability. Tricycle drive has a similar feature, with the advantage of being mechanically simpler. In general, for both tricycle and car drive, the two fixed wheels will be connected to a drive motor and the steerable wheel(s) will not be driven.

With car and tricycle drive, it is not necessary to monitor wheel velocity in order to make the robot go straight. Simply positioning the steerable wheel at its neutral position is sufficient. This simplicity, however, comes at a price: *differential and synchro drive robots have a subtle advantage over car and tricycle drive types*; the difference is their kinematics.

With car and tricycle drive, the robot's orientation and its position are coupled: in order to turn, it must move forward or backward. The robot cannot go directly from one position and/or orientation to another, even if nothing is in the way. However, a robot based on differential or synchro drive can, by turning in place, effectively decouple its position from its orientation.

5 How to Make a Robot Go Straight

For its simplicity, and the capability of turning in place, a differential drive design is chosen. In this section we present different methods to achieve a straight trajectory.

Three different solutions are proposed: using internal sensors to feed back motor velocities, using external sensors to feed back a landmark, and a mechanical design based on an *adder / subtractor* of motor power.

5.1 Using Internal Sensors

The goal is to make both wheels turn at the same velocity. This solution consists of a *closed-loop control algorithm*, getting feedback from internal sensors like the shaft encoders.

The basic idea is to take the desired velocity command, send the command to the motors, see how fast the motors actually spin, and then measure that speed and compare it to the commanded speed. The difference is called the error signal.

In the solution proposed by Jones et al. [3] and depicted in Figure 2, a proportional-integral controller is used. The top and bottom loops are proportional controllers because the difference between the desired speed and the actual speed is multiplied by a constant, and fed back to the motor to adjust the motor speed.

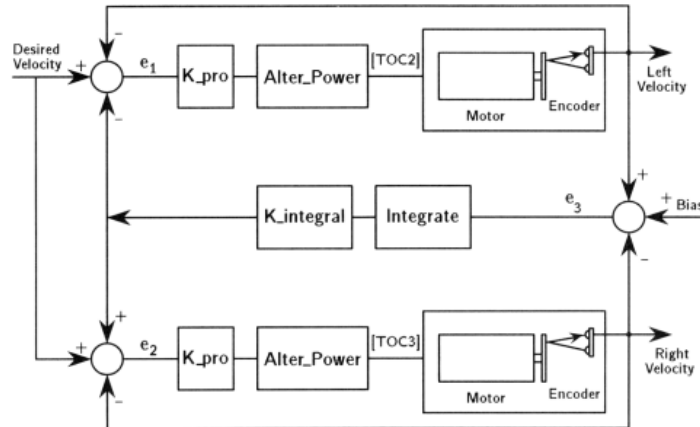


Figure 2. A proportional-integral loop to control the speed of the robot and synchronize two wheels so that the robot will travel in a straight line (from Jones et al. [3]).

The integral controller looks at the actual speeds of both motors and compares them. In this way, one motor is sped up while the other is slowed down until they reach speeds sufficiently close together. There are two input signals: desired velocity and bias. The bias term is used for inputting the turn command.

The main problems of this solution are the low resolution of Lego encoders, the response time of the loop, and the computing resources needed from the processor.

5.2 Using External Sensors

An alternative solution is to make the robot follow landmarks with its external sensors, e.g. distance to a wall. The approach is based also in a feedback control loop, but the control signal is now provided by an external sensor.

In a realistic minirobot environment, lines are not drawn on the floor, and vision is not available, thus most common landmarks are walls. Distance sensors can provide an accurate measurement to the control loop. However, the same problems of response time and processing power are presents.

We have used Sharp GP2Dxx detectors which are quite immune to ambient lighting conditions, and indifferent to the color of the object being detected. They use triangulation and a small linear CCD array to compute the distance and/or presence of objects in the field of view. However, the output of these detectors is non-linear with respect to the distance being measured.

Despite this non-linearity, a feedback control loop can be easily implemented. The main problem is the adjustment of the control gain: in order to keep a straight trajectory, a high gain must be selected, but sensor noise causes the robot to veer continuously from side to side of the trajectory line.

5.3 A Mechanical Solution

As opposed to the previous solutions, we wondered whether it was possible to change the power transmission while keeping the differential drive scheme. We aimed to drive both wheels from the same motor, and use a second motor for steering, much like the velocity control loop presented above.

