Methods for Cropline Following

Michael Happold

CMU-RI-TR-00-14

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

May 2000

Copyright © 2000 by Michael Happold. All rights reserved.

Methods for Cropline Following

Michael Happold

1.0 Introduction

The Demeter Project of the National Robotics Engineering Consortium of Carnegie Mellon University seeks to develop a robotic harvester capable of guiding itself through fields of crop. A variety of techniques for guidance have been experimented with, from GPS to stereo vision. This report details efforts at guidance by means of processing 2-D imagery taken from color cameras positioned on each side of a harvester. These efforts are meant to provide a low-cost and computationally inexpensive solution to the positioning-problem.

Finding the dividing line between cut and uncut crop in 2-D images, the cropline, is naturally formulated as an image segmentation problem. Although extensive work has been done in this area, most segmentation algorithms do not operate in real-time. Real-time in this context means cycling at 4-5 Hz., which is the approximate minimum cycling time to smoothly guide a harvester travelling at 4 m.p.h. Even with the rapid increases in the speed of computing hardware, sophisticated segmentation routines often take several seconds, if not minutes, to complete. Therefore, a suitable trade-off between the reliability of results and the speed of the algorithm must be found.

This report covers four methods for finding croplines. These are: a modified version of the algorithm presented by Ollis and Stentz [13]; a model-based color segmenter; a texture-based segmenter; and a color edge-detector. The speed and accuracy of these four algorithms are compared on images of alfalfa and sudan in flat and bedded fields.

2.0 Prior Work

Hayashi and Fuji [9] proposed differentiating images in the lateral direction to find edges corresponding to the cropline. A Hough transform is then applied to find a straight line through these edge points. This technique is inadequate for images in which cut and uncut crop have similar intensity levels and images in which the cropline is not a straight line.

Billingsley and Schoenfisch [1] introduced a technique for guiding vehicles in row crop by fitting lines to the rows and monitoring the movement of the vanishing point and centers of three of the lines. Although fields with beds often present lines similar to those found in row crops, these lines disappear if the crop is thick, so this technique would not be applicable to this domain.

Gerrish et al. [8] combined edge-detection and template-matching methods to find crop edges in images of fields. They tested their technique on crop images such as alfalfa with mixed results. Too many spurious edges were found, created by shadows as well as the cut crop.

Zhang, Reid, and Noguchi [18] describe a multi-sensor system for guiding an agricultural vehicle in row crops. They use a near infrared filter to distinguish between soil and crop. They provide results for a vehicle guided by vision, GPS, and a fusion of the two. Vision guidance resulted in tracking to within 0.15 m of the desired path. They do not, however, describe the field and crop types in which they tested their vision system, and so it is unclear from this work how applicable an infrared filter is to distinguishing crop from soil in general.

Brandon and Searcy [2] use run length encoding to pick out edges of rows in row crops in thresholded grayscale images. The rows were identified as the centers of the blobs bounded by the edges. This method is designed exclusively for row crops, where the assumption that intensity edges correspond to row boundaries holds.

Ollis and Stentz [12] describe a color discriminant for finding the boundary between cut and uncut crop. This work is superseded by Ollis and Stentz [13] and Pilarksi et al. [14]. Pilarski et al. describe results for both GPS- and vision-guided systems. The vision algorithms described in these papers allow for detection of croplines of arbitrary curvature, an important improvement over techniques relying on hough transforms.

3.0 Problem Definition

Cropline tracking is complicated by the wide variety of crop and field types found in agricultural settings. Crops can be planted on flat fields or on fields with beds. The latter are fields with evenly-spaced, raised mounds. When cutting in fields with beds, the harvester rides with its front wheels on top of the beds. This results in crop of varying heights being left behind between the beds, obscuring the soil from the tracker in the next pass. A tracker that depends on discriminating crop from soil would be hard pressed to find a cropline in such fields.

Crop height and color varies widely as well. Two types of crops were considered in the experiments here: alfalfa and sudan. The color of these crops not only differ from one another, but also from crop of the same type planted in different soil and climate conditions. A further complication is the possibility of crop such as sudan drying out. Dried crop often more closely resembles cut crop or soil than it resembles fresh, uncut crop.

