13

Case Studies

13.1 ROBOCOW—A MOBILE ROBOT FOR TRAINING HORSES

The National Centre for Engineering in Agriculture received a startling project proposal concerning a "robot cow" that could fool a horse. As discussions progressed, it rapidly became clear that the design requirements would be extremely hard to meet. The purpose was the training of horses for cutting contests, where horse and rider must control the movement of a young cow. The business proposal was made by an acknowledged "cutting champion."

In heading off a young cow that is trying to rejoin the herd, the partnership of horse and rider depends entirely on the ability of the horse to recognize and anticipate the intentions of the cow. In training the horse, it has been usual to use a "borrowed" calf—or two, or three or more.

As fast as the horse learns, so the cow also learns and very soon refuses to cooperate. So, to train one horse takes the use of many cows and considerable expense. A robot cow, on the other hand, would be consistent and predictable by the rider if not by the horse.

Robocow, as it was quickly named, must perform a memorized sequence of actions, so that with no more than a two-button controller the horse rider can select, start, pause, or resume the cow's performance. In addition, it is important that when completing a sequence designed to bring it to the starting point, the cow can repeat the sequence several times before large positional errors are built up.

Some of the sequences can be preprogrammed during manufacture, but there is also the need to provide the ability to memorize special individualized routines on the farm. For this, a standard radio-control joystick system is used.

At the outset, the mechanical performance requirements were challenging, to say the least. The cow must reach a speed of 20 km/h with an acceleration of several meters per second per second. The terrain was specified as "beaten earth."

13.1.1 Overview

The selected geometry was a steered tricycle with driven front wheel—the same system as the fairground "dodgem." The steering can turn through half a circle, so that the cow can spin about the center of its rear axle. It can actually accelerate faster in reverse, when the weight is thrown onto the driven wheel.

Navigation of Robocow (Fig. 13.1) depends on odometry. The undriven rear wheels are equipped with Hall effect sensors that enable their angles to be monitored at all times. Heading is deduced from the difference between the wheel rotations, and the coordinates are estimated by integrating forward motion multiplied respectively by the sine and cosine of the heading.

Steering control uses a highly nonlinear algorithm that drives the system to a new setting in a fraction of a second, very similar to the position control discussed in Chapter 3. The overall result is a lively beast where performance is limited more by skidding on the dirt surface than by any limitations in the drives or controls.

13.1.2 Mechanical Design Considerations

Since the drive is applied through the front wheel, the two rear wheels are undriven and have no reason to slip in the direction of progress. They can therefore be used for reliable odometry, provided they do not leave the ground.

A single 120 W motor did not give the lively action the clients were seeking. With two such motors mounted on the front-wheel assembly, symmetrically placed fore and aft of the wheel, there was more than enough drive to skid the front wheel.

The steering uses another very substantial motor of 60 W rating, so that it can be driven from one extreme to settle at the other in well under one second.

