[30] Wallace, R. “Robot Road Following by Adaptive Color Classification and Shape Tracking”.

[19] Ollis, Mark & Stentz, Anthony. First Results in Crop Line Tracking. Proceedings of IEEE Conference on Robotics and Automation (ICRA ‘96), Minneapolis, MN April1996, pp.951-956.

[20] Pomerleau, D. A. Neural Network Perception for Mobile Robot Guidance. Kluwer Academic Publishing, Boston MA, 1994.

[21] Pomerleau, Dean. RALPH: Rapidly Adapting Lateral Position Handler. Proceedings of the 1995 IEEE Symposium on Intelligent Vehicles, Detroit, Michigan.

[22] Pomerleau, Dean. private correspondence, April 3, 1995.

[23] Reid, J.F. and Searcy, S.W. An Algorithm for Separating Guidance Information from Row Crop Images. Transactions of the ASAE. Nov/Dec 1988 v 31 (6) pp. 1624-1632.

[24] Reid, J. F. and Searcy, S.W. Detecting Crop Rows Using the Hough Transform. Presented at the 1986 summer meeting of the ASAE, San Luis Obispo, June 1986.

[25] Reid, J.F. and Searcy, S.W. Vision-Based Guidance of an Agricultural Tractor. IEEE Control Systems Magazine. April1987 pp. 39-43.

[26] Rosenblatt, Julio. DAMN: A Distributed Architecture for Mobile Navigation. Ph.D. thesis, Carnegie Mellon Robotics Institute, 1996.

[27] Tsai, Roger. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, August 1987, pp. 323-344.

[28] U.S. Dept. of Commerce. 1994 Statistical Abstract of the United States, 114th edition.

[29] Wallace, R. et al. “First Results in Robot Road-Following”, IJCAI-85, August 1985.

[9] Hayashi, M. and Fujii, Y. Automatic Lawn Mower Guidance Using a Vision System. Proceedings of the USA-Japan Symposium on Flexible Automation, New York, NY July 1988.

[10] Hoffman, Regis et. al. Demeter: An Autonomous Alfalfa Harvesting System. ASAE paper no. 963005.

[11] Healey, Glenn. Segmenting Images Using Normalized Color. IEEE Transactions on Systems, Man and Cybernetics, 1992.

[12] Jahns, Gerhard. Automatic Guidance in Agriculture: A Review. ASAE paper NCR 83-404, St. Joseph. MI (1983).

[13] Jochem, T and Pomerleau, D.“No Hands Across America Home Page”, http://www.cs.cmu.edu/afs/cs/usr/tjochem/www/nhaa/nhaa_home_page.html.

[14] Klassen et. al. Agricultural Vehicle Guidance Sensor. Paper 93-1008 Presented at the 1993 Internation Summer Meeting of the ASAE, Spokane, WA June 1993.

[15] Kluge, Karl. YARF: An Open-Ended Framework for Robot Road Following. Technical Report CMU-CS-93-104, Carnegie Mellon University, 1993,

[16] Mitchell, Tom. Machine Learning. Mcgraw-Hill Series in Computer Science. WCB/McGraw Hill, 1997.

[17] Nilsson, Nils. A Mobile Automaton; An Application of Artificial Intelligence Techniques. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) 1969.

[18] Novak, C.L. and Shafer, S.A. Color Vision. Encyclopedia of Artificial Intelligence, 1992, J. Wiley and Sons, pp 192-202.

8. References

[1] Billinglsey, J. and Schoenfisch, M. Vision-Guidance of Agricultural Vehicles. Autonomous Robots, Vol. 2, No. 1, 1995, pp. 65-76.

[2] Brandon, J. Robert and Searcy, Stephen W. Vision Assisted Tractor Guidance for Agricultural Vehicles. Transactions of the Society of Automotive Engineers, 1993, paper 921650.

[3] Cao, Zuo Lang et. al. Region Filling Operations with Random Obstacle Avoidance for Mobile Robots. Journal of Robotic Systems, 5 (2), pp. 87-102 (1988).

[4] Chateau, T. et al. An Original Correlation and Data Fusion Based Approach to Detect a Reap Limit into a Gray Level Image. Proceedings of the 1997 International Conference on Intellignet Robots and Systems (IROS ‘97), pp. 1258-1263.

[5] Crisman, J.D. Color Vision for the Detection of Unstructured Roads and Intersections. Ph.D. thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1990.

[6] Duda, Richard and Hart, Peter. Pattern Classification and Scene Analysis. 1973, J. Wiley & Sons, pp. 114-118.

[7] Fehr, Bernard and Gerrish, John. Vision-Guided Off-Road Vehicle. Paper 89-7516 Presented at the 1989 Winter meeting of the ASAE, New Orleans, LA, December 1989.

[8] Gerrish, John et. al. Path-finding by Image Processing in Agricultural Field Operations. Transactions of the Society of Automotive Engineers, 1987, paper 861455.

principal advantage of this formulation is the opportunity to exploit the extensive research scrutiny to which Markov processes have been subjected.

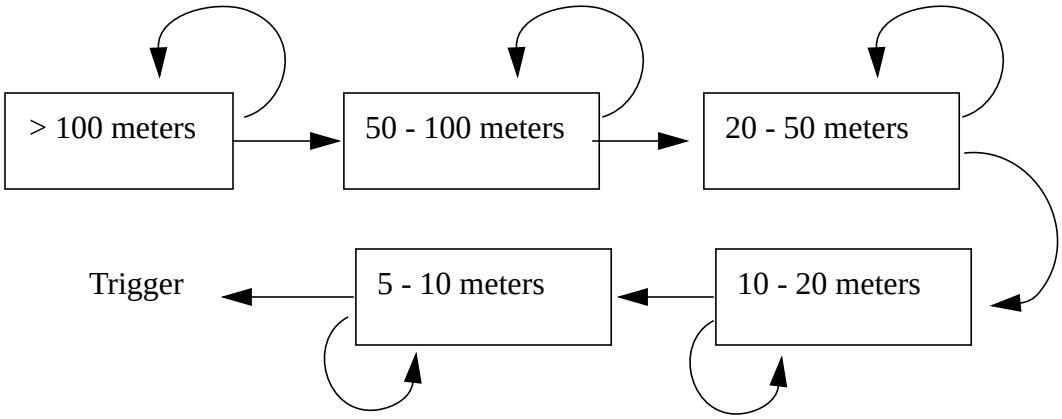


Figure 7-1: A simple Markov chain for end-of-row detection.

7. Future Work

Our boundary tracking results suggest several areas of subsequent research. A careful investigation of sensor types is perhaps the most promising for improving reliability; as discussed in Chapter 3, replacing the red, green, and blue filters of a commercial RGB camera with something optimized for alfalfa might make the problem much easier. There are a wealth of learning techniques which could be investigated for constructing the probability density functions used by the obstacle detection algorithm and the Bayesian boundary detection method. Clearly there is also promise in investigating other similar surface covering problems, particularly on other agricultural tasks with large economic potential (plowing, depositing lime, etc.)

Work on shadow compensation could take a couple of different directions. One idea is to develop a more sophisticated lighting model which is better able to describe the lighting near shadow edges, perhaps by modeling interreflections from sunlit to shadowed regions. It may, however, simply be too difficult to accurately model these effects in real time. An alternate approach is to actually control the physical lighting so that shadows are avoided. At first glance, this seems an impossibility, since the artificial light source would have to dominate the effects of natural lighting and since the sun deposits a huge amount of power on average (amount?) on a square meter of the earth's surface. Using a strobed light source synchronized to a shuttered camera, however, it might be possible to emit a brief high energy pulse during the actual fraction of a second in which the picture is taken; If the shutter is only open .1% of the time, this effectively cuts down on the necessary illumination power by a factor of 1000 over a non-shuttered setup.

It might be interesting to view the end-of-row detection problem as a Markov state estimation problem; that is, at the beginning of a row, the machine is known to be in the first state of Figure 7-1, and in each successive image the perceived end-of-row could be used to calculate a new distribution of probabilities among the states. An end-of-row trigger would not actually occur until the probability of reaching the final state became sufficiently high. The

such influence on the researchers, they will surely only become more critical for a product designed to be used by laymen. Along with the unexpected success of the simple crop line tracker compared to its more complex counterparts, our experience seems to confirm an age-old maxim of engineers: keep it simple, stupid.

The conclusions above are part of the contribution of this thesis. The most significant contribution, however, is undoubtably the development of a commercially viable guidance system for the Demeter harvester; as of this writing, over 80 acres of alfalfa have been harvested autonomously from fields in Kansas, California, and Pennsylvania. New Holland has plans to develop the vision system into a commercially available product for sale in the year 2000. We believe that this commercialization of mobile robotics technology, along with the accompanying publicity, will spur additional commercial investment into the field and help mobile robots make their way out of research labs and into the real world.

between the two cameras, especially given that lighting has proven to be an issue of such importance for this work.

We now move on to more mixed results. The end-of-row detection method, while partially successful, has not proven as reliable as our boundary tracker. One reason for this is simply the extreme robustness that is required of this capability; a single missed end-of-row, or a single false positive, is sufficient to throw the machine well off course. Occasional bad output from the boundary tracker, on the other hand, can be easily filtered out.

Our end-of-row detection method is also less elegant than the boundary tracker. The need to manually set parameters which control how different the out-of-bounds region looks compared to the in-bounds region is surely responsible for some of the difficulty. An attempt might be made to adaptively control these parameters, but training data for end-of-row images comes in much less often than the training data for the boundary tracking discriminant, for which thousands of points can be collected form every image. While we propose a more theoretically rigorous method for end-of-row detection in Section 7, this problem still seems plagued with the need for tuning parameters.

We've also had mixed results with our shadow compensation method. Our simple two-source illumination model of sunlight and skylight has proven useful for large, solid shadows such as those cast by the harvester; but the model seems to break down near the shadow edges, which causes failures in small patchy shadows or long narrow ones. A more complex algorithm would require making assumptions either about the inherent color of the environment or about the actual shadow locations. As we will discuss in Section 7, however, we believe that some active illumination methods hold promise.

Though control is not the focus of this thesis, we offer a few comments on the comparison of image-based control versus the calibrated, pure-pursuit method. Though the pure pursuit model is more theoretically correct and we successfully designed both the target and software required for the necessary camera calibration, the success of the simple image-based method combined with the hassles of camera calibration led to the abandonment of pure pursuit at a relatively early stage. This is an important point, for if convenience issues exert

6. Conclusions & Contributions

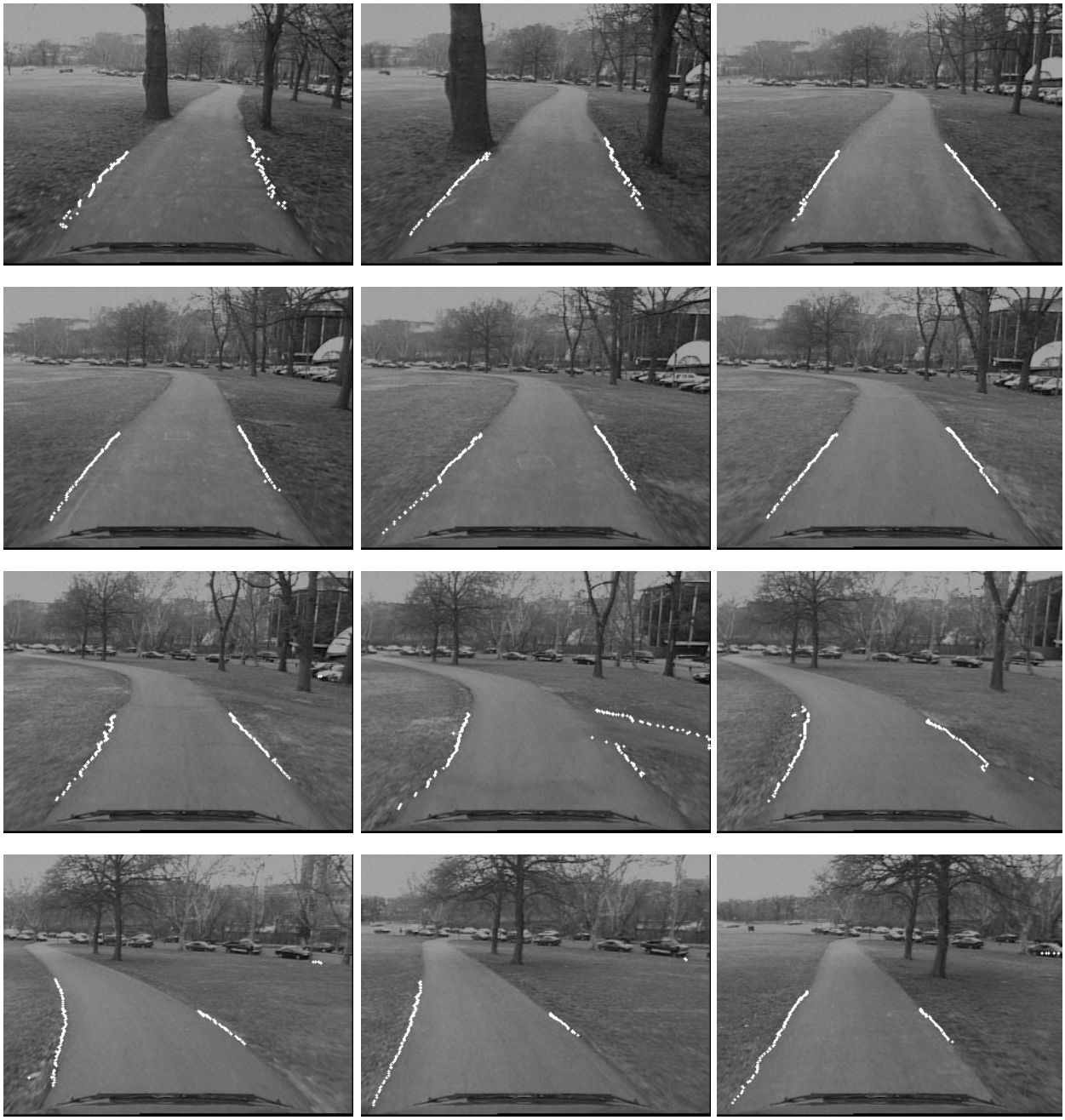
Our main conclusion is that surface covering applications can be usefully viewed as a class of applications with common needs, and that the common domain knowledge of these tasks can be exploited for towards the development of useful perception algorithms. This claim is supported by the use of our boundary-following algorithms in guiding an actual harvester and also by the preliminary results from the snow plowing and trash compaction applications using non-color based discriminant functions.

Each of the three boundary following methods discussed have some advantageous characteristics. The lack of tuning parameters, simplicity, flexibility of its output model, and rapid running time of the step-function based method have made it our method of choice for alfalfa harvesting. In another application in which the distributions of the processed and unprocessed areas are less unimodal, it is likely that the Bayesian method would be more appropriate; and in arenas in which the boundaries change slowly or can be regulated to simple curves, the potential computational complexity of the curvature postulation method may be less important than its ability to take advantage of multiple environmental cues.

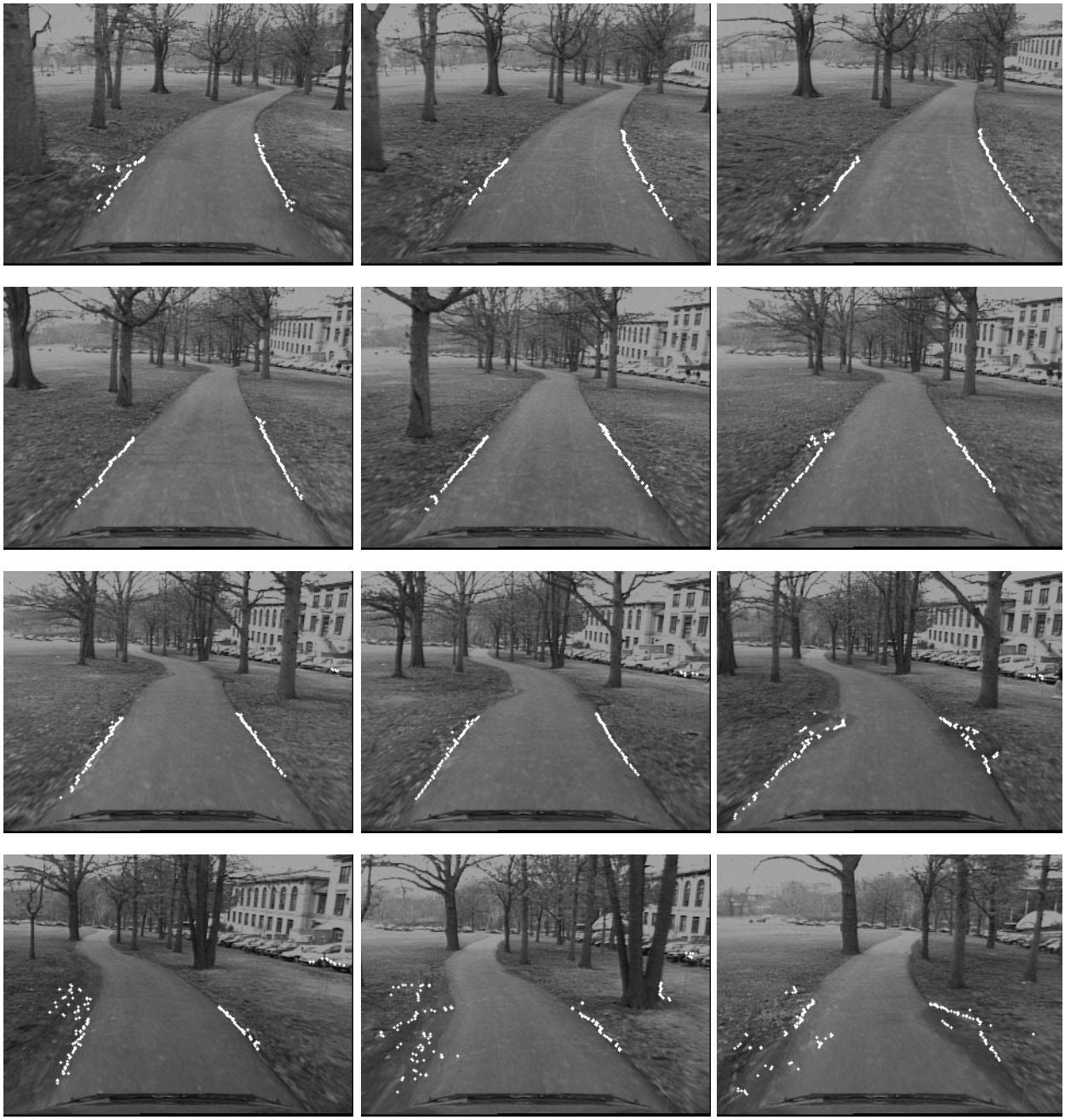
From the experiments with adaptive algorithms we come to several conclusions. First, in at least some applications, there is a significant advantage in being able to adapt to new conditions in merely a few seconds time; as our alfalfa image sequence from Pennsylvania shows, field conditions can change very rapidly from one part of a field to another. Second, the remarkable insensitivity of all of the adaptive algorithms to misclassified training data is evidence that this robustness feature is a function of the actual geometry of surface covering perception. In essence, the same simple geometry that allows for the development of very fast boundary following algorithms also helps stabilize the learning algorithms. This observation tips the scales heavily in favor of collecting training data and doing boundary processing all from the same camera; the use of a separate training camera offers the promise of collecting training data with fewer misclassifications, but any benefits accrued here would likely be outweighed by the additional issues brought on from lighting differences

Road-follwing images (cont)

Road-follwing images (cont)

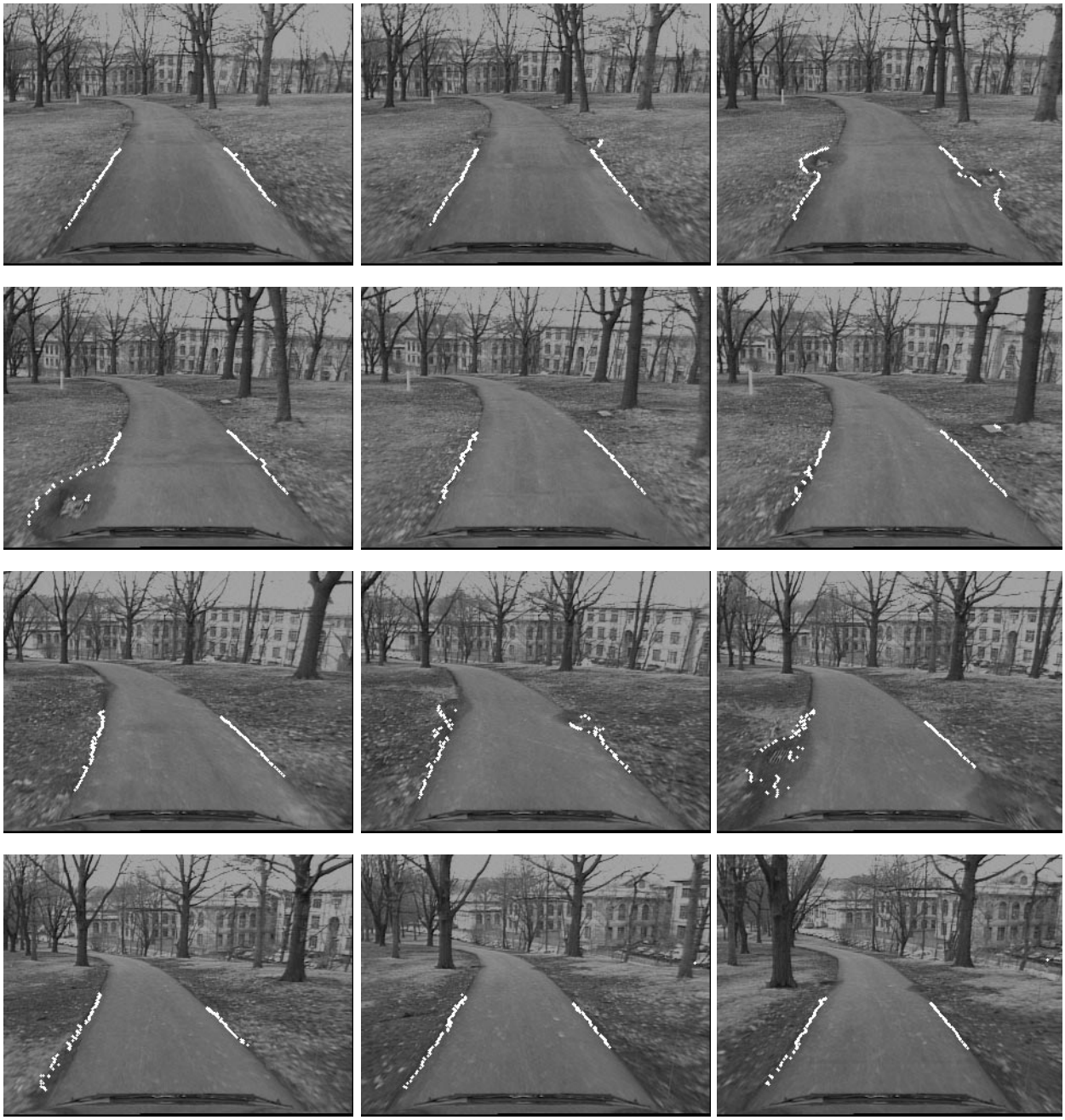


Road-follwing images (cont)



Road-follwing images (cont)

Figure 5-8: Road-following images.