The diversity of crop and field types can be seen in Figures 1-4.

FIGURE 1. Alfalfa on Beds

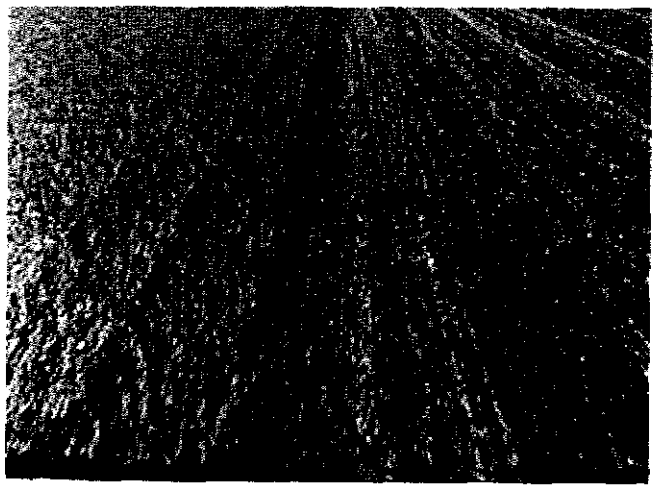


FIGURE 2. Alfalfa on Flats



FIGURE 3. Sudan on Flats

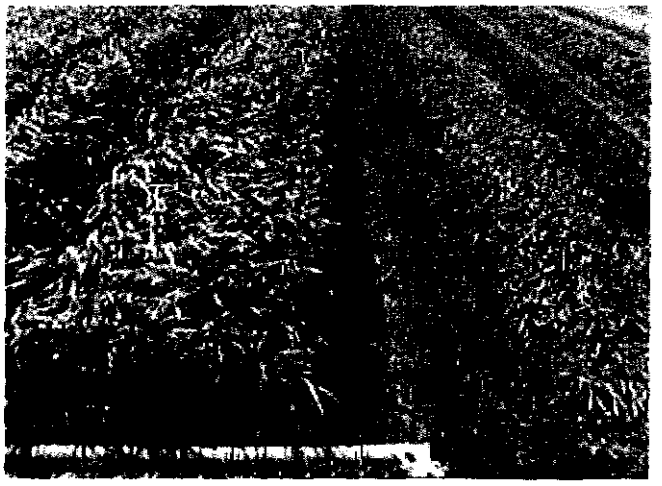


FIGURE 4. Sudan on Beds



4.0 Ollis' Adaptive Algorithm

The algorithm described by Ollis [13] represents the first feasible method for vision-based cropline following. It is both computationally inexpensive, running at 20 Hz on a 200 Mhz. Pentium, and fairly reliable. Ollis reports that when measured against an ideal curve fitted to the cropline, the color adaptive algorithm produces a postulated line with an average error in the position of each point of 7.8 pixels (in images of 640 columns). The algorithm is not sensitive to intensity similarities between uncut and cut crop, unlike thresholding methods described above.

The algorithm seeks a best-fit step in each scanline of an image. The best-fit step is defined as the position in the scanline that minimizes the square root of the sum of the sum of the square differences (SSD) of the pixel values on the left of the position from their mean and the SSD of the pixel values on the right from their mean. This is represented as:

(EQ 1)

$$error(i, j_c) = \frac{\sqrt{\sum_{j=0}^{j_c} [d(i,j) - \mu_{left}]^2 + \sum_{j=j_c+1}^{j_{max}} [d(i,j) - \mu_{right}]^2}}{j_{total}}$$

where j_c is the current column position for scanline i, j_{total} is the total number of columns, and $d(i,j)$ is the adapted value for the pixel at i,j.