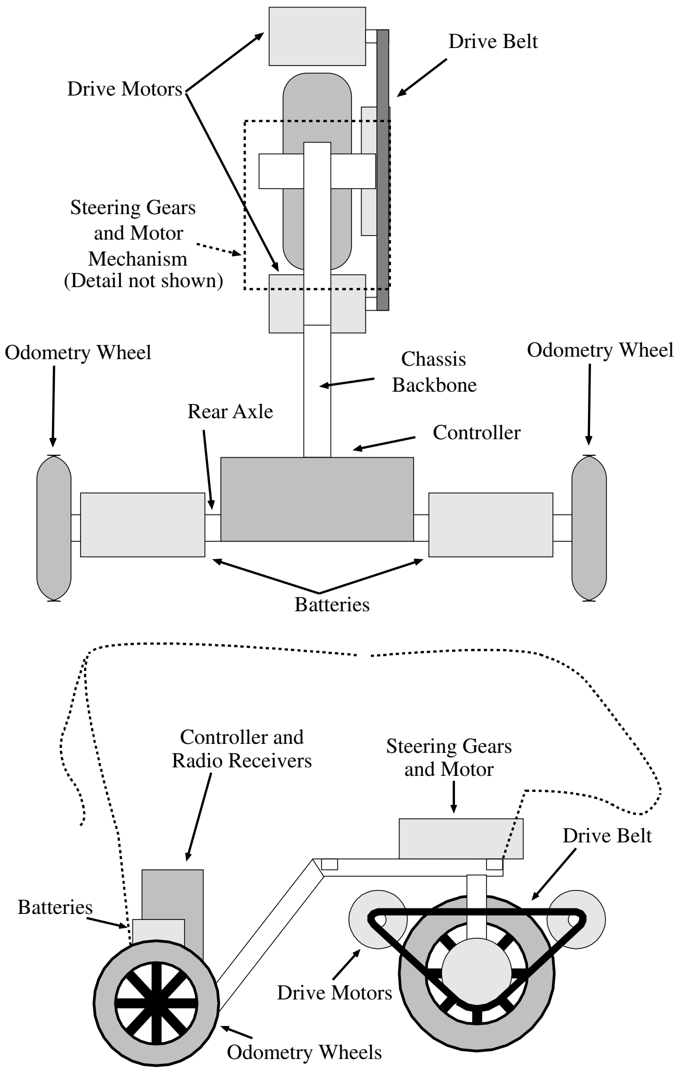

Figure 13.1 *Two diagrams of Robocow.*

Two 12V lead acid batteries are mounted symmetrically near the rear wheels. There is a tradeoff between wishing to keep the center of gravity over the driven wheel and the need to keep it aft to lessen the risk of rolling.

The diameter of the front wheel is 310 mm and of the rear wheels is 300 mm. The wheelbase is 585 mm fore and aft, while the rear wheels are 760 mm apart.

The body of the first prototype was formed by stretching a cloth "cow suit" over a light tubular framework. A much more realistic body has now been molded in polystyrene foam.

13.1.3 Operation and Control Design

The rider must be able to operate Robocow with a simple pushbutton controller with one-handed operation. A two-button motor-vehicle remote-locking device was used. With long and short presses, acknowledged by beeps from the cow, this gave all the command power that was needed.

For programming the "dances," a radio-control joystick was used, with fore and aft movement setting the speed and side-to-side motion commanding the steering. Extra controls such as those used for selecting "record" mode were mounted on the cow's rump.

By now we are starting to build up a substantial list of tasks for the microcontroller to perform. One approach might be to look for a sophisticated multitasking operating system, but the straightforward approach is much simpler. An HC11 was chosen with ample capability for the task itself, but it is not a device on which you would want to heap "system software." The tasks are

- Measure the angles turned by the rear wheels and calculate odometry.
- Measure the steering angle and its tacho to close the steering loop.
- Read the joystick signals.
- Read the two-button radio signals and obey them.
- Read and debounce the control switches on the cow's rump.
- Check progress and step through the stored "dance," controlling the speed.

The dance is stored as a sequence of segments. Each defines a steering angle and a target speed. By limiting the precision to 14 steps of speed and steering, represented by values in the range –7 to +7, the pair of values can be held in a single byte. For each segment there is also stored a termination condition in a second byte.

If the segment is a turning one, the termination condition defines the heading angle at which the segment ends. If the steering is required to be straight in the segment, its termination condition defines the aggregate distance to be covered until the next segment starts. Segments are short enough that using a single byte will identify the least-significant byte of the distance without ambiguity. If the speed command is zero, the condition determines the time that must elapse before continuing.

The termination conditions are absolute to avoid accumulating errors; that is, the distance termination is not simply the length of the segment but the

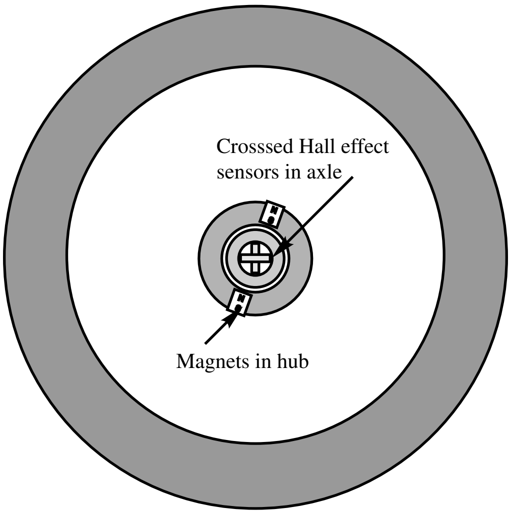

Figure 13.2 *Hall effect sensor, shown in a wheel.*

total distance covered since starting. Similarly, the heading is absolute, measured from the start condition in terms of the difference between left and right wheel rotations.

When required to run straight, any error in steering calibration could cause the path to be a large circle. This is avoided by applying heading error feedback to the steering, nudging it by an angle limited to a few degrees. It keeps the machine straight without being perceptible in its behavior.