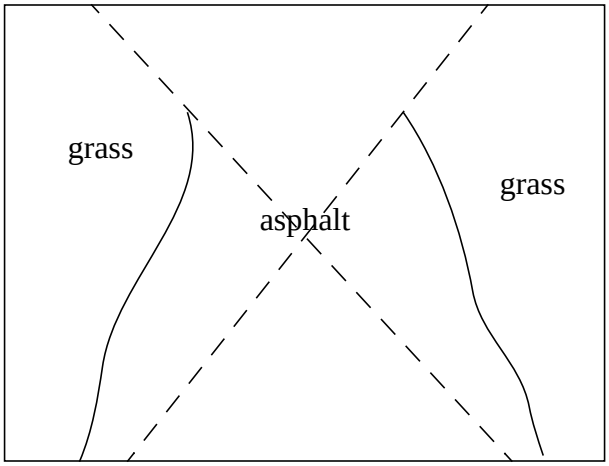


Figure 5-7: image model for road-following task.

discriminant d(r, theta) for selected values of theta to determine the r which best segmented the image. Values of theta near the vertical were not processed, giving an image model as shown in Figure 5-7. As with the alfalfa images, a Fisher discriminant in color space was used; results of this method on a set of road images from Schenly Hill are shown in Figure 5-8. Given the immense amount of research currently dedicated to road following, we hesitate to make any claims about the use of this algorithm for this application; these figures do, however, help demonstrate the flexibility of the ideas in this thesis.

apply to additional applications. In particular, any perception problem that can be transformed into the forms shown in Figure 5-4 can be addressed with our methods.

5.3.1 Radial transform

Consider, for example, a polar coordinate transform, in which the variables change from i and j to r and theta. This transform can be used to solve images of the form shown in Figures 5-5 and 5-6; Figure 5-5 shows a boundary for which theta is a single-valued function of r, and Figure 5-6 shows a boundary for which r is a single-valued function of theta.

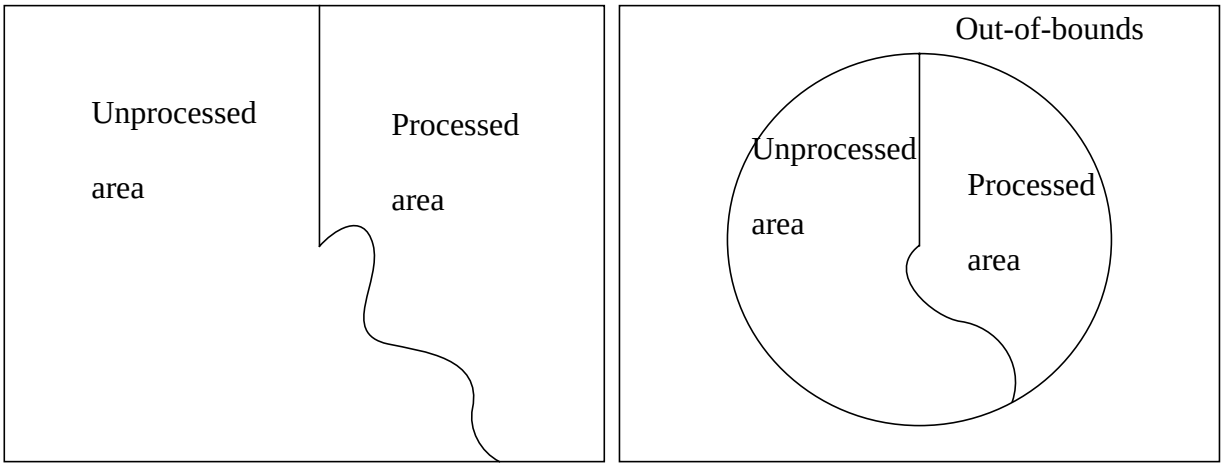


Figure 5-5: Radial transform model: theta single-valued function of r

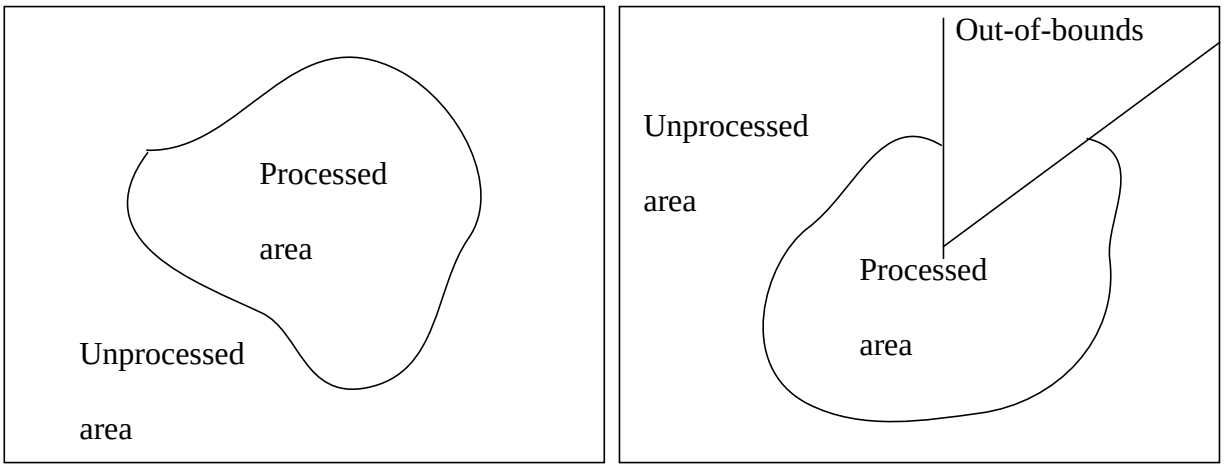


Figure 5-6: Radial transform model: r single-valued function of theta

As an exercise, we used a variant of to segment the edges of an asphalt path from the surrounding grass. Assuming r as a single-valued function of theta, we fit step functions to the

plot of the optimally weighted discriminant strength for the left image of Figure 5-2. However, when the step-function algorithm is applied and the results are smoothed, the boundary can still be picked out reasonably well.

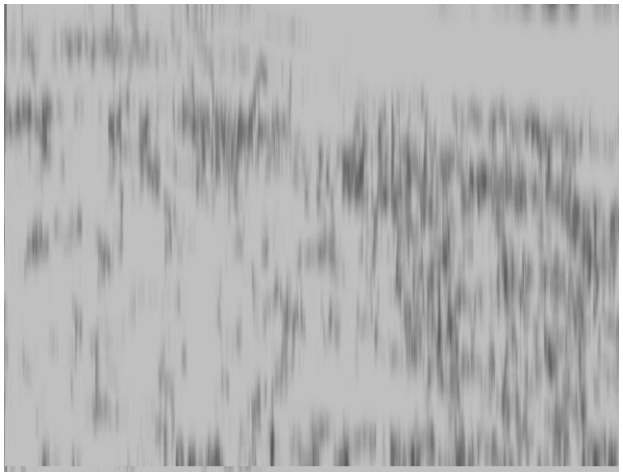


Figure 5-3: Texture discriminant for a trash compaction image.

5.3 Road Following (by geometric transformation)

All of the algorithms in this thesis are based on the simple geometric models described in Section 2.3; that is, that the images to be processed have the form shown in Figure 5-4.

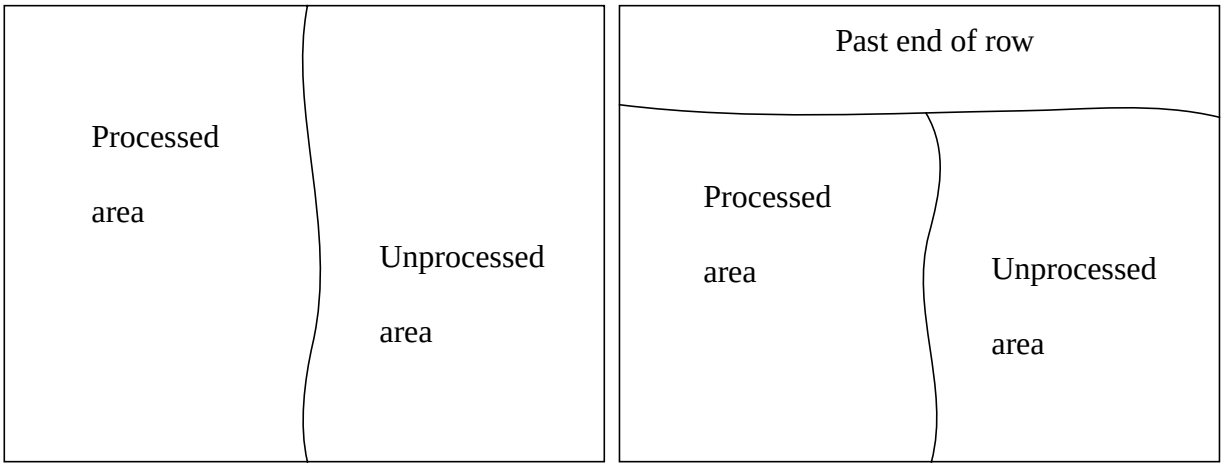


Figure 5-4: Assumed image models.

These simple models are the key to allowing the perception algorithms to run in real time, and also play a crucial role in the stability of the adaptive learning. In this section, we explore how simple geometric transforms of these image types enable our algorithms to

is the plow line between the already-cleared snow and the snow which has not yet been cleared.

We took some images from partially-plowed parking lots in Pittsburgh, and tested the step-function-based crop line tracker on them. Instead of using an adaptive discriminant in color space, we ran the algorithm on black and white images and used a fixed discriminant of pixel intensity, assuming that the plowed areas would be darker, on average, than the unplowed areas. No other modifications to the algorithm were made; results are shown in Figure 5-1.

5.2 Trash compaction

Figure 5-2: Some boundaries between compacted and uncompacted trash.

Figure 5-2 shows some images taken from a large county landfill near Pittsburgh. The landfill employs large compacting vehicles (visible in the background of the images) whose task is to compact all trash in an area so that it can be covered in dirt and readied for more dumping. This compacting task can also be viewed as a surface covering task; in this application, the boundary of interest divides the compacted from the uncompacted trash.

It is clear from inspection that a color-based discriminant would perform poorly on images from this site. Instead, we formulated a set of texture-based discriminants which measure the variation in color in the image across scales of 5, 10, and 25 pixels. Results from our optimized texture discriminant are quite noisy; for example, Figure 5-3 shows a grayscale

5. Results from other applications

The Demeter harvester was the primary testbed for the ideas described in this research. In order to validate the generality of our methods, however, we investigated the some other surface covering applications as well; in this section we present those results.

Ideally, we would have retrofitted and controlled several different surface covering vehicles using the perception algorithms described in Section 3 for guidance, just as was done with the Demeter harvester. The expense and time required to automate additional applications ruled out such a course of action. We have, however, tried processing data from a few other applications; the first is clearing snow from a parking lot, and the second is compacting trash at a landfill.

These two applications were chosen because the images they present are quite different from those of the alfalfa field. While we could have chosen additional agricultural applications with more economic significance, these two choices present an opportunity to test very different discriminants than the color-based ones discussed in the previous chapters.

5.1 Snow plowing

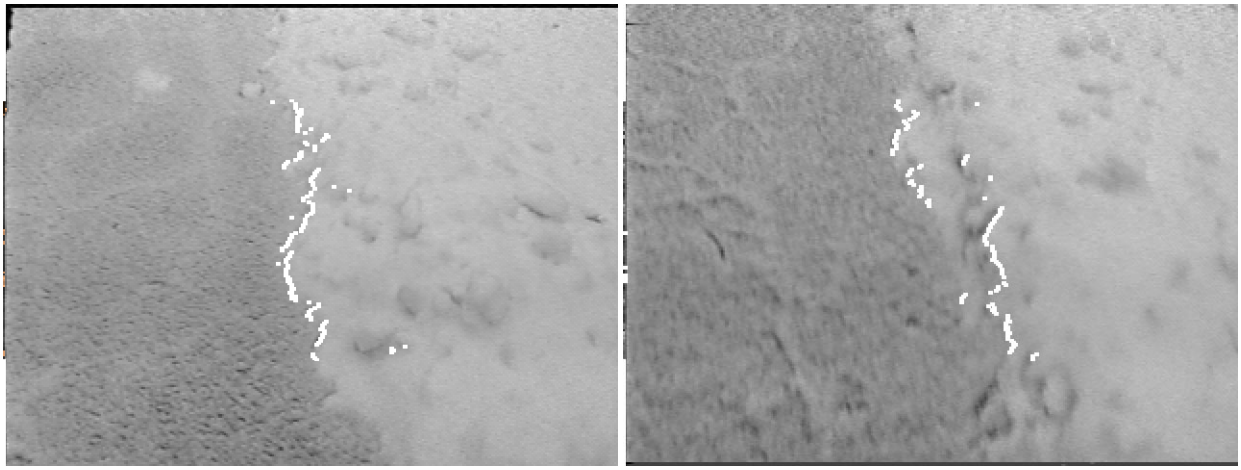
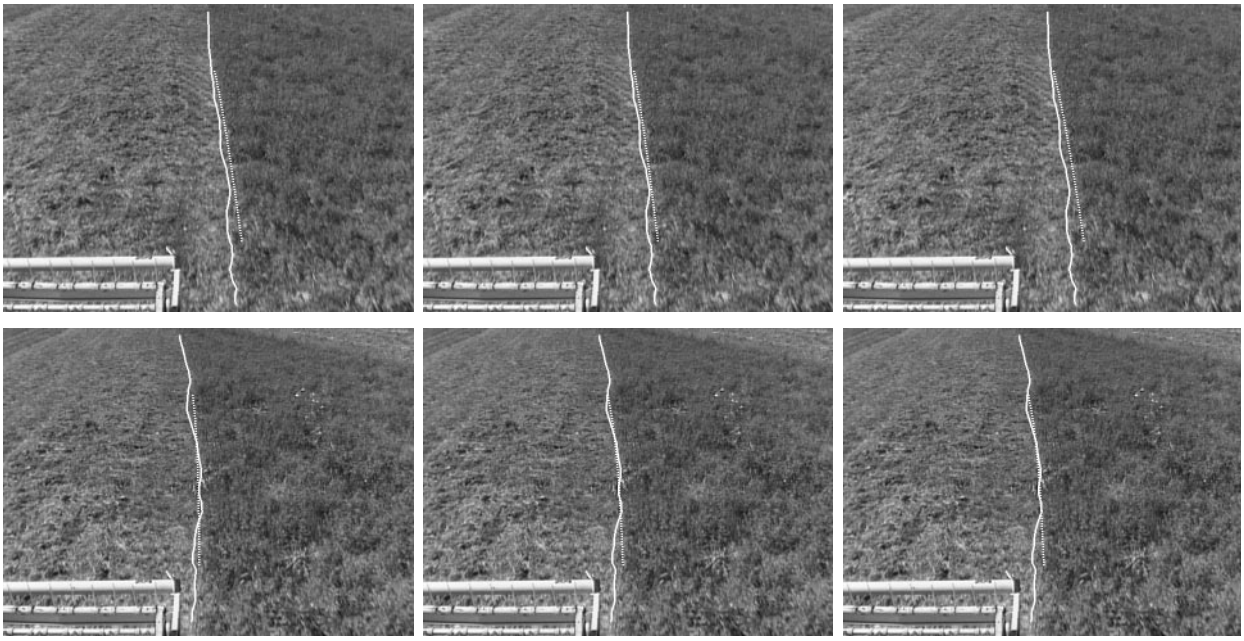


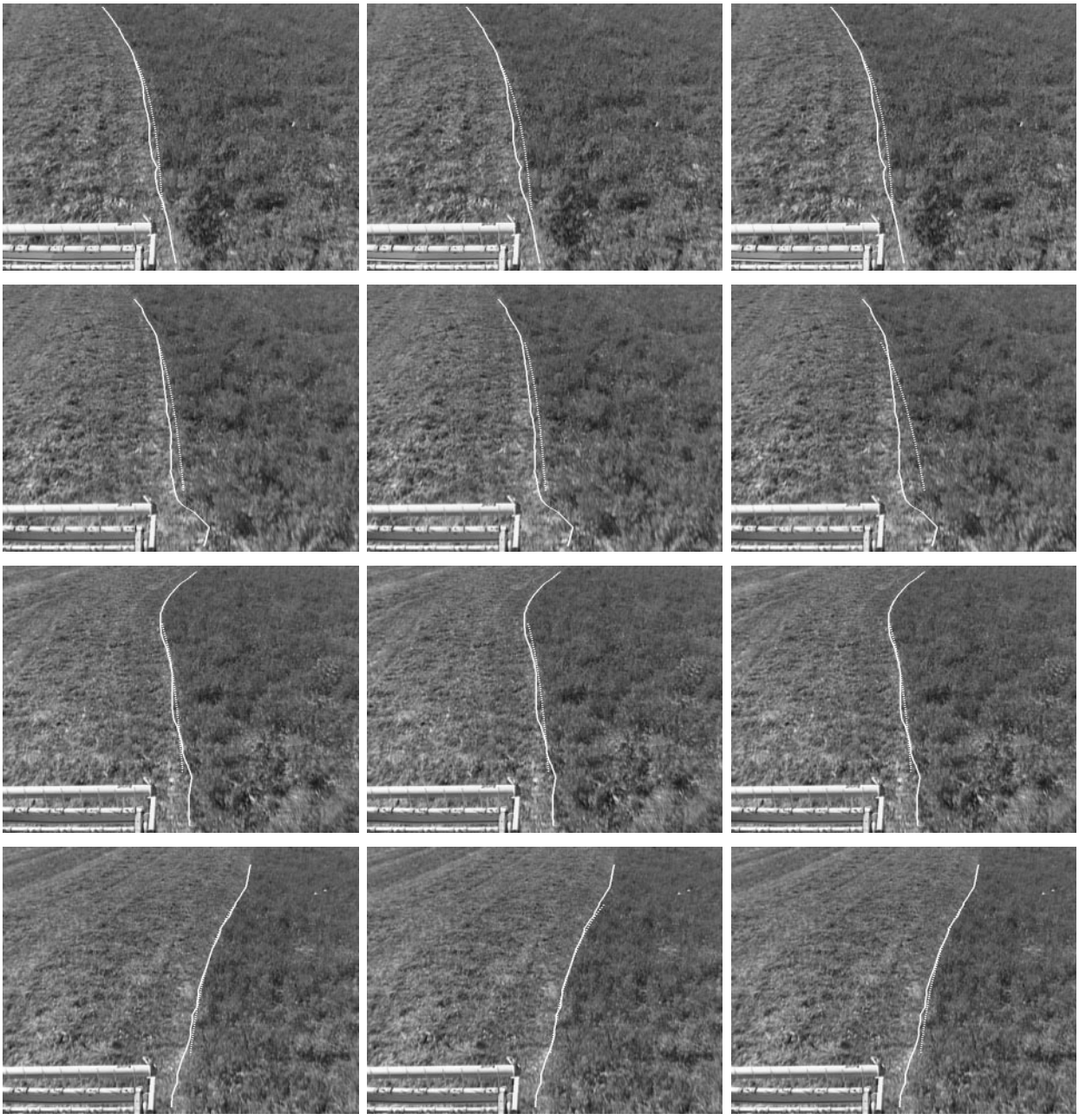
Figure 5-1: Boundary between plowed and unplowed snow.