On each pass of the algorithm, the adapted value for each pixel is the weighted sum of its R, G, and B values. The weights (w_r, w_g, w_b) are initialized to 1.0 for the first pass. For each subsequent image to be processed, the weights are adjusted to reflect the segmentation obtained in the previous image. The cropline computed in the previous image delineates two classes: left-of-the-cropline and right-of-the-cropline. The Fisher Linear Discriminant for these two classes is computed, yielding weights that project pixel values onto a line that maximizes the ratio of interclass to intraclass scatter. Thus, the weights adjust the pixel values to achieve the greatest separation between the two classes. The criterion function, $J(w)$, to be maximized for the Fisher Linear Discriminant is expressed as follows [5]:

(EQ 2)

$$J(w) = \frac{w^t S_B w}{w^t S_W w}$$

where w is the weight vector projecting samples onto the line, S_B is the between-class scatter matrix, and S_W is the within-class scatter matrix. For a two-class problem, S_W is the sum of the scatter matrices of the two classes, with scatter matrix for class i defined as:

(EQ 3)

$$S_i = \sum_{x \in X_i} (x - \mu_i) \cdot (x - \mu_i)^t$$

and S_B defined as:

(EQ 4)

$$S_B = (\mu_1 - \mu_2) \cdot (\mu_1 - \mu_2)^t$$

Figures 5-8 demonstrate the affect of applying the adaptive weights. In figures 5-7, the raw intensity values for R, G, B across a scanline are shown. Figure 8 shows the weighted sum of the R, G, and B values for this same scanline. A clear step emerges from what are otherwise noisy signals.

FIGURE 5. Graph of Red Values across the Scanline

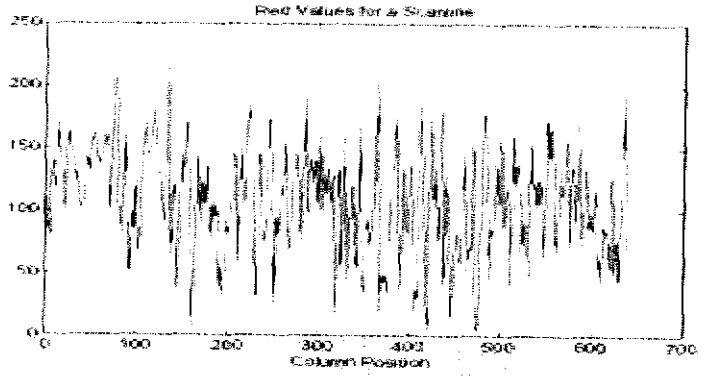


FIGURE 6. Graph of Green Values across the Scanline

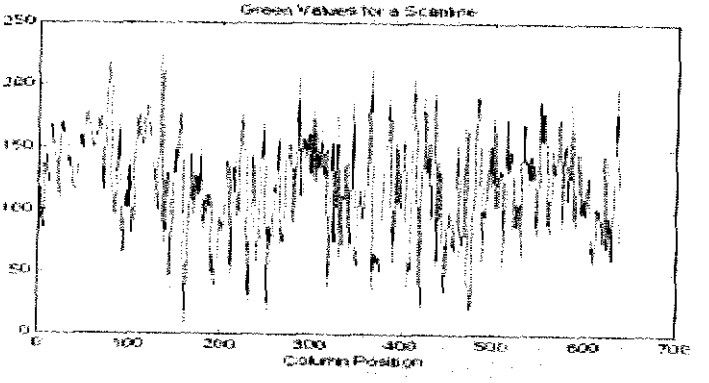


FIGURE 7. Graph of Blue Values across the Scanline

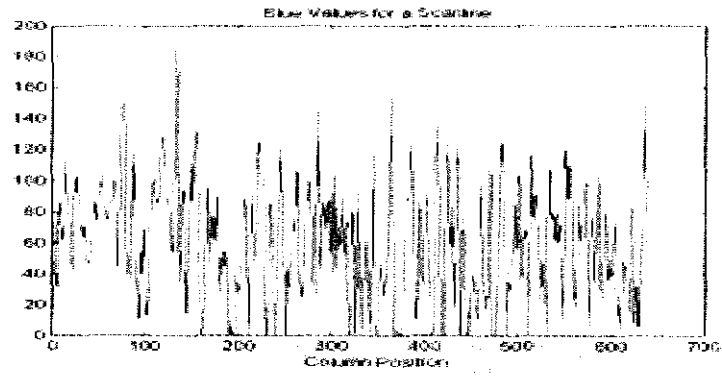
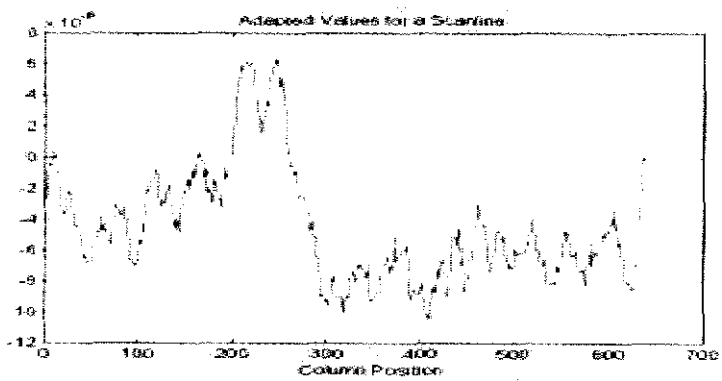