13.1.4 Sensors and Control Loops

At the heart of both odometry and steering sensors is the simple Hall effect angle-sensing device mentioned in Chapter 2. Two UGN3504 analog magnetic sensors are mounted with their sense axes perpendicular to the axis of rotation and perpendicular to each other. In the case of the wheels, they are mounted within the rod that forms the wheel bearing. A magnet is mounted on the wheel with its axis radial, normal to the rotation axis (see Fig. 13.2).

The sensors therefore give signals representing the sine and cosine of the wheel rotation angle. In fact, two such magnets are used, mounted on either side of the axis, so that second-harmonic distortion is minimized if the sensors are not exactly axial.

The sensor signals are encoded directly by the 8-bit ADC channels of the 68HC11 microcomputer that controls the cow. A simple and novel routine extracts the angle from these two signals.

The routine is based on the approximation

$$\text{Angle} = \left(\frac{3.7(\sin - \cos)}{2.7 + \sin + \cos} + 1 \right) \frac{\pi}{4}$$

which is accurate to a fraction of a degree.

In the software, the angle is represented by a single byte as "binary degrees" or "begs," where 256 begs make one complete circle. The increments are thus slightly less than 1.5°.

The routine for calculating the angle is very simple. During an initial setup, the sensors have been calibrated to find the mean and amplitude of their variation, values now held in nonvolatile memory.

First the appropriate datum value is subtracted from each sensor signal and the sign is noted. This will determine the quadrant of the final result. Now the positive values `SINA` and `COSA` are calculated by negating these values, if necessary, and multiplying them by the corresponding calibration factor to normalize them to a range of 0 to 127.

The value π/4 corresponds to 32 begs, so the first-quadrant angle is now given by

```
237*(SINA-COSA)/(346+SINA+COSA)/2  +  32
```

since 2.7 * 128 = 346 and 64 * 3.7 = 237

The wheel angle is "extended" into a multibyte value that is long enough to hold the total number of wheel revolutions for a performance. If the new "local" value is within a count of 64 of the previous value, it is clear whether a carry or a borrow should be propagated into the higher bytes. If the change is greater than a quarter of a revolution, an error is indicated.

The alternative to this analog technique would be to use a bidirectional counter to count pulses from an encoder disk. In that case, unless a hardware counter is used, the service routine would have to interrogate the transducer more than 256 times per revolution of the wheel. With the analog sensor, four or more interrogations per revolution will be sufficient.

In the wheel angle routine, a highpass routine with which you will now be familiar gives an estimate of the speed of each wheel.

The steering sensor has an identical pair of "crossed Hall effect" sensors and uses the same subroutine to calculate the angle. The crispness of the steering is made possible by the addition of an analog tacho. This signal is encoded by an additional ADC channel, bringing the total of channels to encode to seven.

13.1.5 Software Structure and Timing

The software framework was designed by Jason Stone, of the NCEA.

After initialization, the software enters an "idle loop" waiting for commands from the two-button switch or the rump switches. Whether recording or playing a dance, the routine also follows a simple loop. All the control is applied within interrupt routines.

A timer interrupt initiates a control cycle every 4 ms, which starts by reading the sensors and updating the odometry. The steering is serviced and a counter selects every tenth interrupt, so that speed control and the keypad routines are serviced every 40 ms.

Every 4 ms the steering angle is measured and a required steering velocity is computed. This velocity demand is limited in magnitude. The measured steering velocity is added and the result controls the bang-bang (with dead-zone) drive to the motor. The deadband is only one "beg" when the cow is moving or when the demanded angle is nonzero, but is increased when at rest to conserve battery life.

The other two important sensor channels are the joystick controls. A conventional model aircraft radio system is used, in which the commanded values are represented by a variable pulsewidth. These pulses are received at times that the software cannot "expect." Their widths must therefore be measured in another interrupt routine, where they are "parked" for processing every 40 ms.

Speed control is applied only every 40 ms, unless there has been a change in command. The velocity is, however, computed every 4 ms.

The lower 2 bytes of the multiturn wheel angle are used as an input to a numerical lowpass filter, with time constant 64 ms, and the difference between actual and filtered values will represent an actual velocity. The quantization level of this velocity is one sixteenth m/s. It is divided by 8 to give a quantized speed zone in the range ±7, where each unit is 0.5 m/s and the top controlled speed is 3 m/s. If the demanded speed is +7 or −7, continuous full power is applied to the motors.

If the measured speed is less than the demanded speed, 12 V of acceleration is applied to the motors. If the speed codes are equal, or if the measured speed exceeds demand by one unit, the motors freewheel. If the speed exceeds demand by two or more counts, braking drive is applied. Speed zero is deemed to belong to the reverse direction set, so that when moving forward and commanded to stop, braking will continue until the zero-speed zone is reached.