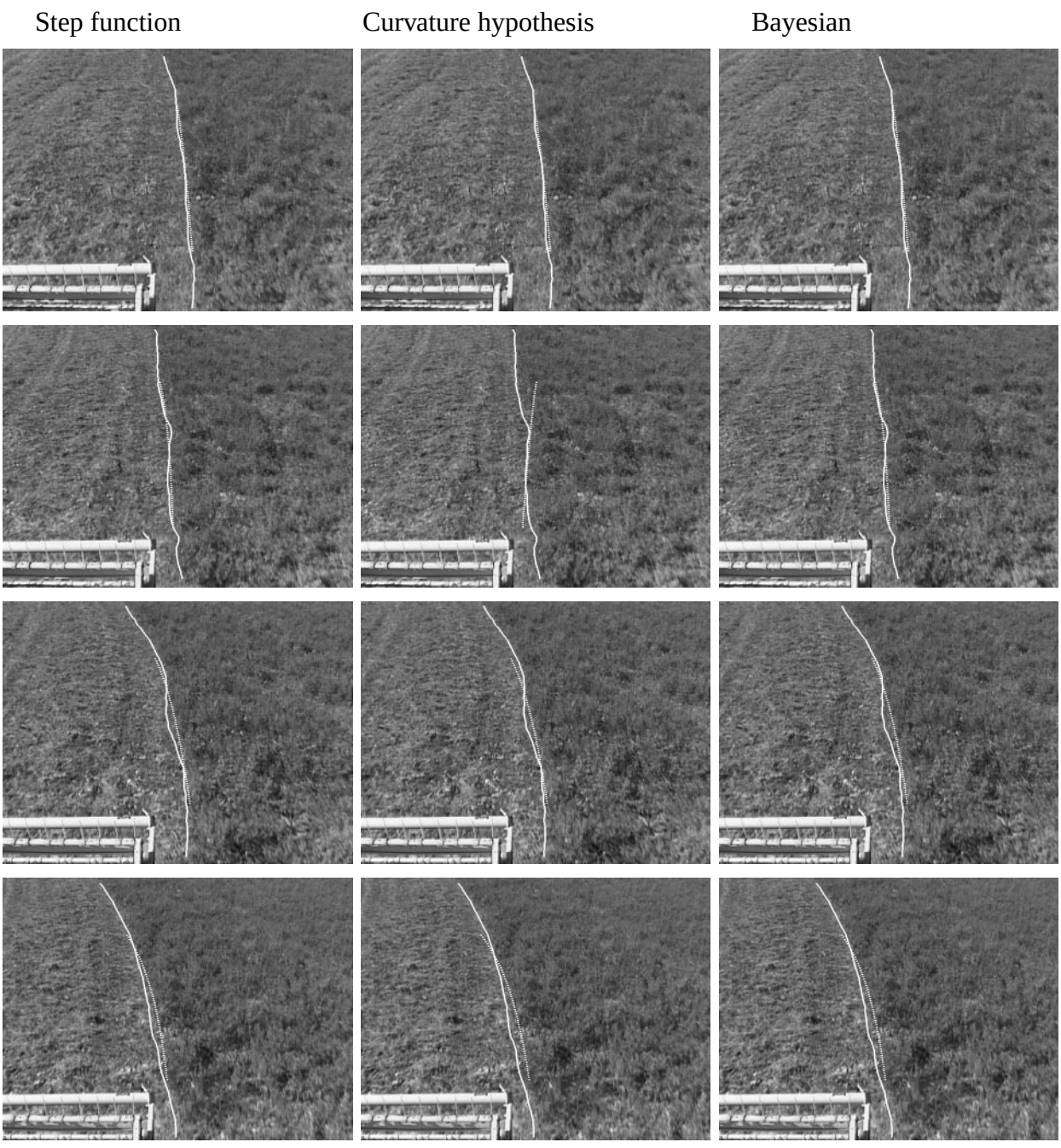
Removing snow from a parking lot or an airport runway is another example of a surface-covering application that could benefit from automation. Here, the boundary to be detected

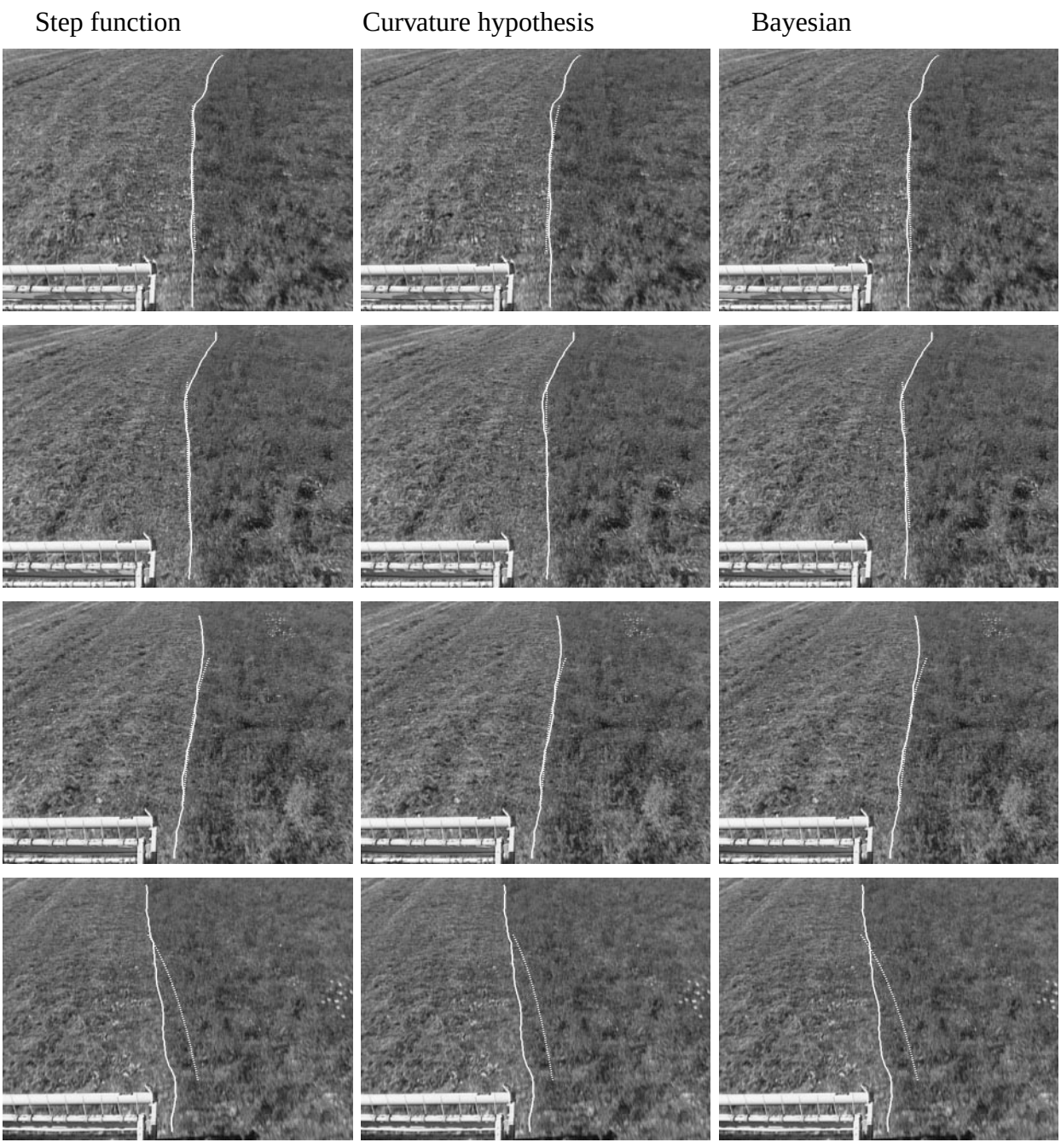
Step function Curvature hypothesis Bayesian

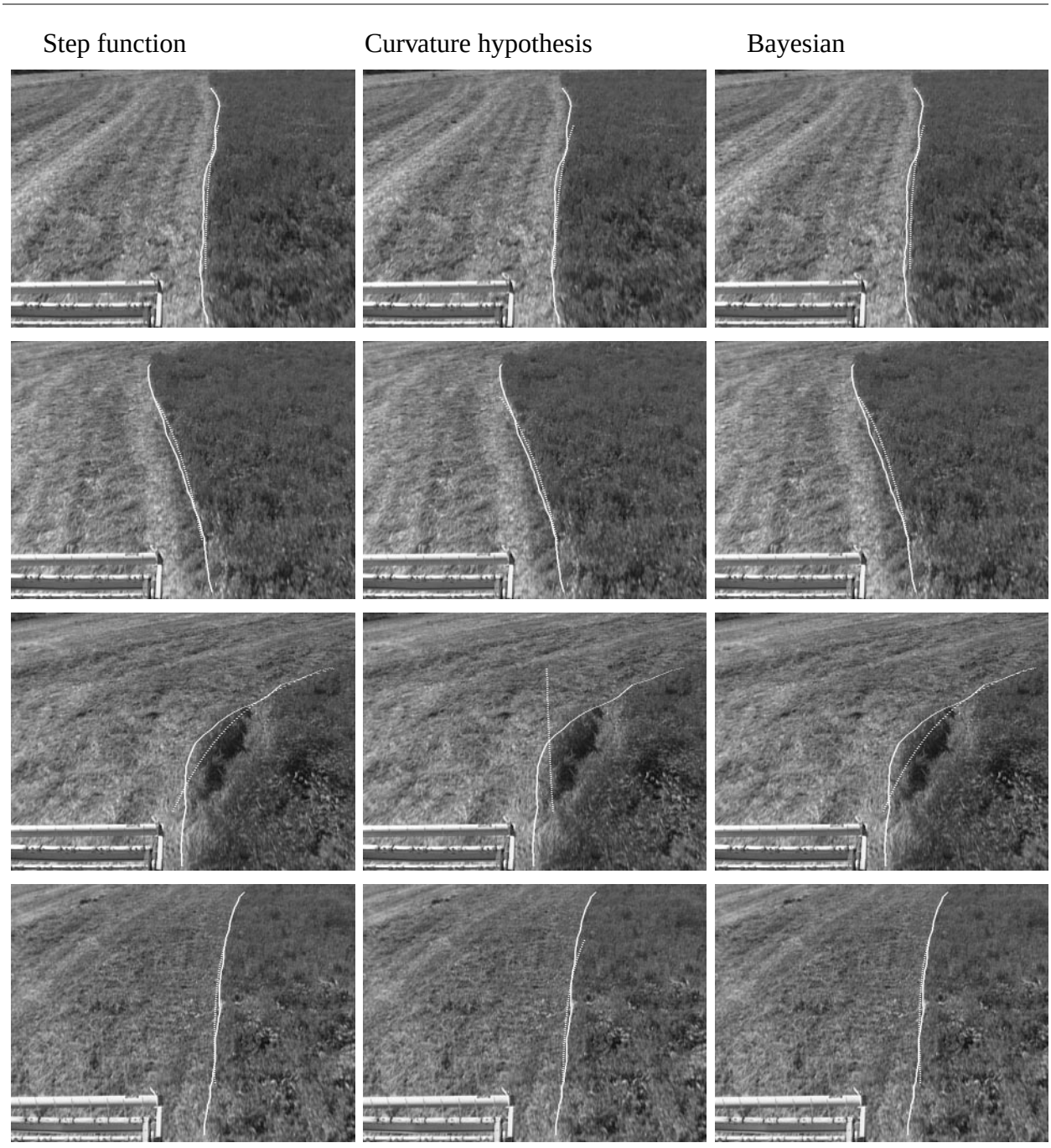


Step function Curvature hypothesis Bayesian




Step function
Curvature hypothesis
Bayesian


Step function
Curvature hypothesis
Bayesian


Step function
Curvature hypothesis
Bayesian

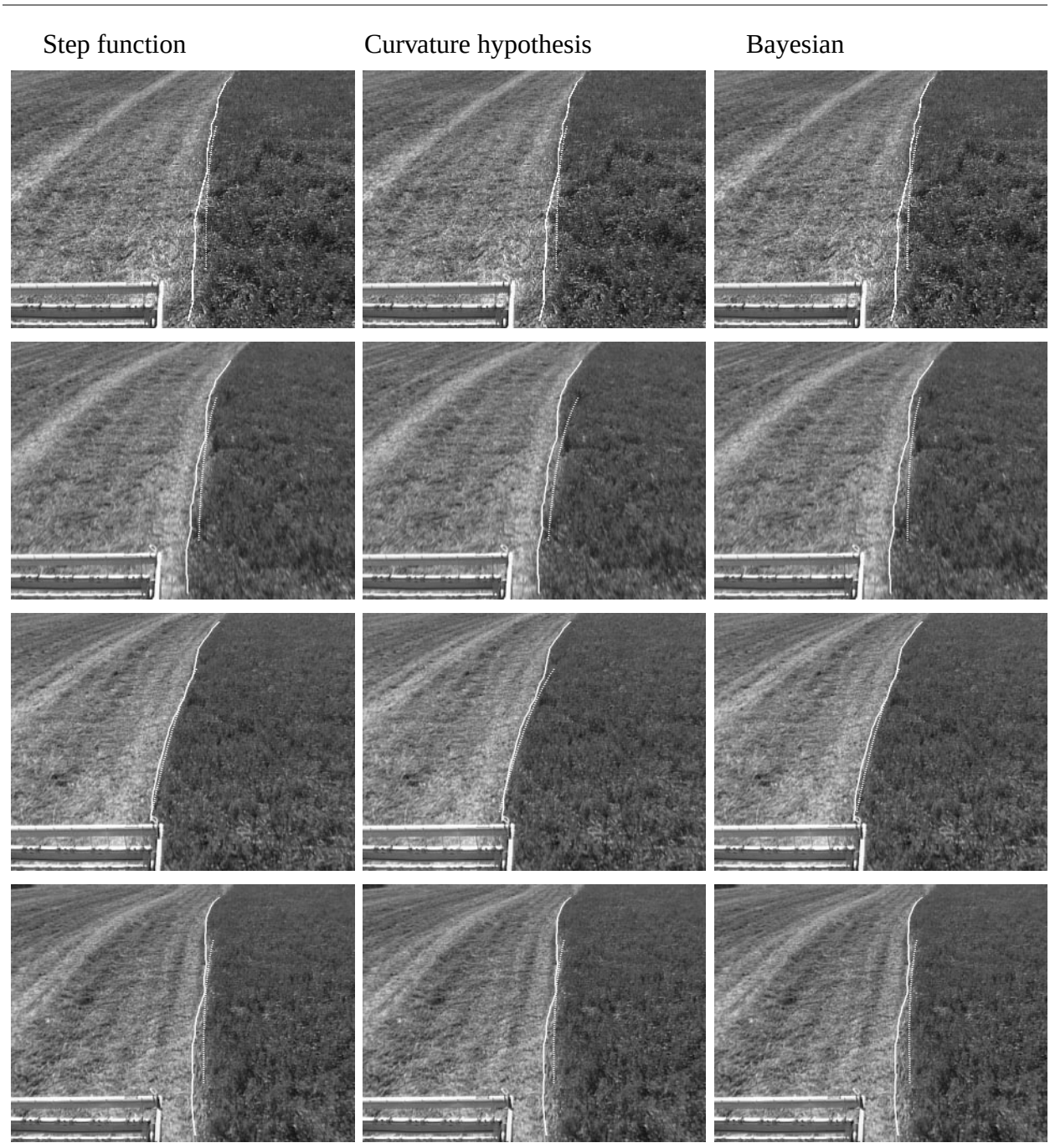
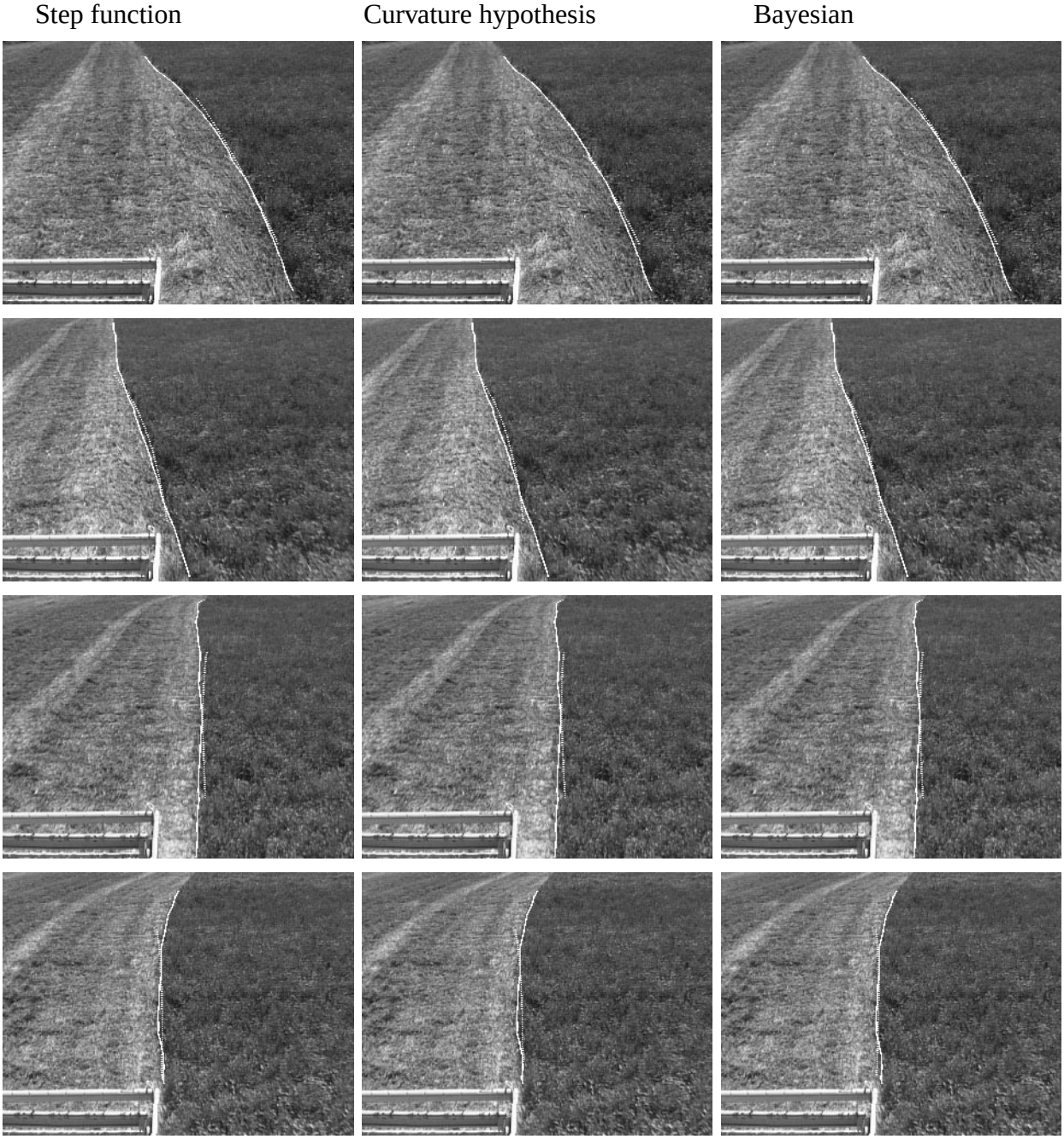

Step function
Curvature hypothesis
Bayesian

Figure 4-17: Comparison of boundary following methods.



method	*% mislabeled*	*Iteration 1 SSD*	*Iteration 2 SSD*
step function	10	7.8	7.7
	30	7.8	7.7
	50	60.3	9.7
curvature hypothesis	10	10.2	10.4
	30	10.1	10.3
	50	59.9	13.8
Bayesian	10	9.2	9.1
	30	9.3	9.1
	50	41.0	9.5

Table 4-3: Sensitivity of the three adaptive segmentation methods to erroneous training data.

contains mostly processed surface and the other contains mostly unprocessed surface causes the boundaries to converge naturally.

Conclusions

Clearly, the use of color increases the accuracy of the algorithm substantially. It is also interesting that the optimal fixed discriminant underperforms the adaptive one, because the experiment is somewhat biased in favor of a fixed discriminant; all the images in the training set were taken at the same time of day from the same row of a single field, and the fixed discriminant was optimized using the same image sequence as the test data. The advantages of the adaptive algorithm would only become more significant under a test with more varied environmental conditions.

Experiment 3: What are the convergence properties of each algorithm?

All of the adaptive methods described are supervised learning techniques; that is, they require a set of labeled examples in order to learn. In our application, the learning examples for one image are taken from the computer-generated segmentation of the previous image, instead of from a human expert; thus, any error in this segmentation will result in some of the training examples being mislabeled. Since segmentations are never perfect, the robustness of the adaptive algorithms to mislabeled examples is of considerable interest.

The goal of this experiment is to quantify the robustness of each of these methods to mislabeled training examples. We accomplish this by intentionally randomly mislabelling some of the examples, and then measuring the quality of the resulting segmentation in the same manner as in the previous two experiments. Using the new segmentation for training data, we re-segment the image and iterate the process. Table 4-3 shows the average errors of the resulting segmentations after 1 and 2 iterations.

Conclusions

All three of these methods are clearly robust to random errors in the training data; even when the classifications are entirely random (half of the data is mislabeled), the adaptive methods converge to quite good results after only two iterations. Since all three algorithms share this result, we conclude that this convergence is due to the nature of the learning problem rather than the characteristics of a particular adaptive method. We feel that this result is due to the geometry of the images being segmented; the fact that one side of the image

ignoring any color information; and 3) an RGB discriminant which adapts separately to each image. Results are plotted Figure 4-16 and summarized in Table 4-2.

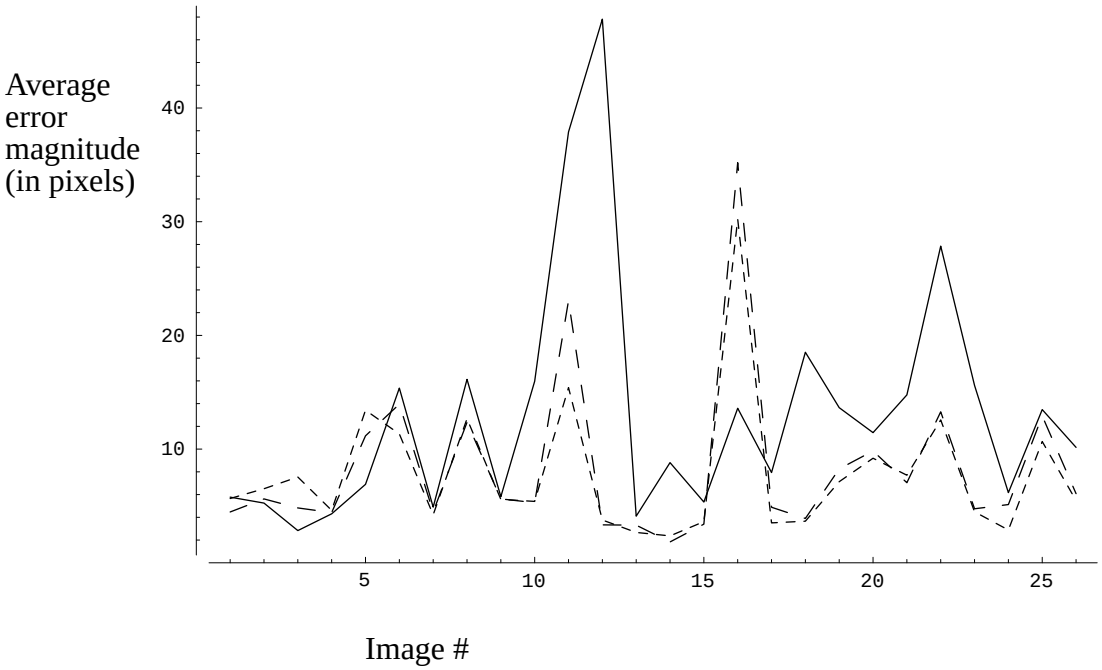


Figure 4-16: Plot of error magnitude for three discriminant types.

Method	*avg. error magnitude*
Intensity	13.1
color (fixed)	8.4
color(adaptive)	7.8

Table 4-2: Error magnitudes for some discriminant types

However, in another application in which the distributions of the processed and unprocessed areas are less unimodal, it is likely that the Bayesian method would be more appropriate; and in arenas in which the boundaries change slowly or can be regulated to simple curves, the potential computational complexity of the curvature postulation method may be less important than its ability to take advantage of multiple environmental cues.