FIGURE 8. Graph of Adapted Values across the Scanline



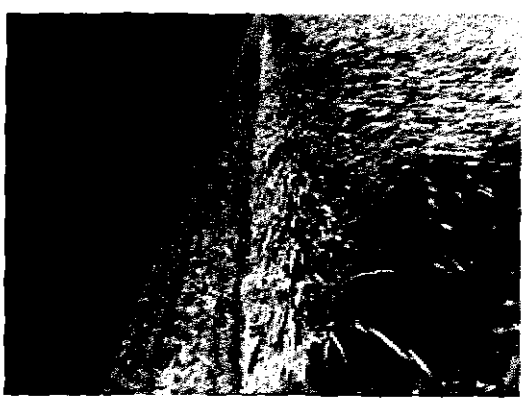
5.0 Modifications to Ollis' Adaptive Algorithm

The success of the adaptive algorithm depends on whether a key assumption holds for the images in question: the location of the cropline must correspond to a bimodal step. That is to say, the image must be segmentable into two regions. And these two regions must be separable by means of the linear discriminant used. This adds an additional assumption, namely that cut and uncut crop classes are linearly separable.

The first assumption is often violated in images of crop. Rows of cut crop produced by the harvester, called windrows, appear in the images. Windrows produce two additional steps, one on each side of the windrow. Depending on the appearance of the cut sections between windrows and uncut crop and between windrows, the best step in the image might be a windrow edge. This has been found to occur frequently in images of sudan.

To avoid segmenting images along the windrow edges, two methods for forcing the bimodal assumption were developed. The first attempts to restrict the region of the image that is processed by the adaptive algorithm to only those portions which include crop and the cut region adjacent to the crop, as shown in Figure 9. This method simply cuts out a section of the image and processes the rest. The second method considers the entire image, but restricts how much of the image is included in the step computation for each column in the image. As the algorithm moves across the columns looking for a step, the amount of the image on the side of the dividing line considered to correspond to cut crop that will be used for the step computation is limited to the number of pixels that would be expected to be found in the section between the uncut crop and the closest windrow.

FIGURE 9. Example of Restricting Area of Processing: Dark region on the left is not processed



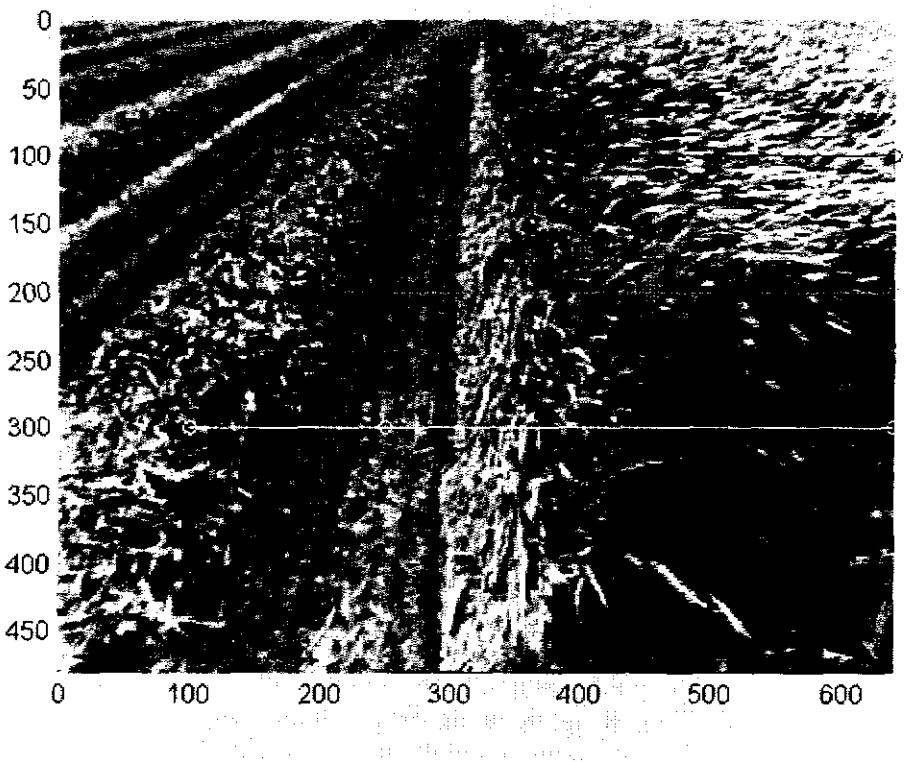
Each of these methods requires two user inputs. These are the desired width of the cut and the expected width of the windrow. Because the width of the windrow is controlled by settings on the harvester, the user can easily provide this information. Similarly, the desired cut width is known, because the fields are laid out in sections of uniform size separated by raised dirt borders.