13.1.6 In Conclusion

There is some video of Robocow in action on the Web at www.essmech.com/13/1/6. An early prototype was seen on UK television in a *Tomorrow's World* program, while another prototype was placed on display in the Powerhouse Museum in Sydney.

13.2 VISION GUIDANCE FOR TRACTORS

Some years ago, a vision guidance system was developed at the University of Southern Queensland to the stage where commercial exploitation was attempted. Six prototypes were tested by farmers in Australia, and two more were put on trial in the United States. Over the years of the project, there were several changes of imaging technology but the fundamental principles remained consistent. New funding has seen a rebirth of the project, now to be integrated with GPS guidance.

13.2.1 Introduction

The system derives its guidance signal from a videocamera image of the rows of a crop ahead of it, such as cotton. The patented strategy makes it relatively insensitive to additional visual "noise" from weeds, while by tracking several rows at a time it can tolerate the fading out of one or more rows in a barren patch of the field. The image of each row is tested for "quality."

Experimental results showed that the system was capable of maintaining an accuracy of 2 cm. Farmer responses from the extensive field trials were full of enthusiasm—but they still did not purchase the system in sufficient quantities to keep it alive.

The need for automatic guidance of farm vehicles had been recognized for a considerable time. Many guidance methods had been considered, ranging from buried leader cables to beacons, surveying instruments, or satellite navigation. GPS was in its infancy at the time of the original project. All had their drawbacks. The most appealing method was to follow human practice and take guidance from the crop itself, steering the vehicle by means of the view of the rows ahead.

Consistent accuracy of row following allows cultivator blades to be set much closer to the plants, greatly increasing the efficiency of weed control and circumventing the need for additional spraying. Meanwhile, the driver can give greater attention to the cultivation operation and the state of the crop.

But enough of the sales talk. How does it work?

13.2.2 Design Tasks

The design presented a succession of problems:

1. Acquiring an image
2. Determining what pixels represented "plant" and which were soil
3. Separating each row from the others
4. Fitting a line to the center of each row
5. Analyzing slopes and intersections to find a vanishing point
6. Deducing turning of the tractor by using movement of the vanishing point

7. Deducing position error of the tractor from the changes in slopes of the fitted lines
8. Constructing a steering signal

And that is just the vision part of the problem. Then we had the tasks of

9. Adding hydraulic valves to actuate the steering
10. Measuring the steering angle for feedback
11. Closing the steering loop
12. Designing the overall feedback loop that applies the vision-based demand

This last stage is far from trivial. For safety, the steering loop had a slew rate limiter. This was in the form of a simple oil-flow restrictor. In the event of a malfunction the tractor steering could not suddenly slew and cause the tractor to roll over. But the introduction of this nonlinearity brings some severe control problems.

A simulation at www.essmech.com/13/2/2.htm shows that while a small disturbance might be corrected quickly and easily, a larger disturbance can send the same system into oscillation. As we will see later, the control must be designed with nonlinearity in mind.

13.2.3 Image Acquisition

The early work was based on vision systems with very limited capabilities. Far from hampering the project, these limitations almost certainly contributed to its success. It is my opinion that other researchers were led astray by an excess of data and that problems were tackled that did not really relate directly to the fundamental task of steering.

The first of our experiments used a binary "frame grabber" that yielded a black-and-white image—no gray levels—with a resolution of 768 horizontal points by 96 rows vertically.

The image transfer was performed by direct memory access (DMA) to be captured in an array in main memory. Here the software was able to access it for processing. At some cost in overall speed, part of the image was intermittently copied directly to the display memory so that it could be seen on the computer screen and the effectiveness of the algorithm could be assessed by eye. Only a decade before the time of writing computers were much slower.

A later version used a camera interface developed for the consumer market, the "Video Blaster"—marketed in numerous revisions. A full-color image was captured in the onboard memory, and this image could be merged "live" in a window forming part of the display.

The system did have attendant disadvantages, of course. The image memory was mapped at a high address in extended memory, usually selected to be at 15 Mbytes. (That was high in those days!) Occupying 0.75 Mbyte of addressing space, modest computer speeds meant that care had to be taken to select only a small proportion of the data.

With the availability of color, better discrimination was achieved. A field with a newly shooting crop may be littered with light-colored detritus that makes it difficult to discern the crop rows if brightness alone is used. The use of a green filter over the lens provides no improvement. With color, however, it was possible to use the chrominance signal rather than luminance to acquire an image based on the "greenness" of each point.