The solution is based on the Lego *adder / subtractor* first designed by Leo Dorst [2], and modified by Mark Overmars [6]. As shown in Figure 3 two differentials are used between the motors and the drive wheels. Such a system simultaneously adds and subtracts the rotation of both motors: one wheel is powered with the resulting addition, while the other wheel is powered with the subtraction.

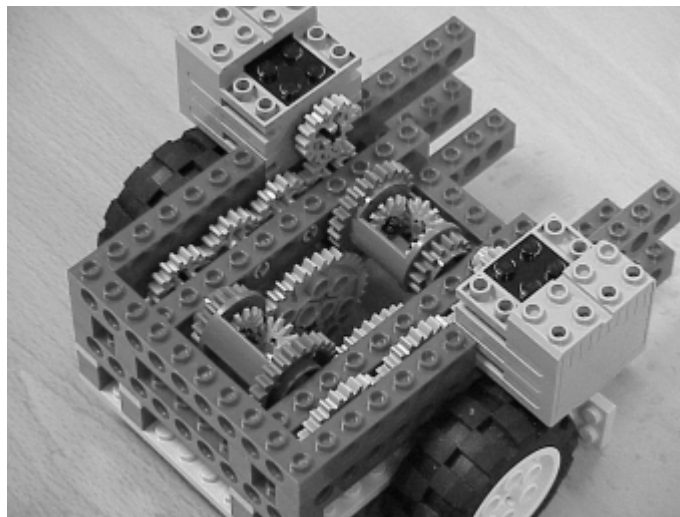


Figure 3. Design of adder/subtractor.

As a result, one motor is responsible of the translation velocity of the robot, while the other motor stands only for rotation. Of course, if this motor is stopped, the robot will travel along a straight line trajectory.

In practice, however, the robot turns in an undesired way. The reason is the friction between the wheels and the floor, which is not exactly the same for both. Though the rotation motor is stopped, it is not *blocked*, so the robot turns even if no voltage is applied to this motor.

The final solution uses a *worm gear* to transmit motor rotation to the differentials, as depicted in Figure 4. This gear has a particular nice property: if the motor stops, its axis cannot be back rotated by the wheels.

Only a small deviation is caused by the *backlash* of the gear train. The only minor drawback of this design is that there is a higher friction than if only tooth gears are used.

The final robot design, with the controller mounted on top of the robot, is depicted in Figure 5. This vehicle is very compact, almost fits in a square space, measuring approximately 19cm

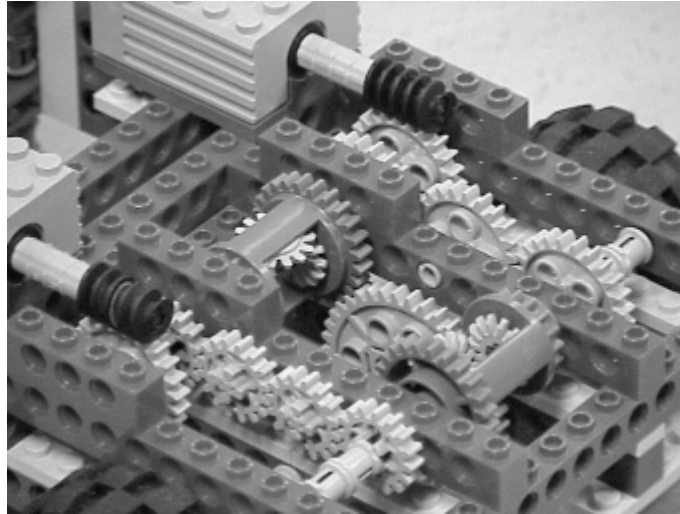


Figure 4. Final gear train, showing the worm gears attached to motor axes.

long, 16 cm wide, and 13cm tall (including the controller). There is enough free space left on the front of the robot to accommodate several sensors.