Using these two user inputs and the known camera parameters, the perspective projection of the valid regions can be computed. For the first method, the position of the ideal cropline in the image given that the harvester is cutting at the desired width is easily computed. Using the expected windrow width and the expected cut width, we can compute an estimated width of the cut region between the windrow and the uncut crop. The rest of the image beyond this cut region is not used.

The second method also uses the expected size of the cut region adjacent to the uncut crop to restrict how much of the image is considered. The difference is that the regions that are processed changes from column to column; no single region is excluded. For each scanline, the algorithm passes over each column and computes the step using all the pixels on what is the expected uncut crop side of the current column, but using only those pixels on the expected cut side which would fall in the immediately adjacent cut region as it is projected on the image with the current column as one of its edges. The number of pixels on the cut side used for computing the fit of the step is therefore a

function of column position, the expected size of the adjacent cut region, and the perspective projection. Figure 10 shows how processing would be restricted for 3 rows in an image. The leftmost circle of each of the lines indicates the left boundary of the region processed for that scanline and the column indicated by the middle circle. The algorithm is searching for a cropline point at the position of the middle circle by considering everything between it and the leftmost circle as cut crop and everything between it and the rightmost circle as uncut. As the middle circle is moved across the columns, the position of the leftmost circle changes, as can the distance between it and the middle circle.

FIGURE 10. Example of Scanline processing restrictions for 3 Rows of an Image



The second method has two important advantages over the first. To exclude the correct region, the first method needs a third user input, namely the initial expected curvature of the cropline. In subsequent steps of the algorithm, the correct curvature can be computed for the croplines found, but this assumes that the curvature does not change substantially from image to image. Furthermore, the second method works even if the true cropline is not close to the ideal cropline in the image. The drawback of the second method is that it can still find best-fit steps that do not correspond to the true cropline, although with less frequency than the original adaptive algorithm.

6.0 Adaptive Color Edge Detection

The work of Gerrish et al. [8] using edge-detection techniques to find crop edges focused on intensity edges in grayscale images. This resulted in the detection of numerous spurious edges, which severely complicated the process of picking out the cropline. Cases in which there is no significant intensity difference between cut and uncut crop regions can also be expected, which would cause their algorithm to fail. Compared to searching for intensity edges, detecting color edges is more likely to pick out the cropline.

Rather than convolve masks with the image to find edges, the color edge detection algorithm developed here passes two adjacent windows across the image rows and compares the color properties of the windows with one another. The comparison produces a single value that expresses the strength of the edge for the column and row under consideration. This method adapts the comparative measure to characteristics of the windows under consideration, as compared with using a static edge detection mask. The value for comparison is computed as the sum of the inter-class squared Mahalanobis distances between the RGB pixel values of the two windows. Squared Mahalanobis distance is given as:

(EQ 5)

$$r^2 = (x-\mu)^t \cdot \Sigma^{-1} \cdot (x-\mu)$$

where μ is the mean of the class and Σ^{-1} the inverse of its covariance matrix. The sum of the inter-class squared Mahalanobis distance is represented as:

(EQ 6)

$$\left(\sum_{i=0}^{N}(x_{1i}-\mu_2)^t \cdot \Sigma_2^{-1} \cdot (x_{1i}-\mu_2)\right)+\left(\sum_{i=0}^{N}(x_{2i}-\mu_1)^t \cdot \Sigma_1^{-1} \cdot (x_{2i}-\mu_1)\right)$$

where x_{1i} and x_{2i} are samples from windows 1 and 2, μ_1 and μ_2 the means of windows 1 and 2, and Σ_1^{-1} and Σ_2^{-1} the inverses of the covariance matrices of windows 1 and 2 respectively.

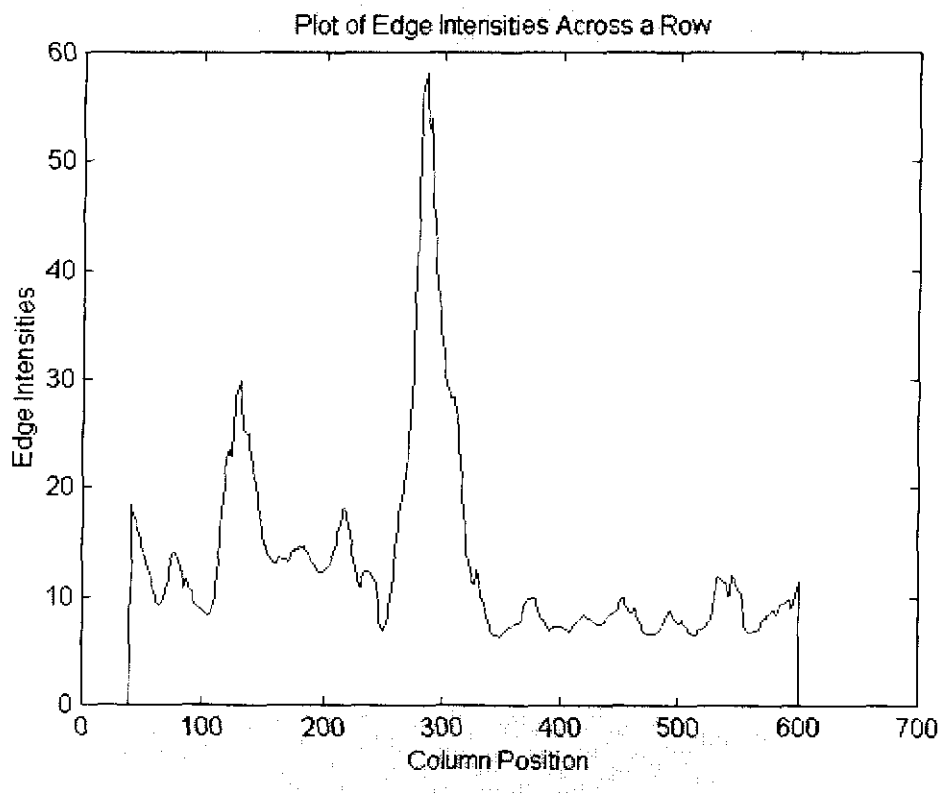
The algorithm proceeds as follows:

- *for each row i in image*
- *for each column j in row i*
- *compute the mean and covariance for the left window with upper left corner at row (i-windowSize/2) and column (j-windowSize+1) and lower right corner at row (i+windowSize/2) and column (j).*

- *compute the mean and covariance for the right window with upper left corner at row (i-windowSize/2) and column (j+1) and lower right corner at row (i+windowSize/2) and column (j+windowSize).*
- *compute sum of squared Mahalanobis distances of each pixel in the left window from the class defined by mean and covariance of right window*
- *compute sum of squared Mahalanobis distances of each pixel in the right window from the class defined by mean and covariance of left window.*
- *sum the prior two sums*

The algorithm produces an edge intensity image with gradually sloping peaks, as shown for a single scanline in Figure 11, rather than the discrete steps produced by many edge-detection routines. This routine is most effective on vertical edges, gradually degrading in performance as the direction of edges becomes more and more horizontal. Horizontal edges can be detected by changing the direction of the windows. In this case, the two windows would consist of a top and bottom window, rather than the left and right windows of the algorithm above.