Today we have a stream of Webcam data with 3 bytes for each pixel, representing red, green, and blue. "Green minus red" is one combination that will give a signal that depends on color, rather than brightness.

Commonality between the evolving hardware versions has been achieved by the use of a function, `picbit(x,y)`, which presents the image in a standard form to the analyzer irrespective of the system from which it is acquired.

13.2.4 Separating Plant from Soil

The level (whether brightness or resolved color component) of the image is now held as a two-dimensional array of values. The first task is to discriminate between the crop and the background field, something made harder by clouds that can change the light levels from moment to moment.

Other researchers had devoted a nine-page paper to this discrimination problem. They argued that the pixel values could be clumped into two separate peaks, corresponding to plant and soil. With the "leafiness" of plants and the lumpiness of soil, this did not seem to have always been the case.

We found a much simpler approach to be successful. From the state of the crop we know roughly the proportion of the ground that is covered. As we pick pixels to analyze, we keep count of the numbers that are reported respectively as plant and as soil. If their ratio is higher than the expected groundcover ratio, we increase the threshold by one step; if it is less, we decrease the threshold. It is as simple as that.

Within a few frames the row images are seen to "fatten up" to match our density expectations. If they do not match the view from the cab window, a tap on a button will change the density parameter until they do. The simplicity of this level adjustment strategy is a heritage of the original binary frame grabber, which made a more complicated strategy unreasonable.

13.2.5 Separating the Row Images

We used a simple technique that depends somewhat on a circular argument. If we know where the rows are, we can define "keyholes" in the image so that the pixels of any keyhole will contain only the image of one row, plus some soil either side of it.

Now our task is reduced to one of finding how the keyhole should be moved to the center of the row within it. Since this involves only calculation, rather than steering movement of the tractor, the correction can be applied to the very next frame to track the rows as they appear to move about.

That still leaves the problem of finding the rows in the first place. But when the tractor is driven straight, we know when to find them. When the quality of fit is insufficient, the windows drift back to the central "straight-ahead" position. Only when it has a good lock on the rows can the system signal that controls automatic steering be engaged.

13.2.6 Fitting Lines to the Rows

The condition of the crop changes through the growing cycle. Initially the plants appear as rows of small dots among other scattered random dots that are weeds. Later they fuse to form a clear solid line. Before long, however, the lines have thickened and threaten to block the laneways. Great tolerance in the vision algorithm is thus required to fulfill all the seasonal requirements.

Figures 13.3a–13.3f are slides from an early presentation, showing how it was done.

The lines tilt either side of vertical, so it is logical to use the form

$$x = ay + b$$

to describe them. The horizontal distance of a point from the line is

$$x - ay - b$$

To fit these to the rows, once again simplicity is the order of the day. It would be a mistake to attempt to analyze the shape of the row boundaries, especially in the early stages of growth. Instead, the "plant" pixels can be treated as points on a graph, through which a straight line is to be fitted.

The regression method is used to fit the best straight line to a set of points. The regression line minimizes a quadratic cost function, the sum of the weights of the points times the squares of their distances from the line. This cost function can be thought of as similar to the moment of inertia of the data points, represented as masses corresponding to pixel values, when spun about the best-fit line.

In our case, we are interested in the horizontal distance from the line, rather than the perpendicular distance, so that the cost function becomes

$$C = \sum_{x,y \in \text{keyhole}} m(x, y)(x - ay - b)^2$$

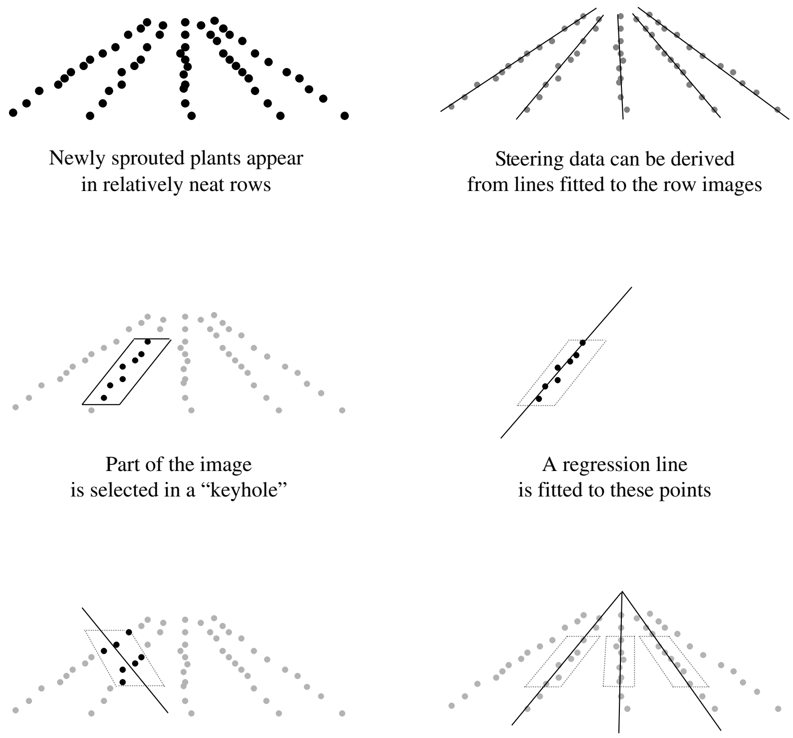
a Newly sprouted plants appear in relatively neat rows
b Steering data can be derived from lines fitted to the row images
c Part of the image is selected in a “keyhole”
d A regression line is fitted to these points
e The moment about this line gives a measure to guard against errors
f Several keyholes are updated to track their regression lines