Experiment 2: Benefits of color and adaptive discriminants

Two of the distinctions that set this research apart from most of its precursors is the use of color for the segmentation and the ability to adapt the algorithm to the environment in real time. This experiment is designed to quantify the advantages provided by both the color and the adaptive capability.

The experiment is carried out as follows: using training points from all of the 25 hand-marked crop images from the first experiment, we calculated the global best-fit Fisher linear discriminant in RGB space. We then compared the segmentation errors produced using: 1) this fixed discriminant; 2) a discriminant which weights all the colors equally, thereby

harvesting domain, therefore, the Bayesian method's ability to represent multimodal distributions seems to offer little advantage.

Conclusions

Road following and boundary tracking for surface coverage are conceptually very similar. In practice, however, the domain knowledge that can be relied upon for algorithm design differs for the two tasks. A consequence of this difference is that techniques which are robust across many types of noise found in road following may appear more brittle when applied to a boundary tracking task. Images of highways, are subject to a wide variety of noise, but the projection of the road shape into the image tends to span a fairly small space of possibilities. Images of crop line boundaries are subject to much less noise (there are few spurious features and feature occlusion is unlikely), but the projection of the boundary shape into the image is less predictable. The curvature hypothesis technique is intuitively better optimized for the first circumstance than for the second.

The experiment discussed above supports this conclusion, as it appears that the most troublesome cases are those in which the boundary is not well modeled by the constant curvature arcs which are assumed. The magnitude of this problem might be reduced by postulating a wider family of curve shapes, but only at the expense of running time; even so, it is unclear whether a wide enough family of boundaries such could be efficiently evaluated to process boundaries such as appear in Figure 4-14.

Each of the three boundary following methods discussed have some advantageous characteristics. As shown in Table 4-1, the average accuracies of all three methods are roughly comparable in the alfalfa harvesting domain, slightly favoring the step function method. The lack of tuning parameters, simplicity, flexibility of its output model, rapid running time, and efficient memory usage of the step-function based method have made it our method of choice for the Demeter system. While the step function method does have difficulties with sparse crop, camera glare, and shadows, these have all proven to cause problems for the other two methods as well.

A variation on this type of failure is shown in Figure 4-15 (taken from another image sequence). Here, it is the orientation of the boundary which is the source of the problem,

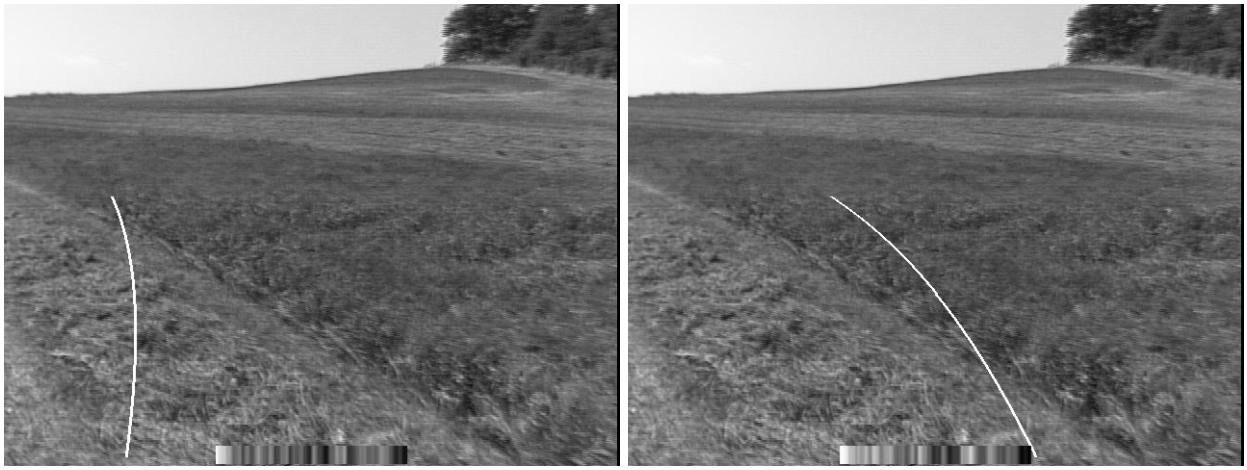


Figure 4-15: The highest scoring curvature (left) and one of the rejected candidates (right).

not its shape. Although the curvature candidate on the right in Figure 4-15 is a more natural choice, it is not surprising that the curvature hypothesis method fails to select it, for there is no structure in the image which aligns well with the high curvature arcs. Again, in a road following domain, this orientation problem is less of an issue; once the vehicle is aligned with a highway, it may be possible to stay aligned for many miles. However, in a surface covering application, it is necessary to reposition the vehicle at the beginning of each row; during this process the alignment of the harvester may vary substantially. The image in Figure 4-15 occurred at the beginning of a row, and is misaligned for precisely this reason.

The differences between the Bayesian method and the step function method in this experiment are minor. In a way, this is quite surprising; in the first ten or so images, there is a prominent windrow (the swath of cut crop laid down by the harvester) on the cut side of the field. This badly violates the bimodal assumption of the step function method, and is exactly the kind of distribution for which the Bayesian method is designed. However, as is shown in Figure 4-15, the step function fitting method actually performs better in the images with a highly visible windrow than in images where the windrow is less apparent. The explanation for this is probably that the step function fitting method works best where the crop is densest, and dense crop also produces a highly visible windrow. At least for the

Figure 4-14: Image #10: The highest scoring curvature (left) and one of the rejected candidates (right)

The left image in Figure 4-14 shows perhaps the most obvious failure of the curvature postulation method. One possible explanation is that the curvature range considered (-.03 to .03) is simply too small for this image. To test this hypothesis, we reprocessed the image using a wider range of curvature candidates (-.07 to 0.07); this allowed curvatures as large as the one shown on the right of Figure 4-14, although the width of the processed window had to be substantially reduced in order to keep the projected curves within the image window. Expanding the curvature range, however, did not alter the curvature of the calculated best estimate, which remained nearly straight ahead.

Thus, curvature range is not the issue. The problem is that because of the irregular shape of the crop line in Figure 4-14, image features do not line up well along constant curvature arcs. Why, then, was RALPH able to function so robustly in the road-following domain? Roads (and especially highways) are designed to minimize both sharp curvature and sharp changes in curvature; this is necessary to allow Ackerman-steered cars to proceed safely at high speeds. Thus, this kind of curvature modeling problem may not be important in the road following domain. In a boundary covering applications, however, the path which is to be followed is not carefully laid out in advance by human engineers. Instead, it is simply a kind of trace of the vehicles' previous path along the surface, and the shape is therefore likely to vary much more widely than that of a typical road.

seen in Figure 4-17, in which the hand-marked boundaries are shown as a solid line and the algorithm-detected boundaries are shown as dashed lines.

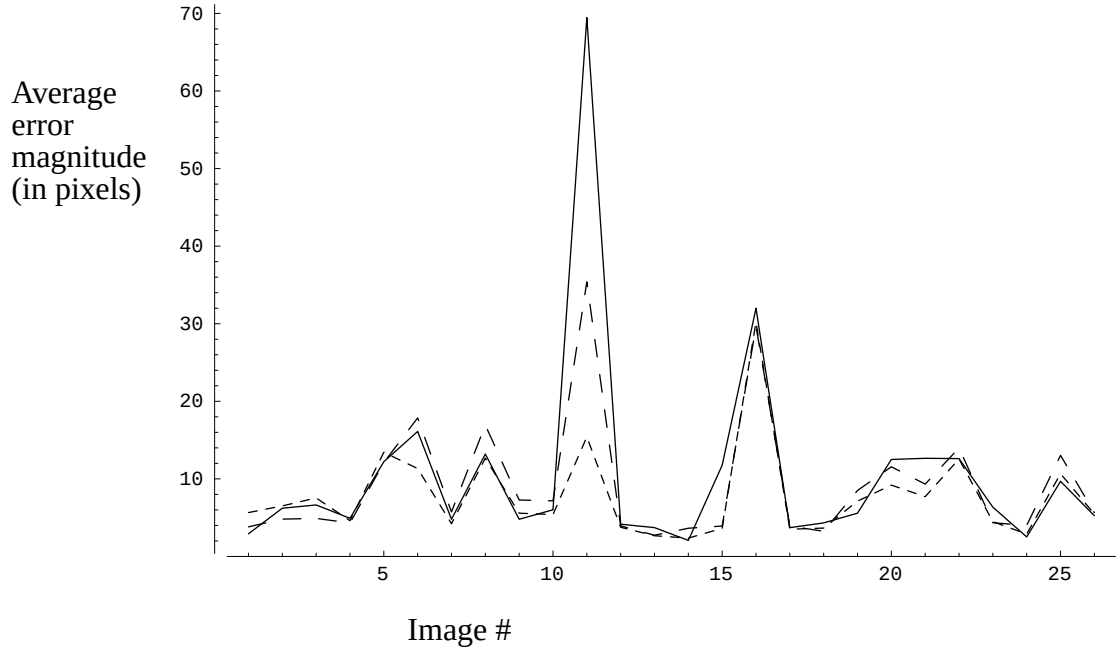


Figure 4-13: Error plot for three boundary sensing methods.

Method	*avg. error magnitude*
step function	7.8
curvature hypothesis	10.6
Bayesian	9.3

Table 4-1: Average error magnitudes for boundary sensing methods.

ary; the curvature postulation method returns a curvature and an offset, whereas the step

Figure 4-12: Average absolute error comparison.

function method returns an array of boundary points.

Because each boundary point is independently estimated, the individual error magnitudes of these boundary points can be high even if, as a group, they provide a statistically good estimate of the boundary. In order to put the methods on more even terms, we fit the boundary point data for the step function and Bayesian methods with the same model which the curvature postulation method naturally produces: a constant curvature arc with a horizontal offset. This curvefitting step is not performed while steering the Demeter harvester; it was done only for this experiment, in order to allow meaningful cross-method comparisons.

Results from this experiment are plotted in Figure 4-13, which shows error statistics for each of these methods. The results are summarized in Table 4-1; the actual images can be

4.3 Image experiments

The qualitative, system wide experiments described in the previous section provide a good measure of the ultimate usefulness of our methods. There are, however, a number of interesting questions which are not easily answerable from such data, such as the following:

- Which of the several boundary tracking methods described in Section 3.1 is the most accurate?
- How much of an advantage does the use of a color camera provide over a black and white one? The use of an adaptive algorithm over a static one?
- How sensitive are the adaptive algorithms to errors in the training data?

In this section, we describe some experiments to investigate these issues. Instead of being performed in the field, these experiments are done with canned image sequences; this allows for more careful control of conditions and a higher degree of repeatability than would be possible in the field.

Experiment 1: Which method works best for crop line tracking?

The goal of this experiment is to find out how accurately each method finds the crop line boundary between cut and uncut crop. We selected a sequence of 26 images from a field in Hickory, PA for which we had camera calibration data, and hand-marked a crop line on each one. This provided the baseline against which the 3 methods were tested.

Initially, we used average absolute error as the metric; that is, for each processed scan line, we computed the average magnitude of the difference between the hand-marked crop line and the automatically segmented one. Figure 4-12 shows a typical result; the solid line shows the hand-marked crop line, the dashed line shows the arc produced by the curvature postulation method, and the scattered points show the output of the step function method. In this example, the average absolute error for the step function method is 11.96 compared to 3.79 for the curvature postulation method.

It soon became apparent, however, that this was not providing a good measure of the relative quality of the data. The problem is that the methods use different models of the bound-

Obstacle detection

The obstacle detection algorithm is not integrated in to the Demeter system. However on a number of trial images it has successfully differentiated between crop and obstacles such as people, as shown in Figure 4-11.

Figure 4-11: Detecting a human as an obstacle: original image (left) and processed image (right).

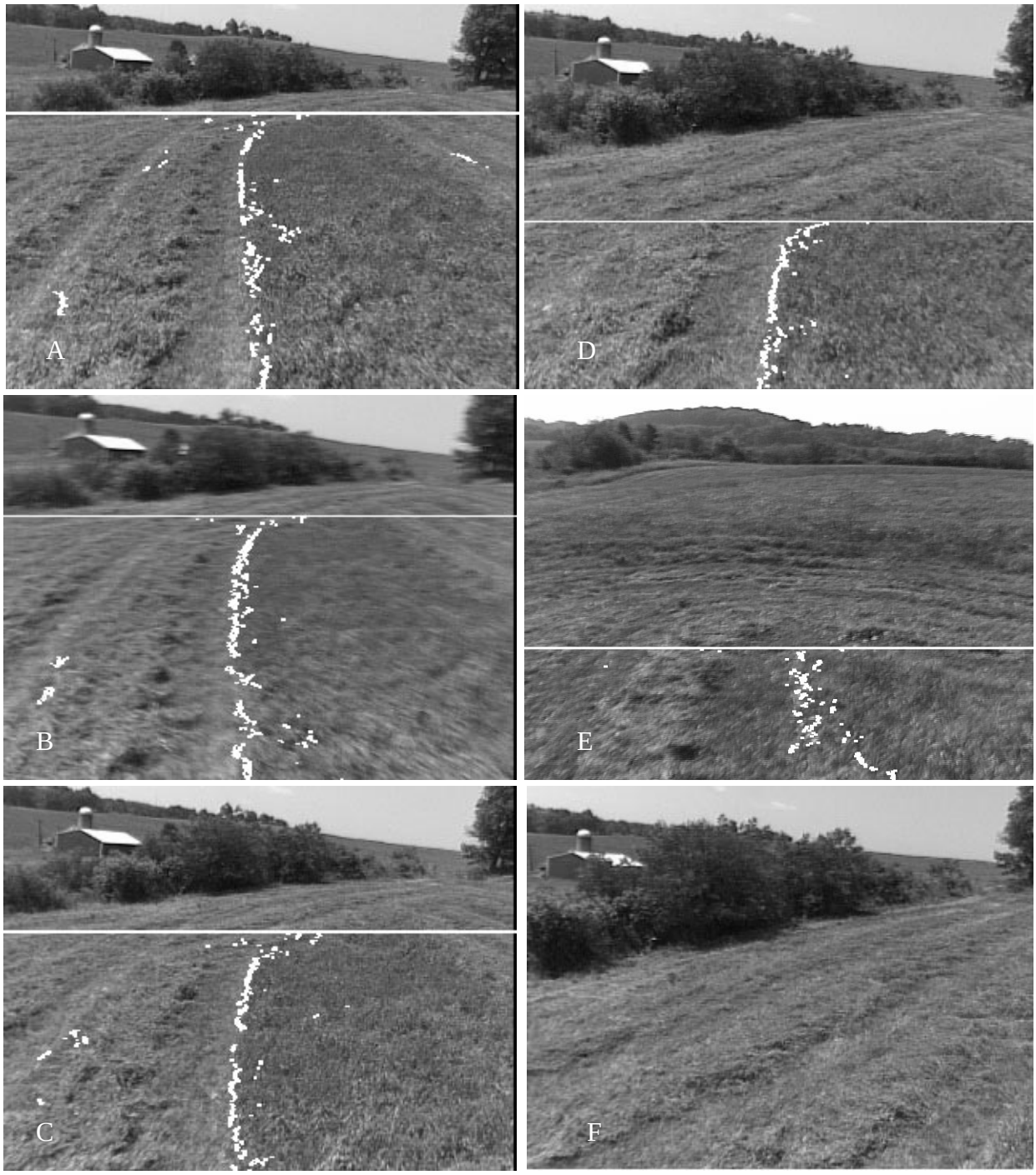


Figure 4-10: Another sequence of end-of-row images.

Small, patchy shadows, however, seem to pose some difficulty. The problem seems to be that the simplified sunlit/skylit model breaks down near the edge of a shadow, where inter-reflections from the neighboring sunlit patches can be significant. Thus, patchy shadows or long narrow shadows often are compensated incorrectly, and the results can interfere with the crop line tracker, as shown in Figure 4-9.

Figure 4-9: A shadow detection failure.

End-of-row detection

The end-of-row detector works well in many cases, even when some of the fundamental assumptions of the algorithm are violated; Figure 4-10, for example, shows a case where the only a few rows of uncut crop are left, and where the processed part of the field can be seen on both sides of this remaining crop island. However, there have been problems with false positives, which can be triggered by patchy or irregular crop. Part of the difficulty is that the system tolerance for error in this regard is extremely low, since there is currently no opportunity to recover from even a single end-of-row which is falsely triggered. This problem is being addressed by using the positioning data from the GPS and inertial sensors to doublecheck end-of-row triggers, so that false triggers that occur in the middle of the field are filtered out. (The crop line follower is not as sensitive to occasional failures, since inertia and the smoothing provided by the arbiter module can guide the machine through short stretches where the boundary detection algorithm has difficulty).

of soil and cut crop from the areas that have already been harvested.

Figure 4-7: A failure of the crop line tracker.

A detailed discussion of the relative merits of the three boundary-following algorithms, as well as additional examples of boundary extraction, can be found in the discussions following Experiment #1 in Section 4.3.

Shadow detection

The shadow detection algorithm works well for large, block shadows such as those cast by the harvester; Furthermore, the same set of compensation values seem to work in multiple locations; Figure 4-8 for example, shows a compensated image from Kansas that was compensated with the same values of C_{red}, C_{blue}, and C_{green} as the Pennsylvania image shown in.Figure 3-18.

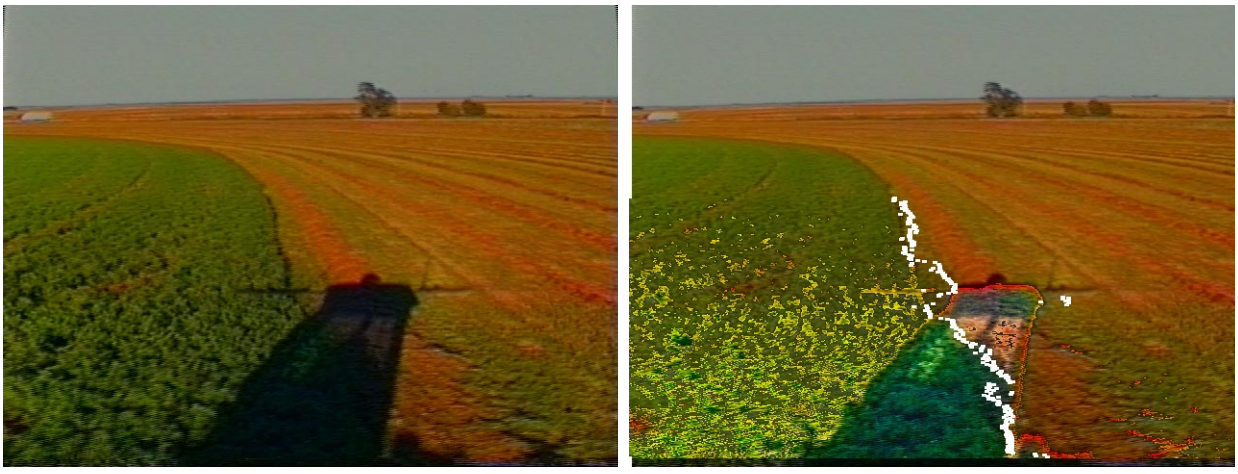


Figure 4-8: A shadow compensation example from Kansas.

effective lookahead distance is controlled by the mounting angle of the camera and the selection of scan lines which are processed. If the camera is pointed down at a sharp angle, and only the bottom few scan lines are processed, the control algorithm will attempt to follow closely only the few feet of boundary immediately in front of the harvester. Conversely, if the camera is pointed down at a shallow angle and/or scan lines from further up the image are included, the harvester may be influenced by the boundary many feet away. Whether or not this is desirable depends on many factors, including the precision needed for the cut, the shape of the boundary, and the speed of the vehicle.

4.2 Results from field trials

To date, over 80 acres of alfalfa have been autonomously harvested in three different states across the country; the bulk of it from rectangular fields in El Centro, CA, but also several acres from circular fields in Kansas and irregular fields in Pennsylvania. In general, the results from the crop line follower works have been quite good; the end of row detector and the shadow compensation methods have been partially successful, but are susceptible to failure in some conditions.

4.2.1 Crop line following

In general, the performance of the crop line follower is quite good during the morning and early afternoon; we have, for example, been able to use it to cut entire 1 mile rows from large circular fields in Kansas without a single failure. However, it degrades noticeably during the late afternoon and into the evening. We postulate that there are several effects contributing to this problem:

- Blinding of the camera by the sun. This can be partially addressed by controlling the camera angle, but unless the field is extremely flat, it is difficult to ensure that the horizon line never pops into view while keeping a reasonable lookahead distance.
- The increased presence of late afternoon shadows.
- The angle of the lighting of the uncut crop. When the sun is overhead, the primary reflected light from the uncut crop comes off the alfalfa leaves; the angled light late in the day illuminates the stalks as well, which sometimes contrast poorly with the mixture

Two examples of this procedure are shown in Figure 4-6. Both the left and right figures show a time sequence of profiles, with the most recent at the top and the composite voting profile at the bottom. The figure on the left comes from a clean cut line; it is clear that all three individual voting profiles roughly agree on the appropriate steering command. The figure on the right comes from a much less well-defined cut line. The correct steering direction in this case is a right turn; unfortunately, it can be seen that the most recent profile (the top one) indicates a preference for a left turn. By using the algorithm described above to create a composite profile, a correct steering command (right turn) is still computed.

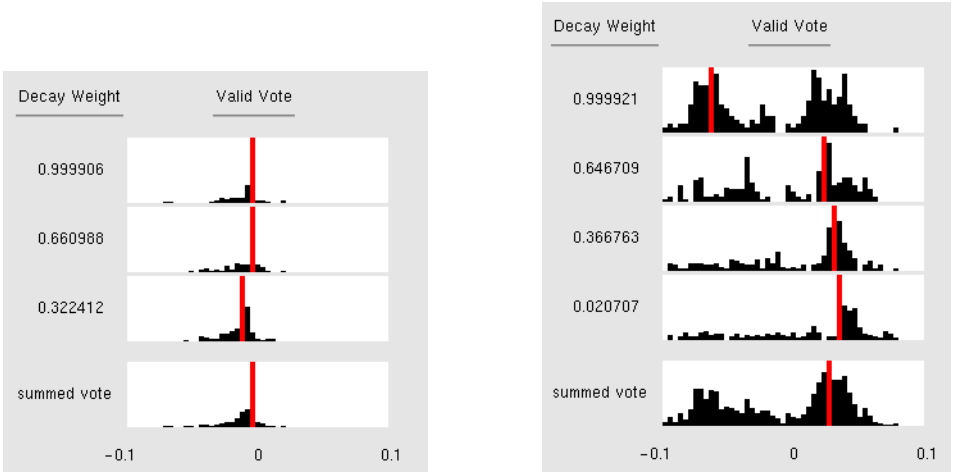


Figure 4-6: Computing a steering command from a set of voting profiles.