FIGURE 11. Color Edge Intensities for a scanline



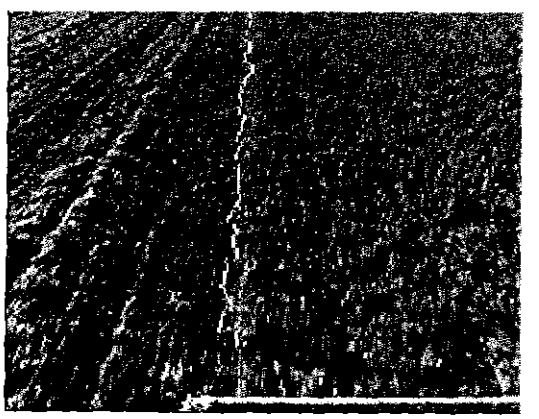
Often, a cropline can be extracted from the distances image by searching for the maximum distance score for each scanline. An example of a cropline found in this manner is

shown in Figure 12. However, this method is not reliable for all images of crop. Just as the cropline might not correspond to the maximum step found by the Ollis algorithm, it might also not correspond to the maximum color edge intensity found in each scanline.

The following procedure was adopted to account for the possibility that the maximum peak does not correspond to the cropline for each scanline:

- *Once the distances image is produced, smooth each scanline, removing the extraneous peaks.*
- *Pick out the top five peaks in each scanline. These are candidate cropline points for the scanline.*
- *Starting at the bottom scanline of the image and moving up, connect candidate cropline points from one scanline to the next using a distance threshold. Croplines should be fairly smoothly connected, so large discontinuities from one scanline to the next indicates the end of a cropline (a threshold for how many scanlines a large discontinuity should persist before the end of a cropline is declared can be used). Candidate croplines that are too short (again a threshold) should be thrown out.*
- *Remaining candidate croplines are compared by computing the average edge intensity on the right and left sides of the base of the peak in question. Knowing which camera is being used provides the expected side where the crop should fall. This side should correspond to the minimum average distance because of the relative homogeneity of the crop. The winning cropline is the one that minimizes average distance on the expected crop side and maximizes it on the expected non-crop side. A single discriminant can be used to pick the winner: the cropline that minimizes the ratio of average distance on crop side to average distance on non-crop side.*

FIGURE 12. Cropline found by Color Edge Routine Searching for Maximum Edge Values



Although the procedure described above requires several thresholds, these thresholds are easily determined and correspond to expected features in an image. Croplines should be continuous and five major peaks can be expected at the top of the image corresponding to the edges of two windrows and the cropline itself. Increasing this threshold to account for spurious peaks in the crop will not significantly add to the computational

burden, as such peaks will be thrown out when candidate croplines are generated because they won't produce a long enough cropline.

7.0 Model-based Color Segmentation

Ollis [11] describes a Bayesian approach to finding croplines. He constructs pdfs of cut and uncut crop by filling out histograms for each class, identifying all pixels on one side of the cropline as one particular class and all pixels on the other side as the other class (which side is cut and which uncut is determined by which camera is taking the image, and therefore which side the uncut crop is expected to be on). The histograms updated on the previous cycle are used to determine the cropline for the current cycle. This is done by computing the maximum likelihood estimate, as given by:

(EQ 7)

$$MAX\left(\prod_{j=0}^{j_{current}} p(cut|f(i,j)) \cdot \prod_{j=j_{current}+1}^{j_{max}} p(uncut|f(i,j))\right)$$

where $j_{current}$ is the current column position and $f(i,j)$ is the pixel for row i, column j. This maximum likelihood estimate relies on the assumption that the cut crop is on the left side of any prospective cropline and the uncut crop is on the right. Reversing this assumption produces an estimate for a cropline to be detected on the opposite side of the harvester.

This method is susceptible to throwing out prospective cropline points because of the influence of a few pixels with high probabilities of belonging to the class opposite the side which they fall on relative to the cropline points under consideration. For example, if most of the pixels on the left side of a prospective cropline point only weakly belong to the left class, but those few pixels on the left that belong to the right class do so strongly, this prospective cropline point is likely to be rejected as the maximum likelihood estimate of the cropline.