Figure 13.3 *Slides showing row fitting.*

(This is actually a double summation, since we must sum over both x and y.) Now we want to find the values of a and b that will minimize C. At this combination of values, the partial derivatives of C with respect to a and b will be zero:

$$\frac{\partial C}{\partial a} = 0, \quad \frac{\partial C}{\partial b} = 0$$

When we differentiate, we are still left with the summation giving two simultaneous equations in a and b, involving coefficients that are the following sums:

$$\sum m(x, y), \quad \sum xm(x, y), \quad \sum ym(x, y), \quad \sum xym(xy) \quad \text{and} \quad \sum y^2 m(xy)$$

which in our code we will call

```
m, mx, my, mxy, myy
```

Instead of *a* and *b*, we give the results the more descriptive names `fitx` and `fitslope`, so that the solution is calculated by

```
fitx = (mx * myy - mxy * my) / (m * myy - my * my)
```

and

```
fitslope = (m * mxy - mx * my) / (m * myy - my * my)
```

The results are delivered to the steering process in the form of the lateral movement of the vanishing point and the slope of the rows in the picture. From these we can calculate the lateral displacement at any distance in the rows ahead.

If we also calculate the value `mxx`, we can find the actual minimum value of *C*. If the fit is good, the result should be small. If the crop is scattered or confused with weeds, however, the moment of inertia will be larger. As a test, the minimized value of *C* is divided into `m` times the moment that we would get if every point were plant, to give a measure of `quality`. The steering information is acted on only if `quality` is sufficiently high.

Often a row may fade out halfway down the field. For this reason, the computation is performed not just for one row but for two or for three. (If all rows are found to be unacceptable, 3 times in succession, then an alarm sounds and control reverts to manual.) Finally, the mean value of all the samples in the keyhole is used to adjust the brightness or greenness threshold for the next frame.

13.2.7 Applying Steering

The main steering tasks were outlined above in items 9–12 in the list in Section 13.2.2. First we must provide a way of converting the electrical signal into the mechanical steering action. One approach would be to turn the steering wheel by means of a motor, but instead we decided to exploit the hydraulic steering of the Case Magnum tractor that had been lent to us.

Two solenoids that drove a relatively simple valve gave us an action in which the steering could be controlled to slew to the left or right. Constrictors, disks with small holes in them, were added to the valve to limit the rate of slew. If anything went wrong with the electronics or electrics, it was essential that the action not be too violent. The human driver must be able to counteract any such steering action.

Now we needed a measurement of the steering angle, to use as feedback. For this, the Hall effect sensor was ideal; it is mentioned in Section 2.2.1 and described in more detail in Section 3.5.2. We can now calculate the angle

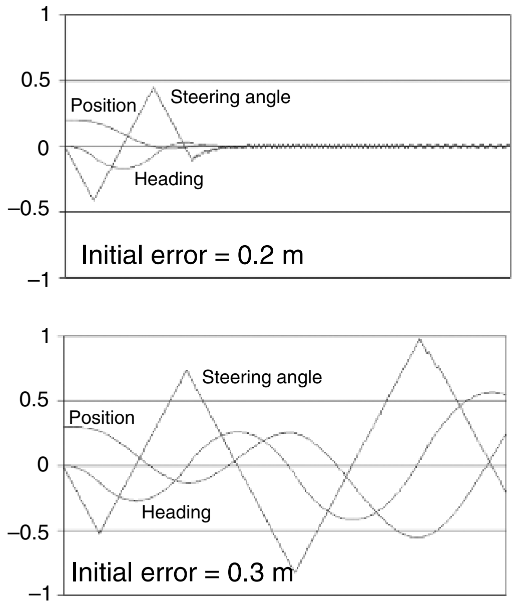

Figure 13.4 *Steering simulation.*

error and simply set one or other solenoid according to the sign of the error, leaving a small deadband in between. The rate limit, although essential for safety, does have a control drawback.