A discussion of the control aspects of this algorithm is beyond the scope of this thesis. However, it is worth noting some of the implications this method has for the crop line following algorithm.

- The fact that in the final step, only the direction for which w_k is largest has an effect means that boundary “outliers” in the image will have no effect on the resulting steering direction, as long as there are a sufficient number of votes for the correct direction.

- The algorithm has the effect that crop line boundary points in the image will vote for steering directions which will bring the edge of the machine’s cutting blade near those boundary points. The “nearness” is controlled by the smoothing step.

- Each of the votes is generated by a scan line from the crop line follower image. Thus, the

of the end-of-row detector has been discussed in detail in Section 3.2, and thus will not be discussed again here.

Turn behavior

The turn behavior is responsible for guiding the harvester through preprogrammed turn sequences based on dead reckoning and GPS readings. The paths are specified in terms of via points, and are planned off-line. The module compares the current estimate of its position to its intended path, and then uses pure pursuit to compute a steering recommendation. As with the crop line follower, this recommendation is used to create a steering profile to be sent to the arbiter.

Arbiter

The arbiter receives steering recommendations from one or more modules, and reconciles them into a single steering command to be sent to the controller. It is also used to provide temporal "smoothing" of steering commands to prevent sudden jerks in the harvester steering.

In our initial tests, the arbiter was configured to simply send the mean of the steering votes to the controller. We have found, however, that a slightly more sophisticated algorithm provides more robust behavior in the face of outliers. The algorithm comes from Rosenblatt [26], and runs as follows:

- Discretize the possible steering commands into N bins.
- Associate with each bin 0...N-1 a weight, w_k; initialize these weights to 0.
- For each vote in each of the steering profiles generated in the last t seconds, compute the bin k into which it falls and increase w_k by a time-decayed increment. These w_k create a composite voting profile.
- Smooth resulting array w_k.
- Find the k for which w_k is largest, and send the appropriate steering command.

calibration, yet still allows for tuning through the selection of the control constant K and the selection of which piece of the boundary to use for generating votes.

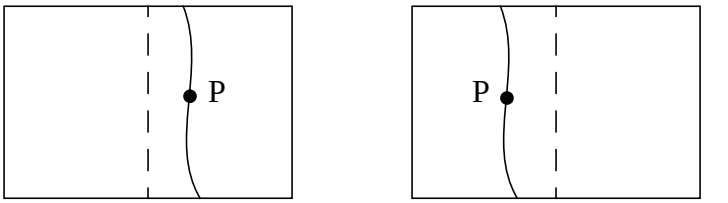


Figure 4-4: Control in image space.

The second method is based on the commonly used pure pursuit algorithm[29]. Here, each boundary pixel votes for the steering curvature which connects the vehicle's current position to the actual, physical world location corresponding to the boundary point, as shown in Figure 4-5. Camera calibration is required in order to map the boundary point P from image coordinates to real world coordinates.

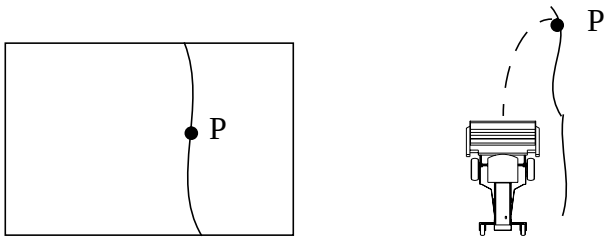


Figure 4-5: Pure pursuit control.

The cameras are calibrated using a method developed by Tsai[27].

Using either of these methods produces, for each image, a collection of steering votes (a "steering profile"). These profiles are then sent to the arbiter, which then sends the actual steering commands to the controller.

End of row detector

The function of the end-of-row detector is to send a trigger message to the arbiter when the end-of-row has been detected. The arbiter uses this message as an indicator to transition from the crop line following behavior to the turn behavior. The image processing portion

```
End

StartState CutOnLeftStartState
```

Crop line tracking behavior

The function of the crop line tracker is to process images digitized from the two cameras, and to issue appropriate steering recommendations so that the harvester follows the edge cut during the previous pass. Two of the boundary detection methods described in Section 3.1 have been used to steer the Demeter harvester: the Bayesian method and the step-function-fitting method. However, the lack of tuning parameters, simplicity, flexibility of its output model, and rapid running time of the step-function based method have made it our method of choice for alfalfa harvesting. It runs quite quickly on the Pentium processor; at full frame rate (30 Hz) have been achieved for the adaptive step-function algorithm, including the shadow-compensation preprocessing, on quarter-resolution (160x120) images.

We have implemented two different methods for converting crop line boundaries into steering commands. The first method merely attempts to control the location of the crop line in the image; the intuition behind this method is that the camera is mounted such that when the harvester is on course, the crop line is centered in the image; deviations of this boundary from the center line are therefore undesirable and should be corrected. Each point on the boundary votes for a steering curvature according to a simple proportional control law:

$$curv = K\left(j_{bound} - \frac{j_{last}}{2}\right) \tag{4-1}$$

where K is a proportionality constant, j_{bound} is the column coordinate of the boundary pixel, and j_{last} represents the maximum column coordinate. Negative curvatures represent left turns and positive curvatures represent right turns; thus, as shown in Figure 4-4, boundary points to the right of center vote for right turns, and boundary points to the left of center vote for left turns. The advantage of this method is that it is simple and requires no camera

```
State TurnRight

Behavior globalTrajTracker

Behavior localTrajectoryTracker Args “180TurnRight $xTurn $yTurn” Ramp-
UpWeight 1.0 RampUpTime 0.1 RampDownWeight 0.0 RampDownTime 0.5

Transition CutOnRight If *EndOfTurn* Continue <globalTrajTracker>

End


State CutOnRight

Behavior CropLineFollower Args “RightFrontCamera” RampUpWeight 1.0 Ramp-
UpTime 0.5

Behavior globalTrajTracker

Transition NearTurnLeft If *NearEOR* Continue <CropLineFollower(“Right-
FrontCamera”) globalTrajTracker>

End


State NearTurnLeft

Behavior CropLineFollower RampDownWeight 0.0 RampDownTime 0.2

Behavior EndOfRowDetector Args “RightFrontCamera”

Behavior globalTrajTracker

Transition TurnLeft If *EndOfRowDetected* Continue <globalTrajTracker>
End

State TurnLeft
Behavior localTrajectoryTracker Args “180TurnLeft $xTurn $yTurn” RampUp-
Weight 1.0 RampUpTime 0.1 RampDownWeight 0.0 RampDownTime 0.5
Transition CutOnLeft If *EndOfTurn* Continue <globalTrajTracker>
End


State Stop
```

```
# the perform a 180 degree spin turn and come back up the next row.
#
#

State CutOnLeftStartState
Behavior CropLineFollower Args “LeftFrontCamera” RampUpWeight 1.0 Ramp-
UpTime 0.5
Behavior globalTrajTracker Args “fieldData.input”
Transition NearTurnRight If *NearEOR* Continue <CropLineFollower(“Left-
FrontCamera”) globalTrajTracker>
End

State CutOnLeft
Behavior CropLineFollower Args “LeftFrontCamera” RampUpWeight 1.0 Ramp-
UpTime 0.5
Behavior globalTrajTracker
Transition NearTurnRight If *NearEOR* Continue <CropLineFollower(“Left-
FrontCamera”) globalTrajTracker>
End

State NearTurnRight
Behavior CropLineFollower RampDownWeight 0.0 RampDownTime 0.2
Behavior EndOfRowDetector Args “LeftFrontCamera”
Behavior globalTrajTracker
Transition TurnRight If *EndOfRowDetected* Continue <globalTrajTracker>
End
```

The execution monitor takes as its input a script, which describes transitions from one behavior to another. A typical piece of this script might read as follows:

```
State CutOnLeftStartState

Behavior CropLineFollower Args “LeftFrontCamera” RampUpWeight 1.0 Ramp-
UpTime 0.5

Behavior globalTrajTracker Args “fieldData.input”

Transition NearTurnRight If *NearEOR* Continue <CropLineFollower(“Left-
FrontCamera”) globalTrajTracker>

End
```

This fragment describes a particular mode “CutOnLeftStartState” in which the crop line follower module, using the left front camera as input, gradually takes control of the steering command over a 0.5 second fade in. Simultaneously, another module runs which estimates the harvesters’ global position.

When the execution monitor receives a message from the global trajectory tracker indicating that the end of row is approaching, the system will transition to another mode “NearTurnRight”; both the crop line follower module and the global trajectory tracker module will continue to operate in the new mode.

A full script input for the execution monitor is shown below. (This script doesn’t correspond precisely to the diagram in Figure 4-3; the initial cut around the edges of the field is not included in the script, and there are separate states for cutting depending on whether the harvester’s left camera or right camera is being used. Conceptually, however, the differences are minor).

```
# Written Sep 3rd, 1996 by Tom Pilarski

#

# Below is an Execution Monitor finite state machine description for

# harvesting a field, either circular or polygonal, in a switchback

# type fashion.  Switchback, being cut down one row and at the end of
```

3. When dead. reckoning/GPS indicates the end of the row is near, run the end-of-row detector.

4. When the end-of-row is detected, turn 180 degrees.

5. Go to step 2.

As discussed by Hoffman [10], this plan can be encoded into a finite state machine as shown in Figure 4-3:

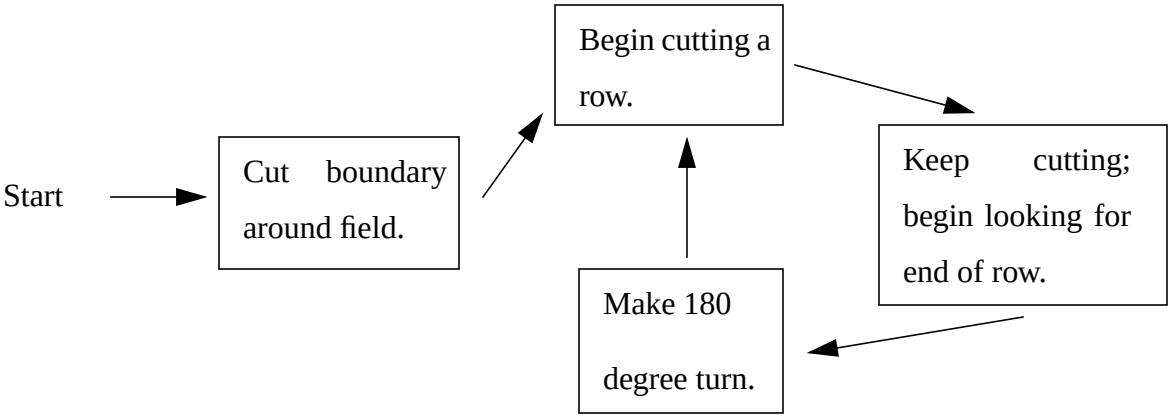


Figure 4-3: Finite state machine for harvesting a field.

Execution Monitor

In its early incarnations, the execution monitor simply started and stopped various behaviors which controlled the harvester; thus, when an “end-of-row” message was received by the end-of-row detector, for example, the crop line following behavior would be shut off and the turn behavior would be initiated. Sometimes, however, the harvester steering can “jerk” during these transitions between states, when the controller suddenly switchefrom taking commands from the crop line follower to taking commands from the turn behavior. In order to allow for smoother transitions, a mechanism was added to allow these behaviors to be gradually faded in and out over the course of a few seconds; thus, for a brief period of time, the controller actually receives a merged steering command from both behaviors. The module which handles this merging is called the arbiter, and is discussed later on in this section.

4.1.2 Architecture

The architecture for the Demeter harvester is behavior based; that is, the logical structure of software is broken down among particular behaviors (such as tracking a crop line or turning the machine 180 degrees). These behaviors are coordinated by a top-level control module called the execution monitor. The execution monitor takes as input a script describing a high level plan for how the field is to be harvested; it then invokes a sequence of behaviors such a way as to carry out the plan.

Consider, for example, a simple, rectangular field with no obstacles. This field might be harvested with a simple back-and-forth pattern as shown in Figure 4-2.

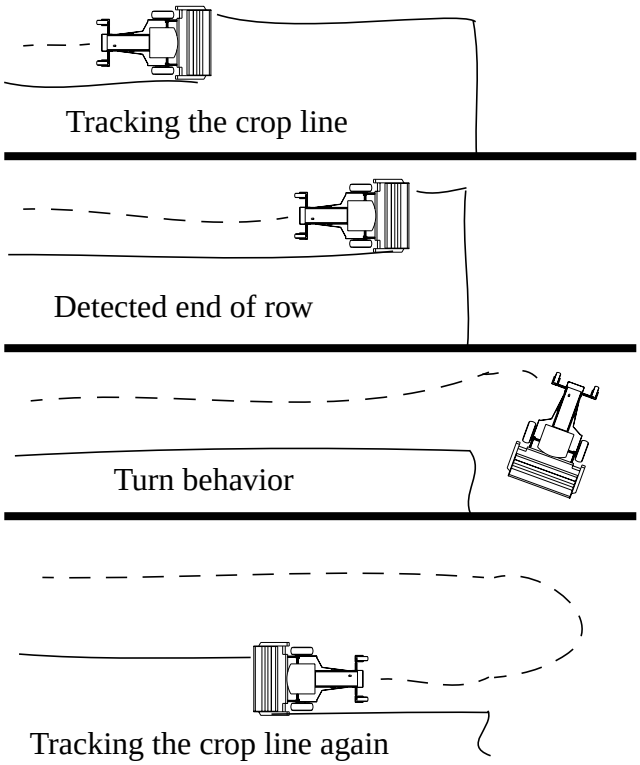


Figure 4-2: Harvesting a rectangular field.

A rough plan for harvesting the field in this manner might encompass the following steps:

1. Using purely GPS based guidance, cut a boundary around the edge of the field.

2. Using the crop line tracker, begin cutting a row.

4. Experiments with the Demeter harvester

Our primary testbed of the perception capabilities described in Chapter 3 has been the Demeter harvester. In this chapter, we present some results from field experiments using this machine. Section 4.1 describes the Demeter harvester, and how some of these perception capabilities were integrated onto the harvester to produce a fully autonomous system. Section 4.2 discusses some (primarily qualitative) results from autonomously harvesting actual crop in several different fields. Section 4.3 presents some more controlled experiments performed on image sequences in order to quantify the performance of the perception algorithms. Results from other, non-harvesting applications will be presented in Chapter 5.

4.1 Description of the Demeter system

4.1.1 The testbed

The harvester is shown in Figure 4-1; it is a New Holland 2550 Speedrower retrofitted with wheel encoders and servos to control a number of machine functions, such as the throttle, steering and cutter bar. A Sun Sparc 20 board running a real time operating system (VxWorks) is used to control machine functions; a 200MHz Pentium PC is dedicated to the perception system. An on board GPS receiver coupled with a fixed base station allows the use of differential GPS-based positioning. Forward-facing RGB cameras equipped with auto-iris lenses are mounted to either side of the cab roof, near the ends of the harvester's cutter bar.

Figure 4-1: The Demeter automated harvester.

were found to be less useful than the experimentally determined values. This discrepancy may be due to the inadequacy of the blackbody spectral distribution function as a model for skylight.

The method described above necessarily makes a number of simplifications. The red, green, and blue filters actually each pass a range of frequencies; SPD variations in sunlight and skylight within a single band are not taken into account. Shadowed areas can receive significant illumination from reflected light from neighboring sunlit areas; such interreflections are not modeled. The differing effects of lighting angle for sunlight and skylight are ignored, as are non-linearities in the CCD chip response.

The values of C_{red}, C_{blue}, and C_{green} depend on color of both the sunlight and the skylight. These colors can vary across different times of day and different atmospheric conditions. In our application, shadows typically cause the most trouble on cloudless days in the late afternoon; we therefore chose coefficients optimized for this case.

Despite these limitations, applying this method allows crop lines to be successfully extracted from a number of images which would otherwise return incorrect results. For example, applying the shadow compensation method described above to the image shown in Figure 3-17 produces a much improved estimate of the crop boundary, as shown in Figure 3-18.

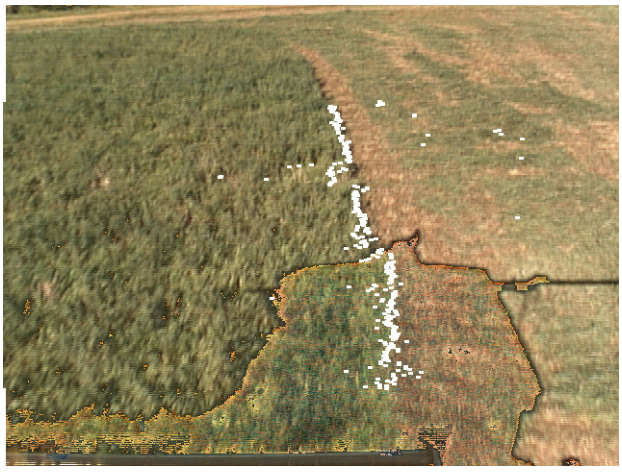


Figure 3-18: A successful example of shadow compensation.

unknown. However, if we approximate $\bar{r}(\lambda)$ as a delta function with a non-zero value only at λ_{red}, then (3-12) and (3-13) simplify to

$$R_{sun} = r_0 I_{sun}(\lambda_{red})\rho(\lambda_{red}) \tag{3-14}$$

and

$$R_{shadow} = r_0 I_{shadow}(\lambda_{red})\rho(\lambda_{red}) \tag{3-15}$$

so that R_{sun} and R_{shadow} can be related by a constant factor C_{red}:

$$R_{sun} = R_{shadow}\left(\frac{I_{sun}(\lambda_{red})}{I_{shadow}(\lambda_{red})}\right) = R_{shadow}C_{red} \tag{3-16}$$

The same analysis can be repeated for the G and B pixel values. Under the assumptions given above, the parameters C_{red}, C_{blue}, and C_{green} remain constant across all reflectance functions $\rho(\lambda)$ for a given camera in a given lighting environment.

Implementing this shadow compensation therefore requires

1) the selection of appropriate constants for C_{red}, C_{blue} and C_{green},

2) a method for determining whether points are shadowed or unshadowed, and

3) “Correcting” the shadowed pixels using Equation (3-16).

Determining whether points were shadowed or unshadowed was accomplished by intensity threshholding. Approximate values for C_{red}, C_{blue}, and C_{green} were hand- selected by experimentation on several images containing significant shadow; for our application values of $C_{red} = 5.6$, $C_{green} = 4.0$, and $C_{blue} = 2.8$ were found to work well.

An attempt was made to calculate C_{red}, C_{blue} and C_{green} values a priori from blackbody spectral distribution models of sunlight and skylight. This calculation produced qualitatively the correct result, e.g. $C_{red} > C_{green} > C_{blue}$; however, the a priori calculated values

crop; depending on local conditions, such natural intensity differences can be a useful feature.

We present a technique for modeling and removing shadow noise which is based on compensating for the difference in the spectral power distribution (SPD) between the light illuminating the shadowed and unshadowed regions. In an ideal camera, the RGB pixel values at a given image point are a function of $S(\lambda)$, the spectral power distribution (SPD) emitted by a point in the environment [18]; for example, R is determined in Equation (3-10), where r_0 is a scaling factor and $\bar{r}(\lambda)$ is the function describing the response of the CCD chip and red filter; typically, this function falls to 0 outside of a narrow wavelength band.

$$R = r_0 \int S(\lambda)\bar{r}(\lambda)d\lambda \tag{3-10}$$

r_0 and $\bar{r}(\lambda)$ are purely functions of the CCD camera; our goal, therefore, is to construct a model of how shadows alter the function $S(\lambda)$.

To a first approximation, $S(\lambda)$ is simply the product of the SPD of the illuminating light, $I(\lambda)$, with the reflectance function of the illuminated surface point, $\rho(\lambda)$:

$$S(\lambda) = I(\lambda)\rho(\lambda) \tag{3-11}$$

Suppose we assume that every point in the environment is illuminated by one of two SPDs; either $I_{sun}(\lambda)$, comprising both sunlight and skylight, or $I_{shadow}(\lambda)$, comprising skylight only. Then the red pixel values for unshadowed regions will be computed by

$$R_{sun} = r_0 \int I_{sun}(\lambda)\rho(\lambda)\bar{r}(\lambda)d\lambda \tag{3-12}$$

and the red pixel vales for shadowed regions by

$$R_{shadow} = r_0 \int I_{shadow}(\lambda)\rho(\lambda)\bar{r}(\lambda)d\lambda \tag{3-13}$$

From Equations (3-12) and (3-13), we see that it is in general not possible to compute R_{sun} from R_{shadow} without knowledge of the reflectance function of the environment patch being imaged. This is problematic, because for our application, this reflectance function is always

harvester body lies directly over the region containing the crop line. This shadow is directly

Figure 3-17: Shadow noise.

responsible for the resultant error in the crop line boundary estimate produced by the crop line tracker.

Shadow noise causes difficulties for a number of reasons. It is often quite structured, and thus is not well modeled by stochastic techniques. Its effects and severity are difficult to predict; if the sun is momentarily obscured by a passing cloud or the orientation of the harvester changes rapidly, the prevalence and effect of shadow noise can vary dramatically on time scales of less than a second.

Normalizing for intensity, though an intuitively appealing method of dealing with shadow noise, fails to be useful in our application for two reasons. The primary problem is that it does not take into account the significant color changes present in shadowed areas. For example, normalizing the image in Figure 3-17 before processing still results in an incorrect crop line boundary estimate. A number of factors contribute to this color shift, but perhaps the most significant is the difference in illumination sources between the shadowed and unshadowed regions[11]; the dominant illumination source for the unshadowed areas is sunlight, while the dominant illumination source for the shadowed areas is skylight. A secondary problem with intensity normalization is that it prevents the crop line tracking algorithm from using natural intensity differences to discriminate between cut and uncut

Figure 3-16: Detecting potential obstacles.