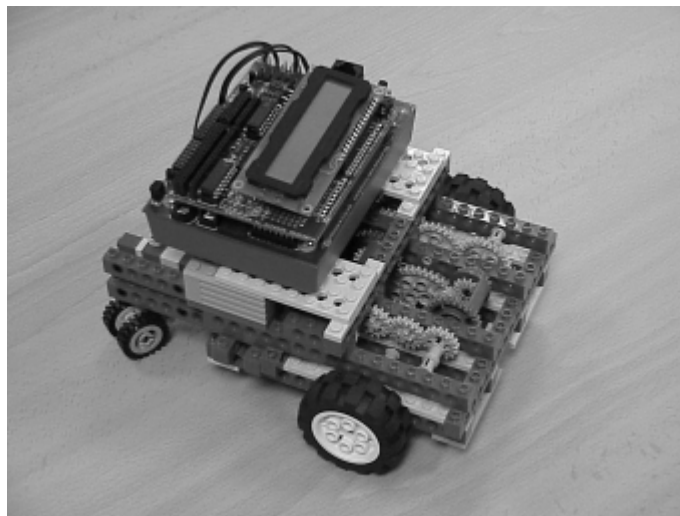


Figure 5. Final robot design, with controller.

We have carried out several experiments to test the quality of the trajectory, when commanded to go straight. As shown in Figure 6, the robot traces a nearly straight trajectory, as depicted by the trace of an attached pen.

Measurements on these trajectories are given in Table 1. As the traveled distance grows, the deviation caused by different velocities is increased too, since the robot describes an arc instead of a straight trajectory. The new design avoids this difference; in fact, only slipperiness can modify the motion of a wheel, but this problem is hardly controllable.

With respect to the traditional differential drive system, it has to be noted that only one motor is powered during translational motion, thus providing the robot with only half of the theoretical maximum power. As a result, it can move too slowly, but in our experiments it has been powerful enough for moving smoothly around a flat, leveled surface.



Figure 6. Trace of the robot for a commanded straight trajectory, using either a classic differential drive (left) or the new mechanical design (right).

Traveled distance	Deviation (classical)	Deviation (new)
20 cm	≈ 1 mm	< 1 mm
100 cm	50 mm	≈ 1 mm
120 cm	73 mm	≈ 1 mm

Table 1. Deviation from straight trajectories, for a classical differential drive, and for the new mechanical design.

Different levels of reduction can be applied to the gear train. The trade off is of course velocity versus force, and the optimal decision depends on the weight of the robot and whether obstacles can be found in its way. The tested vehicle has a 1:40 ratio between the motor and the wheels. With this reduction, the travel speed is about 8 cm/s, and the vehicle is stopped by a force of 0.3 kg. (values measured on a wooden table, with little friction).

6 Summary

Most small educational robots use a differential drive mechanism. To make such robots go straight can be a real pain, consuming a lot of time and effort in order to implement a feedback control loop.

Car and tricycle drive is appropriate for going straight, but it has an important disadvantage in that translation and rotation are coupled, thus making it difficult to work in indoor environments.

We have presented a mechanical solution with a moderate complexity, which is capable of making the robot go straight while still allowing it to turn in place. This design uses differentials for adding and subtracting the power of two motors. Worm gears prevent the motors from being back rotated by the wheels, thus guaranteeing a perfect straight trajectory.

The resulting robot is a differential drive design, where motors do not drive directly the wheels: one of them makes the robot move forward or backward, and the other motor makes it turn in place. Of course, both motors can be run simultaneously.

Experiments demonstrate the feasibility of the design, and the quality of the achieved trajectories. This platform can considerably alleviate the difficulties of robot control, allowing the user to concentrate in higher level tasks.

Future work includes attaching shaft encoders to the motors, and developing the direct and inverse kinematic model of the robot. We believe that our design makes possible the use of such models, since the position error is kept small, at least for short distances.

Acknowledgements. Support from Jaume-I University (*Educative Support Department*) and BP Oil Refineria de Castelló SA is gratefully acknowledged.

References

- [1] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23, 1986.
- [2] Leo Dorst. Lego adder / subtractor, 2000. URL: <http://carol.wins.uva.nl/~leo/lego/diff.html>.
- [3] Joseph L. Jones, Anita M. Flynn, and Bruce A. Seiger. *Mobile Robots: Inspiration to Implementation*. A K Peters, Ltd, 2nd edition, 1999.
- [4] Fred G. Martin. *Robotic Explorations: a Hands-on Introduction to Engineering*. Prentice Hall, 2001.
- [5] Philip J. McKerrow. *Introduction to Robotics*. Addison-Wesley, 1993.
- [6] Mark Overmars. Lego robots tips and tricks, 2000. URL: <http://www.cs.uu.nl/people/markov/lego/tips/index.html>.
- [7] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1994.
- [8] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.