The error signal taken from the vision system is the apparent lateral shift of the rows partway up the picture. Since it is measured ahead of the vehicle, this will have a value that is a sum of the vehicle displacement and a term proportional to heading relative to the row.

In the strategy of a simple linear controller, the demanded steering angle would be made to be proportional to this error. When the rate of change of the steering angle is limited, however, an abrupt onset of limit cycle instability can occur if the initial error exceeds a relatively modest value. This is portrayed in the simulation at http://www.essmech.com/13/2/7.htm and shown in Figure 13.4. The result is to be noted more for its qualitative effect than as an exact prediction of the error magnitude at which instability will break out. Here it shows a nearly ideal response from an error of 0.2 m, while an error of 0.3 m results in disaster.

Of course, the magnitude of the disturbance at which instability occurs can be increased by reducing the steering gain or by choosing a point further ahead from which to take steering data. In either case, however, performance is lost and the response time for recovery from a disturbance is increased.

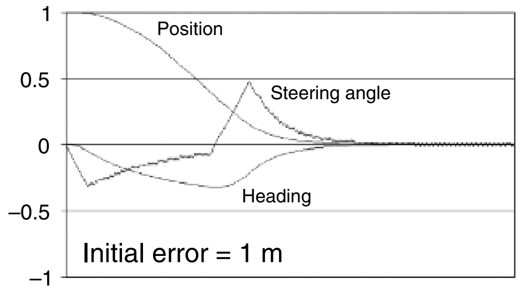

Figure 13.5 *Steering with limited heading demand.*

By rearranging the algorithm in accordance with the topological, nested-loops approach outlined in Section 10.1, we can calculate a succession of demands to which we can apply limits. The simulation mentioned above has been arranged in this form, but the limits have been set too loosely to have any effect. We see a system that is linear apart from the steering rate limitation. You should experiment with the simulation to try various values of limits on steering and heading angles.

Imposing a limit on the heading angle, the amount by which the vanishing point is seen to move, has a dramatic effect as shown in Figure 13.5. Here the limit is set at 50 pixels and an error of one meter is seen to settle with no problem.

Figure 13.5 shows the advantage of the row-fitting approach, identifying the vanishing point movement, over the simpler strategy of inspecting a single horizontal line of the image to measure a displacement.

13.3 A SHAPE RECOGNITION EXAMPLE

As part of a doctorate project, Mark Dunn is working on the discrimination of animal species. Many different animals will approach a watering point in the Australian outback, including the sheep and cattle for which it is intended, kangaroos and other native animals, and also feral species that are regarded as pests.

Feral pigs do untold damage, but feral pork is a commodity that has commercial value. As the animals move past a recognition system, a gate moves to one of two positions, giving access to one of two enclosures. In one of them, animals can reach water and after drinking can exit and go on their way. In the other, feral pigs will gather, drink, and be held for the later attention of the farmer.

The image analysis can be made much easier if the approach has a blue background, such as a tarpaulin, but this might deter the animals from