Note that the issues concerning the use of full RGB color versus normalized color are quite different for the obstacle detector than for shadow compensation and crop line detection. Normalized color is not used as an attempt to remove shadow noise; that is accomplished using the method described in Section 3.4. Here, normalized color is used to compensate for different iris openings between the training image and the images to be processed, and also as a means for reducing the dimensionality of the PDF space. While this does prevent the use of intensity as a metric for distinguishing obstacles, we have found that obstacles typically differ from crop significantly enough in color alone that the use of intensity information is not as necessary; such is often not the case for the crop line follower, which often must distinguish between the quite similar appearance of cut and uncut crop.

3.4 Special topic: shadow compensation.

Correctly interpreting shadows is not a problem which is tied particularly closely to surface covering; rather, it is a common problem of computer vision in environments without controlled lighting (such as the outdoors). We treat it here because it is a concern for our main harvesting application, and is likely to be a concern for other surface covering tasks as well.

Shadow noise can heavily distort both image intensity (luminance) and color (chrominance). An example of a severe case is shown in Figure 3-17; here, a shadow cast by the

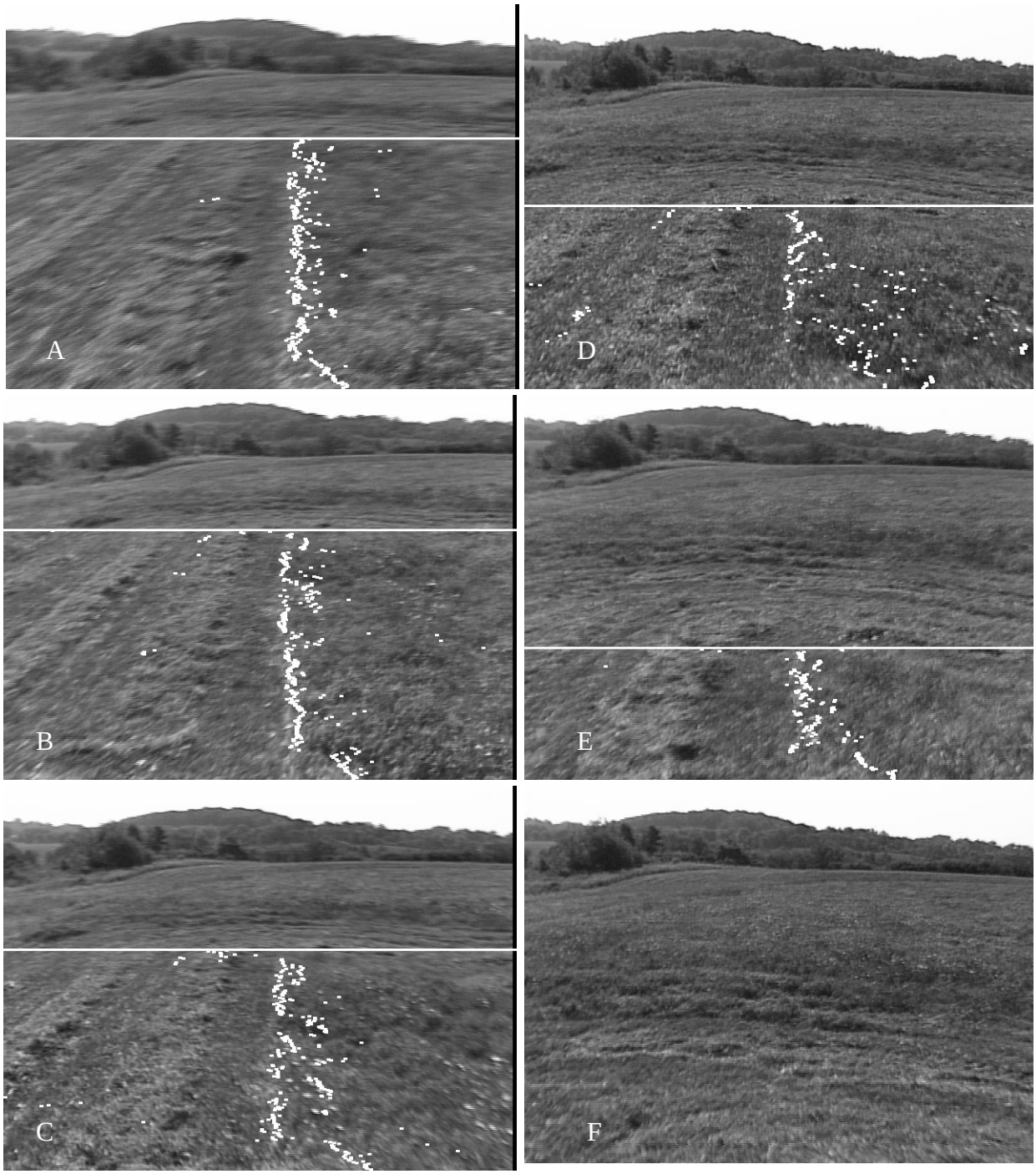


Figure 3-15: A sequence of end-of-row images.

2D histogram in normalized color space, with each cell containing an independent probability density estimate.

4) The end of row is the row i with the highest score S(i).

We experimented with a broad range of binary evaluation functions F(i). Our current version first computes the crop line fitting algorithm described in Section 3.1.1 to each scan line. The location and height of the resulting best fit step functions are then compared to precomputed ranges gathered from training data; if they fall within the allowed ranges, the boundary is accepted as a genuine crop line and the row receives a “beforeEnd” label; otherwise, the row is labeled “afterEnd”.

Our most common use for the end-of-row detection module is to trigger a transition into a turn behavior when the end of row is reached. In order to prevent a single spurious image from falsely causing an end-of-row trigger, this message is sent only after both 1) the distance to the end of row falls below some threshold and 2) a series of processed images, such as those shown in Figure 3-15, indicate the end of row has been coming progressively nearer.

3.3 Obstacle detection

Our obstacle detection algorithm was inspired by the Bayesian algorithm for finding boundaries. The basic idea is that the same PDFs which can be used to determine unprocessed from processed surfaces can also be used to locate potential obstacles in the camera’s field of view. The PDFs are built in the manner discussed in Section 3.1.2, and are combined into a single PDF for both processed and unprocessed surface. Image pixels are marked whose probability of belonging to the this PDF falls below some threshold; these are pixels with low probability of being part of either the processed or unprocessed surface. Next, regions of the image containing a large number of such marked pixels are identified as obstacles. Figure 3-16 shows an example of such an image before and after processing; potential obstacles are marked as a solid region.

As was discussed in Section 3.1.2, there are a number of options available for representing PDFs. Once again, we use a histogram model here, for similar reasons as with the crop line follower. For the obstacle detector in the alfalfa application, however, we used a discretized

Figure 3-14: Locating the end of a crop row.

Our end of row detection algorithm attempts to find the image row i which most cleanly separates those scan lines containing a crop line boundary from those which do not contain such a boundary. The algorithm, described below, first uses a binary function F(i) to classify each image row according to whether it contains a crop row boundary; next, it searches for the row which best divides the "beforeEnd" rows from the "afterEnd".

1) Digitize an image.

2) Remove shadow noise as described in Section 3.4.

2) For each scan line i in the image:

Apply a binary evaluation function F(i) to determine whether row i contains a genuine crop line boundary. Let F(i) = "beforeEnd" if i contains a genuine boundary point, and F(i) = "afterEnd" if not.

3) For each scan line i in the image, compute a score for the scan line S(i) as follows:

a) set S(i) = 0;

b) Increment S(i) for every scan line x from 0 (the top of the image) to i-1 for which F(x) = "afterEnd".

c) Increment S(i) for every scan line y from i+1 to i_MAX (the bottom of the image) for which F(y) = "beforeEnd".

plate matching step is somewhat more specific to road following than the curvature estimation step, and therefore we will not discuss it further here.

Our method of calculating offsets is inspired by the visual examination of the bin arrays shown in Figure 3-12. After the curvature calculation, there exists a pronounced “step” in the bin array at the boundary between the cut and uncut crop. This boundary exists because the array elements to the left of the step represent the summed values of a set of pixel discriminants in the uncut part of the crop along a particular curvature and offset, while array elements to the right represent the summed values of pixel discriminants in the uncut part of the crop. Since the discriminant function is specifically selected to give different values for cut and uncut crop, the step is often quite pronounced. The location of the step is provides the offset, as shown in Figure 3-13.

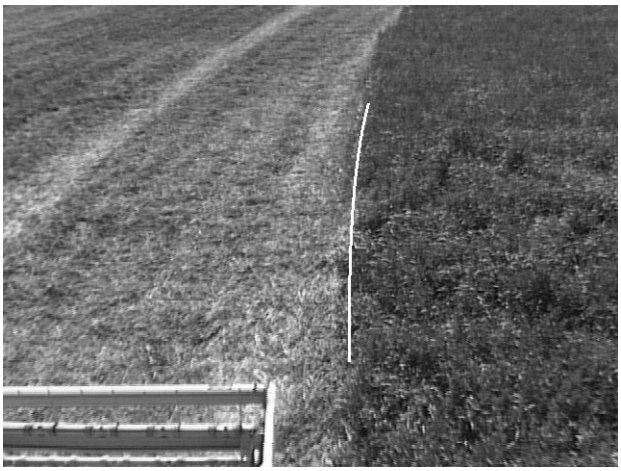


Figure 3-13: Computing the correct offset.

3.2 End-of-row detection

The goal of the end of row detector is to estimate the distance of the harvester from the end of the crop row. When the end of row boundary is approximately perpendicular to the crop line, and the camera is mounted with zero roll (as in our system), the distance to the end of row is purely a function of the image row where the crop line boundary stops. Figure 3-14 shows an image which has been correctly processed; the white line marks the computed image row corresponding to the crop row end.

environment. For the alfalfa field images, another type of enhancement may be more appropriate. We experimented with this method using both raw intensity and using the Fisher linear discriminant (which effectively enhances the cut/uncut boundary and suppresses everything else); the Fisher discriminant gave considerably improved results.

A second important issue is deciding which curvature candidates to postulate. When following roads, RALPH typically hypothesizes constant curvature arcs within some small window around the arc currently being driven. Since RALPH can cycle at over 10 Hz, and since roads generally do not change curvature quickly, the use of this window minimizes processing time and also prevents large discrepancies between successive steering commands.

In our experiments, however, the curvature to be detected changes much more quickly than that of a typical road. Postulating curvatures varying from a sharp right turn to a sharp left on every image can address this problem, at the expense of computational efficiency; for our implementation, we postulated candidate curvatures varying from -0.03 $m^{-1}$to 0.03 m^{-1} in steps of .001. An additional advantage of this approach is that the results of one image do not affect the results of the next, which simplifies the analysis of failures.

Thirdly, a value must be selected for N, the number of bin differences which are added together to compute the score. This parameter essentially controls a cutoff of bin difference magnitudes; the top N values are considered as potential feature boundaries, and the rest are ignored as noise. After some experimentation, we used a value of 28 (10% of the total number of bins) for N as the default; changing this value does not strongly affect the results.

Calculation of offsets

As can be seen in Figure 3-12, the specification of curvature alone is insufficient for calculating a boundary, since a given curvature describes a whole family of arcs. We therefore use a variant of the step-function fitting method described in Section 3.1.1 to calculate the offset of the vehicle from the actual boundary; this combination of curvature and offset uniquely describe a single boundary line. RALPH itself makes use of a second, template-matching step for determining the left-right offset to the centerline of the road. This tem-

which produce smaller discontinuities across many bins. RALPH typically uses an N which is 10-15% of the total number of bins [22].

Figure 3-12: Evaluation of curvature candidates.

This same technique can be used to find curvatures in SCT boundary tracking tasks. Figure 3-12 shows two hypothesized curvatures and the resulting bin arrays on the same image taken from an alfalfa field. (We experimented briefly with RALPH itself as a boundary tracking method; but this produced results which were difficult to interpret, since the system had been tuned and enhanced specifically for the road following domain. Thus images in Figure 3-12, as with all the experimental results shown in this section, are produced from the author's implementation of the curvature hypothesis technique used by RALPH). On each image, the parallel arcs show the candidate curvature; at the bottom of the image, the magnitude of each array bin is shown in grayscale. Applying the evaluation method described above produces a score of 2495 for the curvature candidate on the left and a score of 1955 for the candidate on the right; the left curvature candidate is therefore preferred.

Transferring this application to boundary following is largely straightforward, but there are a few issues which require consideration. One of these is the use of various contrast enhancement techniques. If the images are color instead of black and white, RALPH can be configured to use metrics other than pure intensity during the bin accumulation stage; for instance, in the road following domain, a measure of "blueness" is often added; this has the effect of enhancing road edges, since the road contains more blue than the surrounding

lems can be addressed by "seeding" each histogram cell in both the processed and unprocessed PDFS with one example. This causes all probability calculations to be well-defined, prevents probability calculation results of either 0 or 100 %, and biases probability calculations from sparse areas of the PDF towards 50%, thus minimizing their effect on the maximum likelihood calculation. This induces a slight bias towards 50% for the densely populated parts of the PDF as well, but this bias drops off quickly as the sample density increases.

3.1.3 Boundary sensing by curvature postulation

The third method we investigate for boundary sensing is based on the curvature hypothesis technique used by the recent road-following system RALPH[21]. This method is attractive for two reasons. First, it does not explicitly rely on particular road characteristics or markings, and is in principle just as capable of tracking cut/uncut crop boundaries as painted road lines. Second, RALPH has been extremely successful across a wide variety of road types and conditions, completed a 2849 mile cross country trip of which 2797 miles were driven autonomously[13].

RALPH Overview & Curvature Hypothesis

RALPH begins processing on an image by attempting to determine the intrinsic curvature of the road without regard to the road's lateral position in the image. It does this by generating and evaluating a series of road curvature candidates and selecting the candidate which scores best. Evaluating a candidate is accomplished by projecting the hypothesized curvature onto the image, summing up the pixel intensity values along this curvature into a horizontal array of bins, and then rewarding the existence of sharp discontinuities in the resulting bin array. Intuitively, if the hypothesized curvature is unrelated to any structure in the image, neighboring bins will collect pixels from a similar mixture of features, and the resulting bin array will be very smooth; if instead the hypothesized curvature parallels structures in the image, neighboring bins may collect pixels from two different features, and the resulting array will contain sharper discontinuities. The candidate score is calculated by summing the maximum N difference between adjacent bins; thus, candidate curvatures which produce large discontinuities between a few bins will fare better than candidates

tion 3.1.1 to generate training data. Since this algorithm is quite rapid, it is possible to train new PDFs in real time.

Our current implementation, in fact, trains a new set of PDFs for every image, typically using a few thousand points; it thus retains all of the adaptive capability of the step-function fitting method. Since each PDF has over 32000 free parameters, there are large regions of the histogram PDF which are either empty or poorly defined. These regions correspond to areas in the discriminant space which are rare in the environment; for instance, training sets of cut and uncut alfalfa using the RGB color space PDF typically have dense samples in the parts of the PDF which represent shades of green, and very few samples in the parts of the PDF which represent pink or blue. (See Figure 3-11 for example). As long as the training data is representative, most of the classification lookups will come from the densely populated parts of the PDF.

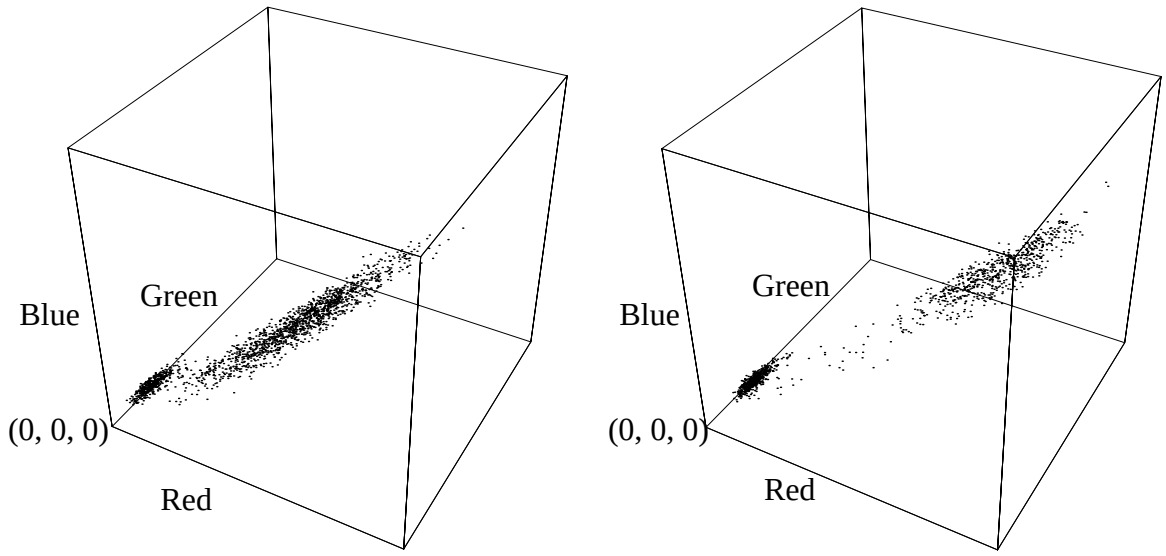


Figure 3-11: Typical training data plots for uncut alfalfa (left) and cut (right) alfalfa in RGB space.

However, the sparsely populated areas of the PDF can cause two problems. First, if the PDF cells are actually empty, probability calculations for those cells can be undefined , 0%, or 100%, all of which have undesirable effects on the maximum likelihood calculation. Second, PDF cells which are nonempty but still sparsely populated are likely to produce more erroneous estimated than their more densely populated counterparts. Both of these prob-

Constructing the pdfs

Traditionally, a wide range of representations have been used for PDFs; multi-dimensional Gaussian models, K-nearest-neighbor approximations, histograms, and neural nets, just to name a few [16]. These representations vary in computational efficiency and in the kinds of PDFs that can be represented. Each of these representations has advantages and disadvantages in training time, representational power, lookup time, and storage space. For reasons explained below, we used a 3-dimensional histogram in RGB; 5 bits for each of red, blue, and green.

Producing the histograms requires an independent estimation of probability for each of 32x32x32 = 32768 different discretized bins; it thus requires over 32000 parameters to describe completely. Compared to a multi-Gaussian representation, this may seem excessive; since there are so many free parameters, many more training samples are required in order to form a reasonable PDF. In our application, however, training data is plentiful, since every image pixel represents a training point (we will discuss how these training points are labeled shortly). Further, updating the PDF with a new training point is quite rapid, since it simply requires incrementing a single counter; this is not the case with, for example, a neural net representation. The large number of free parameters allows the histogram to represent a wider range of density shapes than is possible with a multi-Gaussian PDF, such as used by Crisman [5] in her road-following models. Computing probability densities from the model is computationally fast when compared to any of the alternatives (particularly to K-nearest neighbor); a single table lookup followed by a single division operation produces the required result.

Acquiring training data

While it is conceivable that training data could be collected by hand-labeling individual image pixels as either “processed” or “unprocessed”, this is undesirable for several reasons. The process would be tedious; it would be difficult to decide what a “representative” training set looks like; and adapting to a new environment would require a hand-marking a new data set. A more practical method is to use the step function fitting method described in Sec-

A Bayesian technique can be developed for segmenting scan lines which avoids these limitations. Instead, it requires an estimate of the pdf (probability density function) for the processed and unprocessed areas. The pdfs exist in the space of all the possible discriminants to be considered; for instance, if a color-based system is used, the pdfs would be functions of R, G, and B. We first discuss how these pdfs can be used to segment a scan line; later we will discuss ways of constructing these pdfs from training data; and finally we discuss how the best-fit step function method described in the previous section can be used to produce training data.

Segmenting a scan line

Suppose we are given two pdfs, $p_{processed}(f(i, j))$ and $p_{unprocessed}(f(i, j))$. We model the scan line data as a sequence of j_d points drawn from one distribution followed by a sequence of random points j_{last}-j_d drawn from the other distribution. If we allow j_d to vary between 0 and jlast, there are then $2*j_{last}$ models to consider for any particular scan line: j_{last} models in which the first distribution is the processed distribution and the second distribution is the unprocessed one, and j_{last} models in which the first distribution is unprocessed and the second one is processed.

We can then estimate the location of the boundary by computing the maximum likelihood estimate over all these $2*j_{last}$ models, and using the j_d which corresponds to that model. Computing the maximum likelihood estimate is accomplished by means of the following equation:

$$Max\left(\begin{array}{l} Max\left(\prod_{j=0}^{j_d} p(processed|f(i,j)) \cdot \prod_{j=j_d+1}^{j_{max}} p(unprocessed|f(i,j))\right), \\ Max\left(\prod_{j=0}^{j_d} p(unprocessed|f(i,j)) \cdot \prod_{j=j_d+1}^{j_{max}} p(processed|f(i,j))\right) \end{array}\right)$$

in the linear function which most cleanly separates the processed and unprocessed pixel classes. After each image is processed, the algorithm computes the Fisher linear discriminant in RGB space between the cut and uncut pixel classes; this becomes the discriminant used for the next image.The discriminant function used for the first image is chosen arbitrarily; as shown in the convergence experiment in Section 4.3, however, the method converges to a good discriminant quite rapidly regardless of the initial choice.

3.1.2 Boundary sensing using a Bayesian approach

Finding the best-fit step function to some discriminant function is one way of estimating the boundary point on a particular scan line. It does, however, have some potential problems. It is biased towards the middle of the image; that is, for a data set in which the discriminant is generated randomly, the step function method will more often return boundary points in the middle of the scan line than at the edges. Also, it relies on an assumption that there exists a discriminant between the processed and unprocessed areas such that the interclass distance between processed and unprocessed is greater than the intraclass variation. As shown in Figure 3-10, however, some multi-modal distributions may simply not possess good linear discriminants.

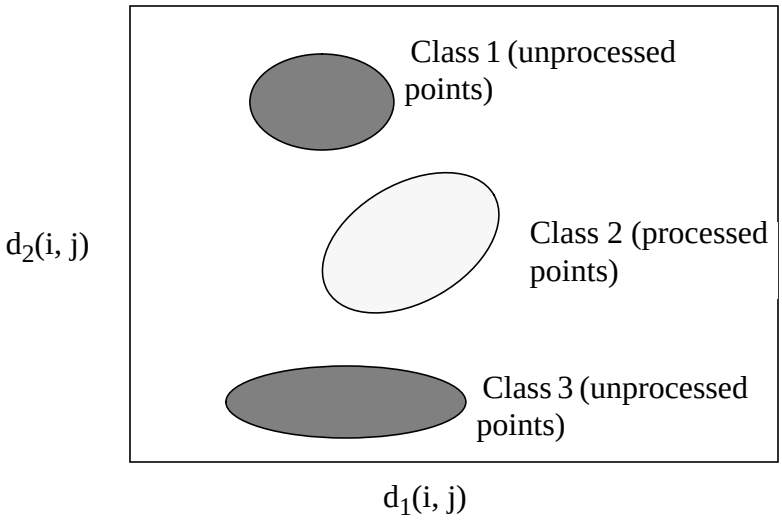


Figure 3-10: A multi-modal distribution.

erally be unprocessed crop, and points in region 2 will generally be processed. These regions can then be used as a source of training points.