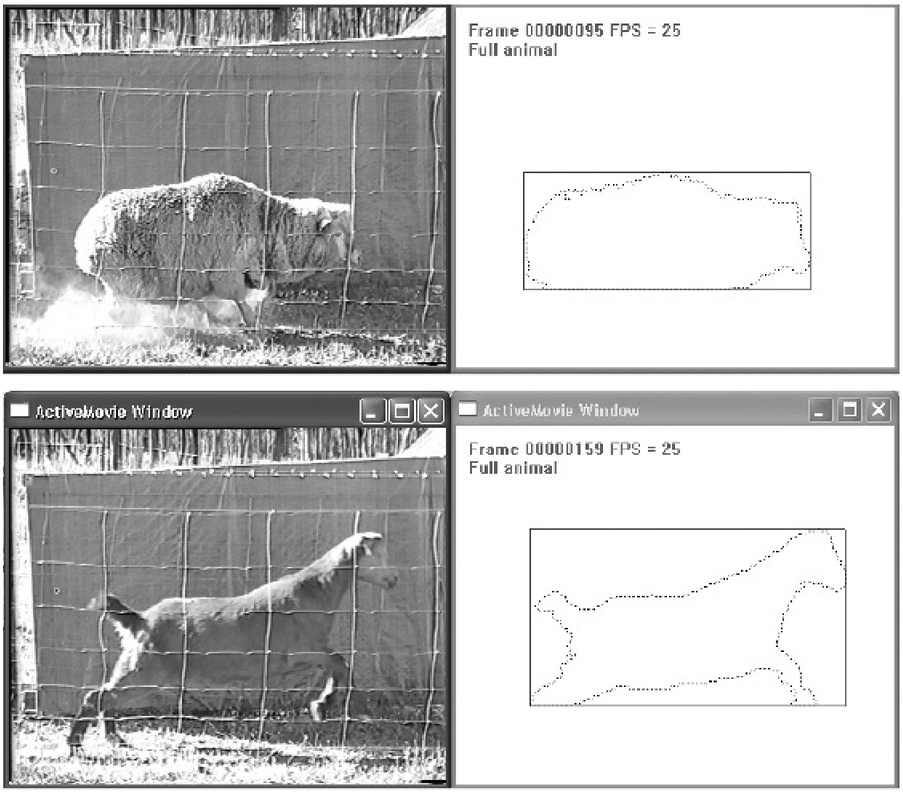

Figure 13.6 *Outlines of sheep and goat.*

approaching. An alternative is to look for changes in the background image, but this can give problems with animals moving in the background. Finding a workable compromise is in the nature of research. Image comparisons using sheep and goats are presented in Figures 13.6 and 13.7.

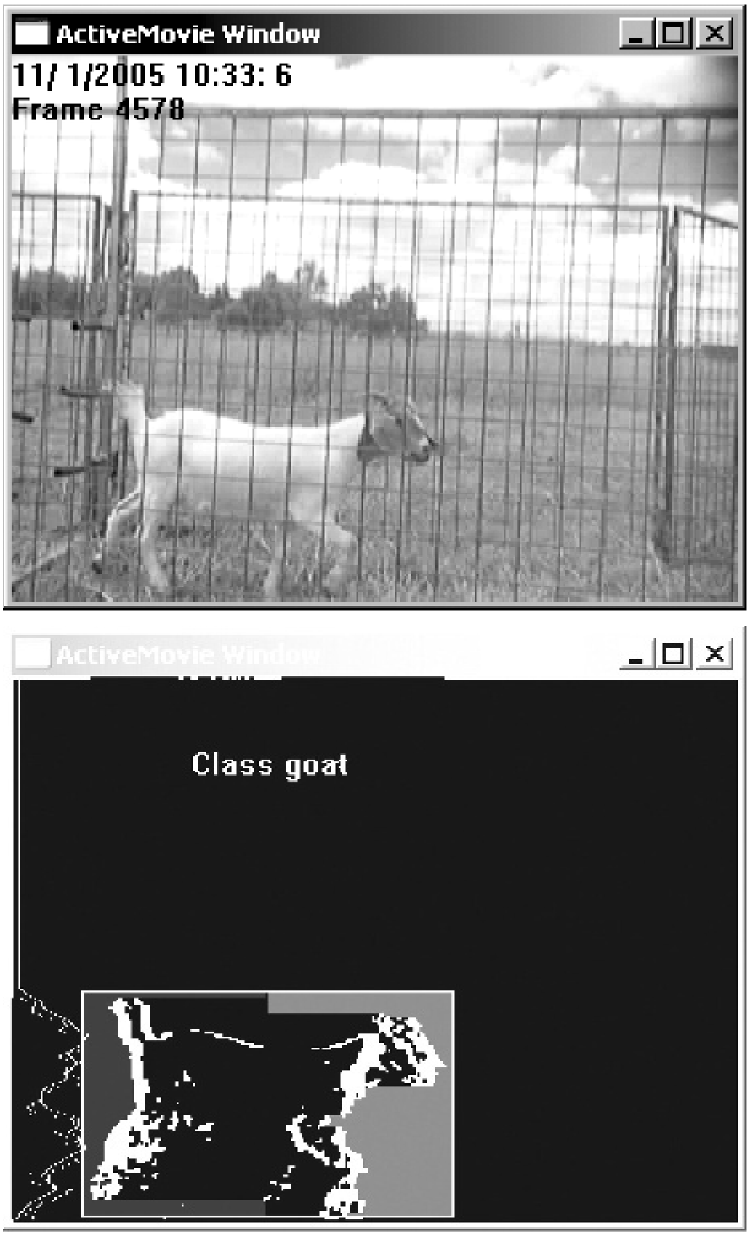

Figure 13.7 *Classification with natural background.*