A single camera system has a few advantages over the two camera system described above. In addition to cost benefits, the training data from the single camera avoids some of the potential problems of the two camera system; for example, there is no longer a need to worry about possible lighting differences between the two cameras. However, since the shape of the crop line in the one-camera setup is not known a priori, there is a risk of misclassifying training data. In fact, as will be discussed in Section 4.3, all of the methods we have developed are largely insensitive to misclassified training data; we therefore abandoned the two-camera approach at a very early stage.

Adaptation using the Fisher linear discriminant

One adaptive method we developed for the step-function-fitting algorithm computes a measure known as the Fisher linear discriminant [6]. The Fisher discriminant computes the line

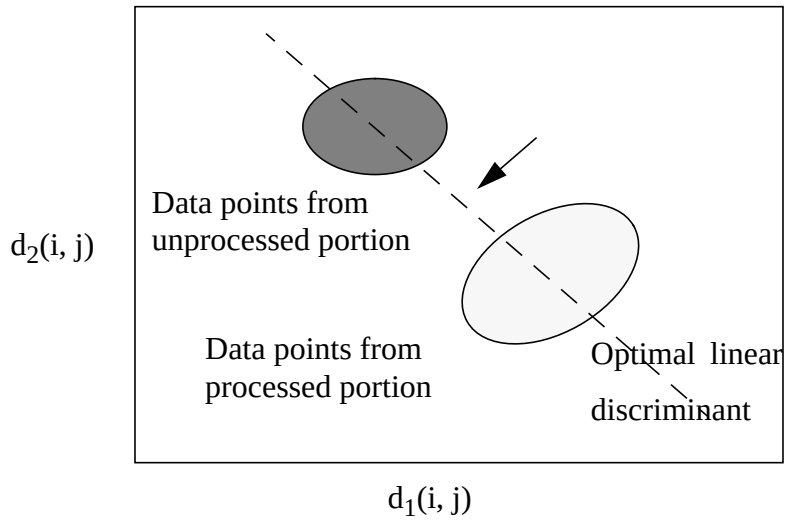


Figure 3-9: The Fisher linear discriminant.

in the discriminant space (for the alfalfa harvesting application, this is simply RGB space) such that when the pixel values are projected onto that line, the ratio of average interclass distance to average intraclass scatter is maximized (see Figure 3-9). Intuitively, this results

Optimally, the training data from camera B should be representative of the image points viewed by camera A. There are a number of reasons why this might not be the case with the setup described above. Since cameras A and B are oriented differently, the angle and distance from which the surface is viewed will be different. For outdoor robots, the robot may consistently cast shadows on one camera's viewing area but not on the other's. In addition, the processed image points in camera B's image will be viewed moments after the area is processed, whereas those viewed by camera A will be somewhat older.

1-camera method

Suppose instead the harvester is outfitted with only one camera as in Figure 3-7:

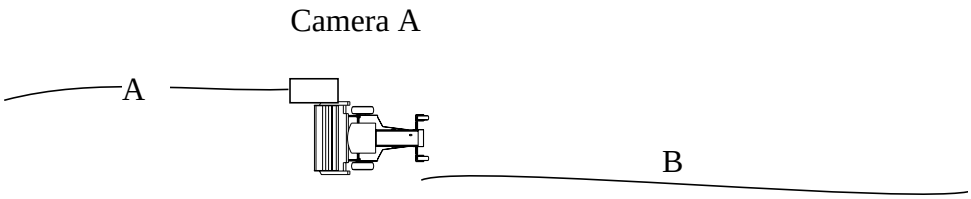


Figure 3-7: Collecting training data with 1 camera.

As long as the boundary curvature is not too large, most of the time crop line B and crop line A will be close to parallel. To the extent that this hypothesis is true, camera A's images are constrained to the form shown in Figure 3-8. This means that points in region 1 will gen-

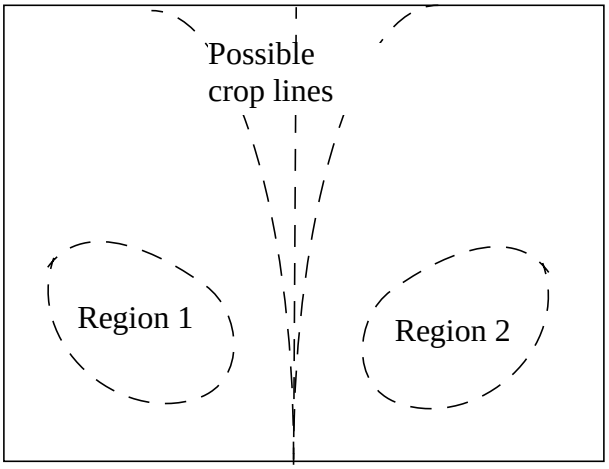


Figure 3-8: Unknown segmentation

2-camera method

Imagine the harvester in the above figure has been outfitted with two cameras, as shown in Figure 3-5:

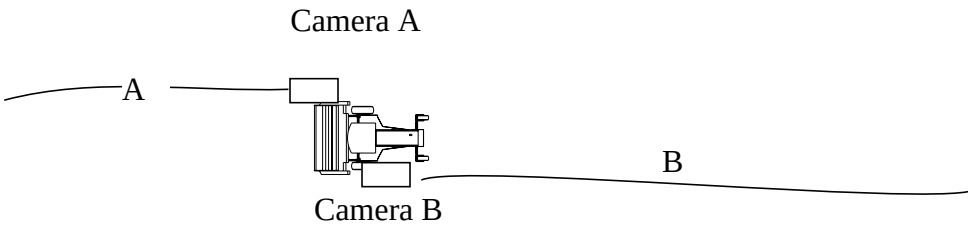


Figure 3-5: Collecting training data with 2 cameras.

Camera A looks out along crop line A; camera B looks directly down at crop line B. As long as the robot is actually processing some area, the correct segmentation of the image which camera B sees will be known quite precisely. For example, if camera B is centered over boundary B with the image rows parallel to the cutting tool, the image segmentation should appear as in Figure 3-6:

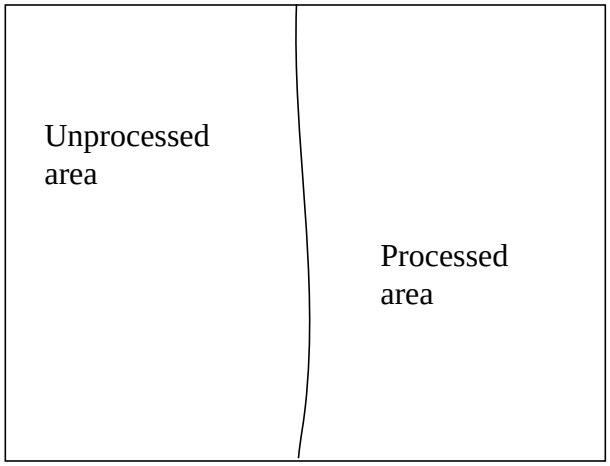


Figure 3-6: A priori known segmentation.

These image points from camera B can then be used as training data.

Figure 3-4 shows the same two images with a simple color-based discriminant measuring the ratio of red to green:

Figure 3-4: Identical images with a discriminant of red/green

This provides some evidence that there is no single, all-around “best” discriminant for alfalfa harvesting; rather, the best discriminant varies with the local environment. Thus, in the harvesting case (and likely in other surface covering tasks), a means of dynamically tuning the boundary following algorithm to adapt to local conditions could improve the robustness of the system. The specific tuning parameter will vary with the application; for a color based system like the harvester, the color discriminant would be adapted, while for a texture based system, the texture operator would be adapted.

Training data for an adaptive discriminant

We have developed a range of techniques for adapting the discriminant to the local environment in real time. These techniques depend on training data, which is used to provide labeled examples (either “processed” or “unprocessed”) of image points to the system. These examples will then be used to modify the discriminant of the boundary following algorithm. Two ways of collecting such training data are given below.

While such statistics are somewhat removed from true performance measures of the system, they do provide some heuristic information.

Use of an adaptive discriminant

Boundary sensing is inherently a local sensing problem, and the results for any static algorithm are likely to vary with local variations in the environment. While it may be possible to tune a static algorithm to work well within a narrow range of conditions, an adaptive mechanism can help provide robust performance across the entire set of conditions under which the robot may be expected to perform.

Experiments using the boundary sensing method on the Demeter automated harvester bear this out. As discussed in Section 3.1.1, it performs a segmentation by means of a color-based discriminant function. The specific color discriminant which works best, however, can vary significantly even within the same crop field. Figure 3-3 shows two images from the same crop field using an intensity-based discriminant:

Figure 3-3: Two images processed with an intensity-based discriminant

in the discriminant between the processed and unprocessed side to be as large as possible compared to the standard deviation (S.D.) of the discriminant within a given side. For the alfalfa harvesting application, we investigated a some discriminants that could be used with an RGB camera, as well as some which could be used with a custom-filtered camera. Table 3-1 displays standard deviations and differences between the mean values for several different discriminants from the RGB data, and Table 3-2 displays the same statistics for some of the best discriminants from the bandpass filter data. The filter data suggest that using a custom-built camera sensitive to only a few narrow frequency bands may provide an advantage over an RGB camera, if the frequencies are customized for a particular crop. (These data were calculated from images taken from an alfalfa field in El Centro, CA).

Table 3-1: RGB discriminants

Discriminant	*cut S.D.*	*uncut S.D.*	*uncut mean - cut mean*
R	22.5	16.8	37.2
G	23.7	20.1	27.9
B	24.3	19.5	31.0
R/(R+G+B)	0.020	0.013	0.033
G/(R+G+B)	0.018	0.007	-0.020
R/G	0.096	0.043	0.137

Table 3-2: Bandpass filter discriminants

Discriminant	*cut S.D.*	*uncut S.D.*	*uncut mean - cut mean*
650 nm	22.2	8.22	59.1
650 nm/750 nm	0.188	0.040	0.493
550 nm/750 nm	0.112	0.057	0.357

$$error(j_d) = \frac{\sqrt{\left(t2_l(j_d) - \left(-\frac{[t_l(j_d)]^2}{j_d + 1}\right)\right) + \left(t2_r(j_d) - \left(-\frac{[t_r(j_d)]^2}{j_{max} - j_d}\right)\right)}}{j_{max} + 1} \tag{3-5}$$

Since Eqns (3-1) through (3-4) can be computed recursively, it takes only a constant number of operations to compute t_l, t_r, $t2_l$, $t2_r$, and error at j_d+1 given their values at j_d. The end result is that the entire algorithm requires only order (j_{max} - j_{min}) time to find the best fit step function for a given scan line.

We can make the computation faster still by computing, instead of error(j_d), some monotonic 1-to-1 function of error(j_d) such as the following:

$$f(j_d) = [error(j_d) \cdot (j_{max} + 1)]^2 - \sum_{j=0}^{j_{max}} d(i, j) \tag{3-6}$$

which can be computed recursively using the following three equations:

$$\begin{aligned} t_l(0) &= d(i, 0) \\ t_l(j_d) &= t_l(j_d - 1) + d(i, j_d) \end{aligned} \tag{3-7}$$

$$\begin{aligned} t_r(0) &= \sum_{j=1}^{j_{max}} d(i, j) \\ t_r(j_d) &= t_r(j_d - 1) - d(i, j_d) \end{aligned} \tag{3-8}$$

$$f(j_d) = \frac{-[t_l(j_d)]^2}{j_d + 1} + \frac{-[t_r(j_d)]^2}{j_{max} - j_d} \tag{3-9}$$

This additional speedup results in a roughly 30% increase in cycle rate compared to computing error(j_d) directly from Eqn. (3-5).

Choosing a discriminant

There are many ways one might attempt to evaluate the relative merit of discriminant functions. In order for the segmentor algorithm to work reliably, it is desirable for the difference

$$m_l(j_d) = \frac{\sum_{j=0}^{j_d} d(i,j)}{j_d + 1}$$

$$m_r(j_d) = \frac{\sum_{j=j_d+1}^{j_{max}} d(i,j)}{j_{max} - j_d}$$

$$error = \frac{\sqrt{\sum_{j=0}^{j_d} [d(i,j) - m_l]^2 + \sum_{j=j_d+1}^{j_{max}} [d(i,j) - m_r]^2}}{j_{max} + 1}$$

Clearly, it requires order n time to compute error(j_d) from the d(i,j) alone. It is possible, however, to compute error(j_d + 1) from error(j_d) in constant time. To accomplish this, we express the calculation of error(j_d) in terms of the following sums:

$$t_l(j_d) = \sum_{j=0}^{j_d} d(i,j) \tag{3-1}$$

$$t_r(j_d) = \sum_{j=j_d+1}^{j_{max}} d(i,j) \tag{3-2}$$

$$t2_l(j_d) = \sum_{i=0}^{j_d} [d(i,j)]^2 \tag{3-3}$$

$$t2_r(j_d) = \sum_{j=j_d+1}^{j_{max}} [d(i,j)]^2 \tag{3-4}$$

From these, error(j_d) is calculated as follows:

The step function is defined by three parameters; j_d, the j coordinate of the discontinuity; m_l, the mean value of the step function to the left of j_d; and m_r, mean value of the step function to the right of j_d. Fortunately, it is not necessary to exhaustively search the full three dimensional space to find the best fit; given j_d, it is possible to rapidly compute m_l and m_r. In particular, m_l is merely the average of the d(i,j) on the left side of j_d, and m_r is the average of the d(i,j) on the right.

Thus, an exhaustive search is only necessary across the 1-dimensional space of j_d. The starting point for our algorithm is thus the following:

smallest_error = infinity

for each possible j_d *from* j_{min} *to* j_{max}

  compute m_l

  compute m_r

  compute error

  if (error < smallest_error)

    smallest_error = error

    *best_*j_d *=* j_d

  endif

endfor

The bulk of the computing time for this algorithm comes from computing m_l, m_r, and the error. Computing these means and errors for the first j_d takes order (j_{max} - j_{min}) time. However, as presented below, it requires only a small constant number of operations to recompute m_l, m_r, and error for subsequent values of j_d.

Let j_d be the rightmost column to the left of the discontinuity in the step function, and suppose that the column numbers vary from 0 to j_{max}. Then the m_l, m_r, and error terms can be calculated as functions of j_d as follows (these are defined for j_d from 0 to j_{max} - 1):

3.1.1 Boundary sensing by step function fitting

We developed our first algorithm by making a series of simple assumptions about the nature of the boundary. We assume, for instance, that this boundary divides the image into precisely two connected regions. We also assume that the boundary between these regions is a single-valued function of the row coordinate, and that this boundary does not intersect either the left or right edge of the image. This boundary function is represented explicitly by the set of pixels which lie on it, so further smoothness or shape assumptions are unnecessary. Finally, assume that our discriminant is basically bimodal; that is, that for the part of the image which images the processed area, the d(i,j) will be clustered around some mean value m_p, and the d(i,j) will be clustered around some different mean m_u for the unprocessed area.

The set of assumptions described above lends itself naturally to a scan-line based algorithm, since each image scan line is responsible for contributing one pixel to the segmentation boundary. Consider a plot of the discriminant d(i,j) of the data along a single scan line (or image row) i. Under these assumptions, the data can be roughly fitted with a step function as shown in Figure 3-2.

Finding the best segmentation is then a matter of finding the best fit step function (lowest least-squared error) to d(i,j) along a given scan line. As we will show, using this definition of best-fit allows us to compute this step function rapidly.

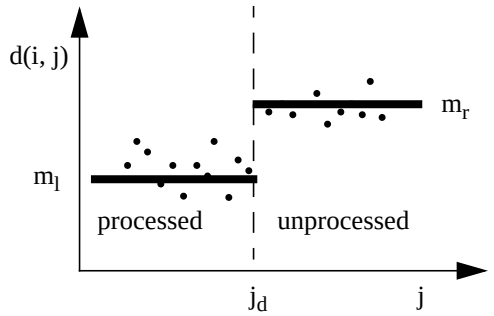


Figure 3-2: A model plot of d(i, j) as a function of j for a single scan line i.

Figure 3-1: Processed/unprocessed boundaries.

be used. Once the function (or functions) are selected, however, the problem becomes much more general:

> **Given: A set of image pixel functions $i, j \Rightarrow \Re$ whose output is related to whether or not the area corresponding to the pixel has been processed.**
>
> **Goal: Robustly detect the boundary between the processed and unprocessed areas.**

This formulation of the problem breaks neatly into two pieces; the selection of an appropriate discriminant function (or set of functions), which is application-specific; and the processing of that discriminant to return a segmentation, which is general across applications.

3. Methods for surface coverage perception

Chapter 2 motivated the need for several some solutions to general surface-covering perception problems. This chapter discusses algorithms we have developed to meet these needs.

In Section 3.1, three different techniques for boundary tracking are presented: a method based on step-function fitting, a method based on Bayesian techniques, and a method based on curvature hypothesis. Section 3.2 discusses a method for detecting the end of a processing row, while Section 3.3 discusses a general method for detecting obstacles in surface covering tasks. Section 3.4 is dedicated to the problem of shadow detection and compensation in outdoor environments. While this last topic is not central to surface covering, it proved necessary to address in order to produce a working demonstration of our techniques for use in alfalfa harvesting.

3.1 Boundary tracking

As discussed in Chapter 2, the goal of a surface covering machine is to cover an entire surface without leaving gaps while minimizing overlap. Doing this effectively requires precise knowledge of the position of the machine relative to the processed/unprocessed boundary. In this section, we discuss a general vision-based method for detecting this boundary.

Figure 3-1 shows work edge boundaries from three quite different surface coverage tasks: alfalfa harvesting, trash maintenance, and snow plowing. Clearly, at some level, a vision-based boundary sensing system will be dependent on the type of boundary created, and thus on the specific surface covering application. A snow removal system might use an image intensity function as a means of discriminating between processed and unprocessed areas, while a lawn mower might use a texture function. Possibly, several different functions could

addressed by the development of some general perception capabilities: boundary tracking, out-of-bounds detection, obstacle detection, and surface processing quality monitoring are four examples. While there is a variety of previous work with relevance to these tasks, much of it makes use of simplifying assumptions which often fail in the real world; the development of methods which avoid this problem while still making use of domain knowledge common across SCTs could contribute to the automation of a variety of applications. Since the practical contribution of such methods depends on their robustness in real-world use, results are best validated by actual systems which demonstrably perform in a range of environments.

sufficient for the job. However, the inefficiency of such a random strategy prevents it from being suitable for tasks in which machine operation is a substantial expense.

One simple alternative is to lay down some sort of track along the surface in a pattern which will cover it, and design the processing machine to run along the tracks. This eliminates the inefficiency of random machines, but at the expense of laying down (and maintaining) the tracks. Similarly, beacon-based guidance schemes require maintaining infrastructure as well.

2.4.4 Goals for this thesis

From the survey above, it is possible to identify some areas of weakness which are common across much of the previous work in the field. These weaknesses are specifically targeted in this thesis.

A lack of closed-loop real-world testing.

It's often difficult to distinguish practical solutions from impractical ones on the basis of theory alone. To be fully validated, techniques should be instantiated in closed-loop real-time systems which are demonstrably robust across a variety of conditions.

Inappropriate assumptions about the environment.

The boundaries created during SCTs are not always straight lines or constant curvature arcs, and methods which make this assumption are likely to be brittle in the real world. At the same time, the kind of a priori knowledge discussed in Section 2.3 should be exploited.

A failure to generalize methods beyond a narrow range of applications.

An effort should be made to distinguish those components of a system which are applicable to a broad class of tasks from those components which are highly application-specific. This speeds the development of other automated systems and increases the relevance of the work to the general robotics community.

2.5 Summary

Surface covering tasks (SCTs) form a class whose automation is of both commercial and theoretical interest. These tasks share a number of needs, several of which could be

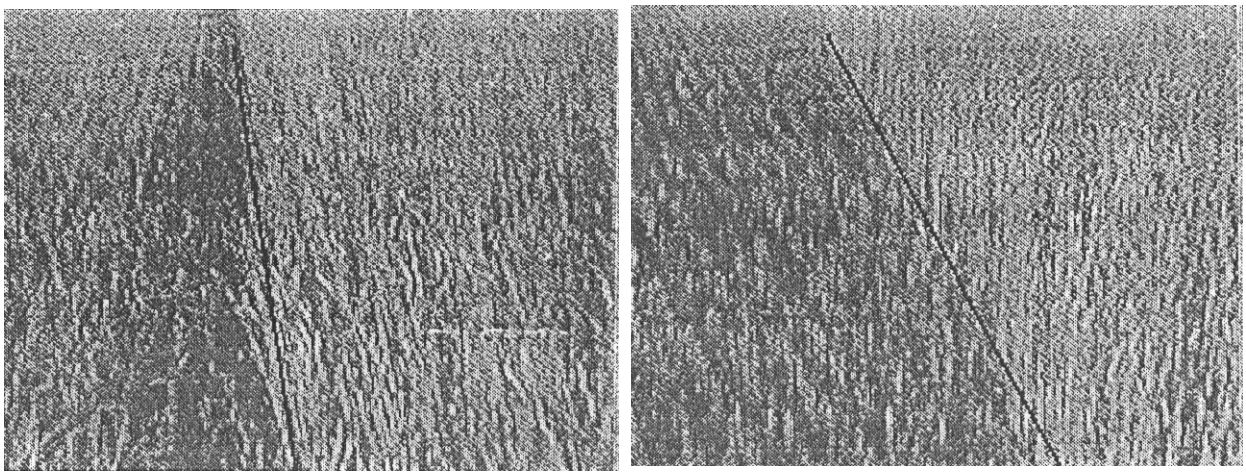


Figure 2-4: Results from Klassen et al. [14] on unspecified crop

Such techniques are, however, optimized to take advantage of a different (and much more uncertain) domain. In surface covering applications, the important feature is known to be the processed/unprocessed boundary; it is known to be continuous (assuming the machine is functioning properly), though it may be quite faint. Systems like ALVINN and RALPH are designed to work in a domain where, at any given time, there are a variety of transient but prominent features (painted road lines, guiderails, etc.) which can be tracked; their biggest asset is their robustness to the disappearance of any single feature, which is much less important in boundary following.

A further parallel between boundary tracking and road following is that both occur outdoors, in natural lighting. The two applications thus share many of the same needs to adapt to lighting changes; and the methods we use in this thesis are similar in many ways to those used by Wallace [30] and Crisman [5].

2.4.3 Other techniques

Systems which automatically clean concrete swimming pool floors have been in use for many years. Their navigation systems are simple; they meander randomly across the surface. They are cheap, reliable, and cover most of the surface in a short enough time to be

Gerrish et al. [8] used a variety of edge-detection and template-matching techniques to pick

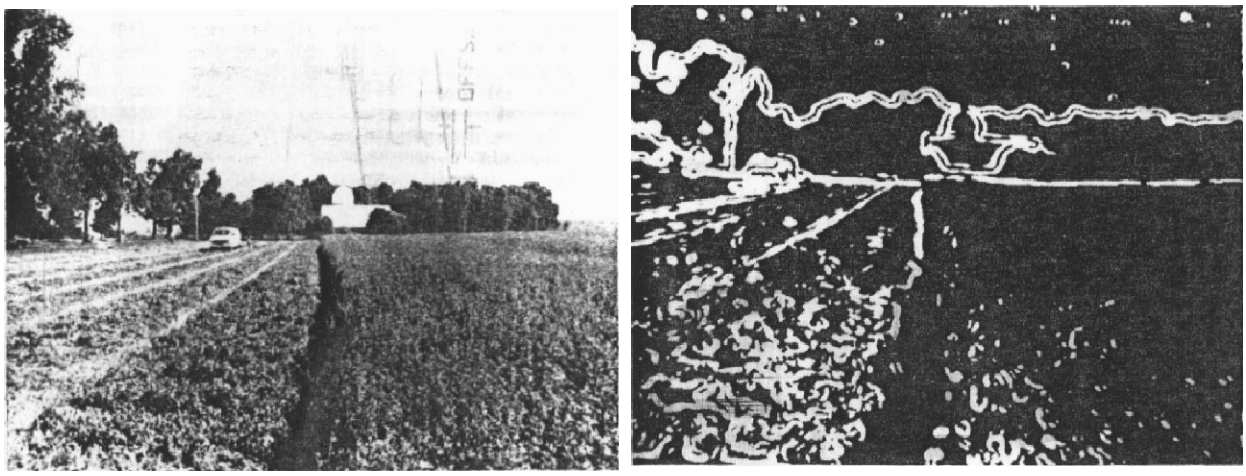


Figure 2-3: Alfalfa results from Gerrish et al. [8] (unprocessed on left, processed on right)

out “work edges”; these were tested on actual field images (including alfalfa). Results, however, were noisy, as shown in Figure 2-3. Their algorithm required roughly 20 seconds using a 68000 processor.

Klassen et. al. [14] present an algorithm for finding the boundary between tilled and untilled soil and cut and uncut crop using a black and white camera. As shown in Figure 2-4, their algorithm is limited to straight lines, and runs at approximately 4 Hz on an 80386 processor. Their algorithm works by finding a boundary point in several image bands and fitting them with a straight line; their methods were not used to guide a vehicle since their stated goal is to provide information back to the operator.

Jahns [12] presents a (now somewhat dated) review of potential automatic guidance techniques for agricultural vehicles.

2.4.2 Road following research

Boundary tracking has many similarities to road following, and several road following techniques could be directly applicable to boundary tracking; ALVINN [20] , RALPH [21], and YARF [15] are two such examples. In fact, RALPH was the inspiration for one of our boundary tracking techniques described in Section 3.1.

this thesis; here, we will concentrate only on how characteristics of SCTs can be exploited to aid in this task.)

2.4.1 Agricultural and lawn mowing research

Hayashi & Fujii [9] have used smoothing, edge detection, and a Hough transform with a black and white camera to guide a lawn mower along a cut/uncut boundary. Their system had a cycle time of 1-2 Hz, and returned only straight line boundaries; no reliability measures are reported, and it's not clear whether this method was ever used to guide an actual vehicle.Chateau et al. [4] use a black and white camera along with several texture and intensity operators to find a boundary between cut and uncut crop; their algorithm finds vertical straight boundaries only, and accuracy figures are not given. Billingsley & Schoenfisch [1] developed a vision system for steering a tractor in crop rows. Their algorithm finds straight lines only. Accuracy measures are reported for two simulated crops; no results for actual crop images are reported. Frame rates of up to 10 Hz are reported using a 80486 machine for analysis.

A number of researchers have investigated guidance sensors for distinguishing crop from soil. Fehr and Gerrish [7] have used intensity and color to guide an actual tractor through a real corn crop through several 50m long rows. Brandon & Searcy [2] use threshholding, run-length encoding, and curve fitting to find crop canopy rows. Their algorithm was tested on a black and white prerecorded video. A cycle time of 0.9 seconds on a PDP-11 was reported. No image results are shown, although they report that "Studies with a limited number of tractor operators indicated that the accuracy of the algorithm was similar to that of humans". The system was not used to control an actual vehicle.

Reid & Searcy [23][25] describe a method of segmenting crop canopies from soil by intensity threshholding; reported processing times are approximately 0.3 seconds/image on a PDP-11. They have also performed some experiments on finding canopy edges with a Hough transform, though no computing or timing data were provided [24]. Some promising results are shown on images of a cotton field, though again, their algorithms have not been used to guide a vehicle.

This fact can be used to provide a priori knowledge for perception tasks; for example, the produced boundaries between the processed and unprocessed surface are continuous and at roughly right angles to the out-of-bounds border. For a given mode of perception, additional information may also be available; imagine, for instance, that a camera has been placed on the vehicle shown in Figure 2-1, pointing forwards along the path of the vehicle and slightly downwards. In the two states shown, this sensor will return the kinds of views shown in Figure 2-2.

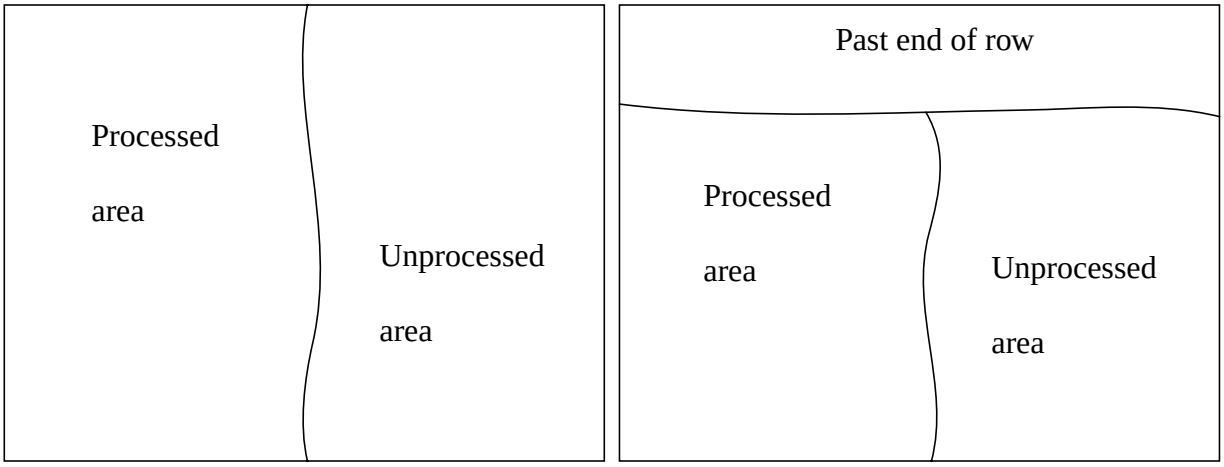


Figure 2-2: Sensor data from common states.

These two view types are therefore quite likely to be encountered by in a typical SCT. Interestingly, none of this knowledge depends on which particular SCT being performed; it follows merely from the need to efficiently cover a surface, and is therefore quite general. Chapter 3 covers ways of exploiting this domain knowledge to provide some of the capabilities described in Section 2.2.

2.4 Relevant previous work

Outside of the planning domain, the general concept of surface covering machines have received little attention in the robotics community. Thus, most of the previous work relevant to perception has been on specific systems, particularly boundary following for agricultural applications and lawn mowing. (There is, of course, also an extensive body of work on obstacle detection for mobile robots. A survey of all such methods is beyond the scope of

effective processed/unprocessed boundary. In such cases, being able to sense the boundary can ensure that the area is covered efficiently without leaving unprocessed area behind.

- In some environments, GPS may not always be a feasible candidate for a global sensor. Two examples might be snow plowing, where the robot needs to function reliably even under adverse weather conditions; and indoor floor care applications, such as vacuuming or mopping.

- GPS still has some accuracy and reliability problems, particularly in sensing orientation. The high-end Novatel differential GPS units on board the Demeter harvester, for example, typically return heading errors of around 5 degrees with occasional forays into the 15 degree range (this is while moving at typical harvesting speeds of ~5mph). The video camera, however, has an angular resolution of less than a tenth of a degree in the center of the image. Even including a typical boundary sensing errors of 8 pixels or so (see Section 4.3), the camera is still able to consistently register heading with respect to the boundary within less than 1 degree without any dependence on the speed of the vehicle.

2.3 Shared domain knowledge

Fortunately, surface covering problems share more than just common needs. SCTs are typically processed in a row-by-row fashion, because this is an efficient method for covering areas with minimal overlap. One result of this row-by-row processing is that the covering machine is likely to spend the majority of its time in one of the two states shown in Figure 2-1.

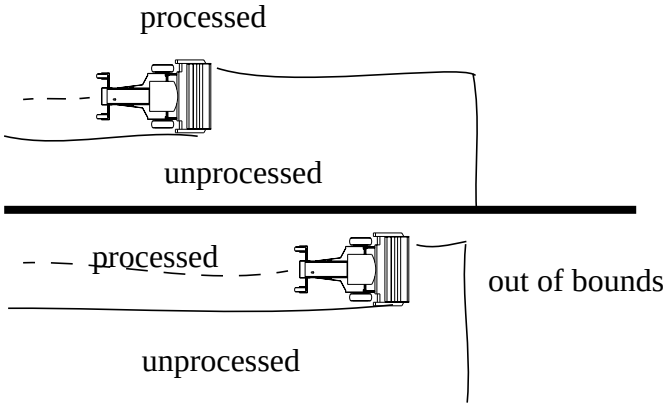


Figure 2-1: Common states of a surface covering machine.

2.2.3 Detecting obstacles

Many SCTs take place in environments which are typically obstacle-free, such as agricultural fields. However, an autonomous vehicle performing all these tasks will still require some type of perception-based obstacle detection in order to avoid injuring humans and damaging the machine itself.

2.2.4 Monitoring processed areas for quality of processing

Lawn mower blades jam, and snow plow heights require adjustment to make sure the snow is properly removed. Duplicating this monitoring capability would have a clear use for autonomous vehicles performing SCTs. While many performance errors can be diagnosed internally, the surest method of verifying that the entire surface-covering function is being properly performed is by sensing the actual state of the surface.

Local vs. global positioning

A number of the positioning tasks desccd ..ribed above could potentially be accomplished purely with a global position sensor such as GPS (the Global Positioning Satellite system), if the area has been previously surveyed. Using local perception, however, has a number of advantages, as described below:

- A global sensor cannot always be placed on the part of the robot which needs to be positioned precisely. For example, on the Demeter automated harvester, the GPS unit is placed on top of the cab to ensure a line of sight to the maximum number of satellites. When tracking a boundary, however, the important thing to know is the position of the cutting head relative to the crop, both of which can be easily viewed from a single camera. On a completely rigid robot on a flat surface, the position of the cutting head can be inferred; small changes in the pitch of the terrain, however, will move the cab roof several inches without moving the cutting head at all.

- The ability to directly sense the boundary rather than inferring it can provide additional flexibility. For example, a GPS guided harvesting machine might work well as the sole vehicle in a carefully surveyed field, but a visually guided machine could be just as easily employed in an entirely new field which has already been partially harvested and in which other non-automated machines continue to work.

- In some applications, the location of the boundary may depend on more than just the path swept out by the robot. For example, in a snow plowing or soil removal application, material from the unprocessed area may spill over onto the processed side, moving the

wide variety of safety issues. Many of the current difficulties of these systems can be traced to the fact that the difference between such domains and the predictable polyhedral world of Nilsson [17] is so large that techniques which are successful in one type of domain often fail badly in the other.

Surface covering tasks generally take place in environments which fall between these two extremes. A typical SCT machines operates in low-traffic conditions within strictly defined areas, with extensive a priori knowledge about what is likely to be found in the environment. Automation of SCTs is thus a logical next step towards the development of robots which work in totally unstructured environments, while simultaneously providing direct practical benefits for real world problems.

2.2 Shared needs of surface covering tasks

The surface-covering problem has been recognized in planning literature for some time [3], where it is often referred to as the lawn-mowing problem. The name, while descriptive, is overly limiting; besides lawn mowing, other SCTs such as floor cleaning, agricultural harvesting, painting, and snow plowing can all require similar kinds of planning. In this section, we consider some of the other common capabilities which would be required to automate these kinds of tasks.

2.2.1 Precise positioning allowing minimum overlap processing without leaving gaps.

The goal here is fairly straightforward; the machine must be controlled in such a way that minimizes overlap with previously covered areas for efficiency reasons, and yet must not leave any part of the surface uncovered. The ability to accurately determine the relative location of the vehicle to the boundary between the processed and unprocessed surface would be helpful here.

2.2.2 Detecting the borders of the work space.

As was discussed previously, most SCTs take place within well defined, bounded regions; lawn mowers need to stay on the grass, and alfalfa harvesters need to stay in the field. A high precision estimate of the location of this region border to the vehicle is necessary to ensure that the surface gets processed just up to, but not past, this border.

2. Motivation for surface coverage perception

Chapter 1 introduced the concept of surface coverage tasks (SCTs). Chapter 2 motivates the need for the work in this thesis: the development of a set of common perception capabilities for SCTs. Section 2.1 presents some practical and theoretical reasons for interest in SCT automation. Section 2.2 discusses some of the shared needs of SCTs, and how they can be addressed by local perception. Section 2.3 discusses domain knowledge which is common across these tasks; in conjunction with Section 2.2 it provides support for seeking solutions which generalize across SCTs instead of developing separate methods for each application. Section 2.4 surveys relevant previous work, summarizes some of the shortcomings, and translates them into goals for this thesis.

2.1 Benefits of surface coverage automation

A sense of the potential economic gain from automating SCTs can be inferred from the resources currently dedicated to these tasks. In the United States in 1994, over $7.4 billion was spent on lawn care; hay sales exceeded $2.9 billion [28]. Even small increases in efficiency are therefore worth investigating, and in fact in some of these applications, the current limiting factor on task performance is known to be the human operator. For instance, hay cutters have been designed which can function at higher speeds than the current standard of 4-5 miles an hour, but the manufacturer found that at these speeds, humans had trouble guiding the machines precisely for long periods of time. Automation which either aids or replaces the human operator could addresses this bottleneck while reducing safety concerns and maintenance costs. Further, the per unit equipment cost for many of these machines is high; the extra cost for the computing and sensing necessary for automation could therefore translate into only a small percentage increase in price.

SCT automation also embodies a number of broader issues in mobile robotics. Early mobile robots typically dealt with known, structured environments; indoor rooms with polyhedral obstacles, for example [17]. Early successes with such systems have resulted in a number of more recent attempts to operate in unknown, complex environments; for instance, automated driving systems must contend with a changing roads, other human drivers, and a

RALPH [20], and a method derived from Bayesian statistics. Using examples from an alfalfa harvesting application, we discuss the advantages of adaptive algorithms over static ones and discuss several methods for implementing an adaptive capability. A partial solution is offered for detecting and compensating for shadows which would otherwise disrupt the algorithms.

Chapter 4 is devoted to results from the alfalfa harvesting domain, which is the main test application for this research. The Demeter automated harvester is described, and the integration of perception into the robot architecture is discussed. Field results are presented; at the time of this writing, over 80 acres of alfalfa were harvested autonomously by vision-based guidance in fields in Kansas, Pennsylvania, and California. In addition to observations from the field trials, several more controlled experiments on image sequences are presented; these experiments attempt to quantify the relative performance of some different algorithms, determine how much of an advantage is gained by using adaptive over non-adaptive perception, and investigate issues of possible instability in the learning algorithms.

In Chapter 5 we explore two other surface-covering applications besides alfalfa harvesting. In order to emphasize the generality of our methods, we deliberately selected applications outside of the agricultural arena, and ones for which color is no longer a key discrimination feature.The first application is clearing snow from a large area, such as a parking lot or an airport runway; the second is trash compaction at a dumpsite. Given the lack of an appropriate testbed such as the Demeter harvester, the investigation is limited to processing images from these environments. We discuss how the perceptive capabilities of our boundary tracking algorithm can be altered with simple geometric transforms, as demonstrated on some images from the road-following domain.

Finally, Chapters 6 and 7 discuss conclusions and contributions from this work as well as some avenues for future research.

one application-- boundary sensing for alfalfa harvesting- and then investigated in lesser depth:

- other perception problems (end-of-row detection, obstacle detection) for the harvesting application; and
- boundary sensing for other applications (snow removal, trash compaction)

Thus, the boundary following task for the alfalfa harvesting application forms the core of this thesis work, and we extend this core along several orthogonal axes: integration into an functioning robot system, development of other perception capabilities, and exploration of other surface-covering applications. In this way we can demonstrate both the practicality of our approach on an actual robot accomplishing a real task while also demonstrating the generality to other applications.

This reasoning explains much of the structure of this thesis. While the topic is perception for surface covering robots, not all aspects will be treated equally; there will be more discussion of boundary following than of obstacle detection, and more discussion of alfalfa harvesting than snow removal.

1.3 Thesis contents

Chapter 2 provides motivation for the work in this thesis. It describes some of the benefits for automating some selected surface covering tasks, such as lawn mowing and snow plowing. Shared needs for boundary tracking, detecting the end of a processing row, and obstacle detection are covered, as is some of the common domain knowledge across these tasks. A discussion is included of the continued need for direct perception despite recent advances in positioning technologies such as GPS. Finally, relevant previous work is presented, along with some implications for our approach.

In Chapter 3, we discuss methods for performing addressing three different surface covering perception tasks: boundary sensing, end-of-row detection, and obstacle detection. The boundary sensing problem is explored in some depth, using three different original algorithms: a simple model-fitting method, a method derived from the road-following system

1. Introduction

1.1 Thesis statement

Many mobile robots have a common set of needs: locomotion, path planning, and obstacle avoidance are common examples. Perception, however, tends to differ widely from one mobile robot to another; an autonomous car might need to sense asphalt roads, while a mail delivering robot might need to sense mail receptacles. Autonomous perception is in general quite difficult, and the need to develop algorithms separately for each application has slowed the advancement of robots into many industries which could benefit from their presence.This limitation could be lessened by the identification of subclasses of mobile robots for which common perception solutions can be developed for several different applications.

Robots whose purpose is to do surface covering tasks (SCTs) are one such subclass. Lawn mowing, snow removal, and alfalfa harvesting are all surface-covering tasks; that is, members of a class of tasks which require a vehicle to be guided over a surface while processing the surface in some way. Many SCTs are good candidates for automation, because they are highly repetitive, take place in at least partially structured environments, and need to be performed on a large scale.

The planning community has recognized for some time that surface covering tasks can be usefully addressed as a general category[3]. Our claim is that the benefit of this categorization is not restricted to planning, but extends to perception as well. In particular, we claim that surface covering tasks share both a common set of perception needs and some common knowledge that can be exploited to meet those needs.

1.2 Philosophy of approach

How are we to demonstrate this claim? The most convincing way would be to develop several different algorithms for doing each of several different shared surface covering perception needs, and use them on several different surface covering machines. Such a project would, unfortunately, require more time and resources than are available for a Ph.D. thesis. Instead, we have investigated in depth several methods for doing one perception task for

Acknowledgements

Thanks first of all to all of the people who worked on the Demeter harvester at one time or another: Mike Blackwell, Kerienn Fitzpatrick, Mike Happold, Regis Hoffman, Alex Lozupone, Ed Mutschler, Tom Pilarski, Henning Pangels, Simon Peffers, Tony Stentz, and Red Whittaker. Without their efforts, this thesis would never have been possible.

I'd like to thank Tony again for his technical guidance, career advice, careful proofreading, and his remarkable patience in his role as my advisor. Thanks as well to the members of my thesis commitee: Martial Hebert, Eric Krotkov, John Reid, Tony Stentz (one more time), and Red Whittaker.

Thanks to friends and housemates for their company, their perspective, and above all their sense of humor during the last five years, particularly to Paula and Geoff Gordon, Tammy Abell, Bert Enderton, Andrew Gove, and Mike Nechyba.

Finally, thanks to my family and most of all to my parents, Dave and Marcy, for their endless love and support.

Abstract

Lawn mowing, snow removal, and alfalfa harvesting are all surface-covering tasks; that is, members of a class of tasks which require a vehicle to be guided over a surface while processing the surface in some way. The planning community has recognized for some time that surface covering tasks can be usefully addressed as a general category. We believe that this viewpoint is useful for perception as well; in particular, we claim that surface covering tasks share both a common set of perception needs and some common domain knowledge that can be exploited to meet those needs.

To substantiate this claim, we have developed methods for addressingthree different surface covering perception tasks: boundary sensing, end-of-row detection, and obstacle detection. The boundary sensing problem is explored in some depth, using three different original algorithms: a simple model-fitting method, a curvature postulation method derived from the road-following system RALPH, and a method based on Bayesian statistics. Using examples from the Demeter automated harvester project, we demonstrate the advantages of adaptive algorithms over static ones and discuss several methods for implementing an adaptive capability. As a sideline, a partial solution is offered for detecting and compensating for shadows which would otherwise disrupt the algorithms.

This work is demonstrated on images from several surface-covering domains: alfalfa harvesting by color segmentation, snow removal, by brightness discrimination, and trash compaction by texture segmentation. The most significant contribution, however, is the development of a commercially viable guidance system for the Demeter harvester; as of this writing, over 80 acres of alfalfa have been harvested autonomously from fields in Kansas, California, and Pennsylvania. New Holland has plans to offer the vision system as part of a commercially available product for sale in the year 2000.

Perception Algorithms for a Harvesting Robot

Mark Ollis

CMU-RI-TR-97-43

Submitted in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

August 1997