An alternative method is to classify each of the pixels individually, and use the image of classified points to segment the original image. A third class, shadows, is added to the classes for cut and uncut crop. The pdfs for these three classes are structured as multivariate normal densities. The general form for the multivariate normal density is given as [5]:

(EQ 8)

$$p(x|w_i) = \left(\frac{1}{(2\pi)^{\frac{d}{2}} \cdot |\Sigma_i|^{\frac{1}{2}}}\right) \cdot \exp\left[-\frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i)\right]$$

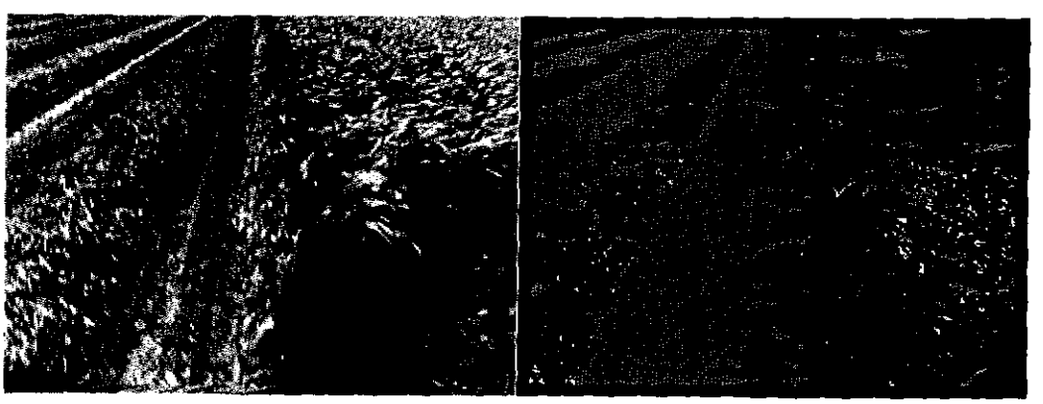
where x is a d-component feature vector, μ_i is the mean vector for class i, and Σ_i is its covariance matrix. The probability of a particular class i given the pixel in question is, according to Bayes rule, written as:

(EQ 9)

$$p(w_i|x) = p(x|w_i)p(w_i)$$

The method classifies pixels in the initial image using class models obtained from previously collected data. Figure 13 shows an example image and the classification of the pixels. Pixels colored in gray are classified as cut crop, those in black as uncut crop, and those in white as shadows. The windrow is classified as uncut crop, which is not surprising given the sharper difference between soil and uncut crop than that between windrow and crop.

FIGURE 13. Image of Sudan and Classification of Pixels



Once a classified image is produced, the cropline is computed. The classified image of Figure 13 indicates that using a connected components followed by a method to extract the edge of the main crop component and a method to smooth this edge would yield a reasonable cropline. Unfortunately, connected components routines are notoriously slow. Instead, the step-function algorithm used by the second modified adaptive algorithm is used here, without, of course, the initial step of multiplying the image values by adaptive weights. Applying this algorithm to the classified image is similar to doing template matching, with the template consisting of a step changing in size as it is moved down the image.

8.0 Texture-based Segmentation

The example of sudan in Figure 3 reveals the similarity in color between uncut crop and cut regions that is often found in such fields. This similarity presents difficulties for any segmentation routine that keys off color differences, especially one that relies on the assumption that the regions in RGB-space corresponding to cut and uncut crop are linearly separable. The difference in texture between cut and uncut regions of sudan, how-

ever, is striking. Using a texture-based segmentation routine, either in conjunction with or as a replacement for color-based methods, is a self-evidently appropriate approach for finding croplines in sudan images. There are a number of reasons, detailed below, why this approach might not be appropriate for finding croplines in alfalfa images.

It is important to keep in mind the constraints placed on computational expense by this problem domain. Many popular texture segmentation routines cannot cycle at a minimum of 4 Hz. on currently available hardware unless run on a supercomputer. For example, the method reported by Daily [3] using Markov Random Fields (MRFs) to segment textures required 60 hours on 8 Lisp machines to test the parameters for the MRFs to determine the best set for segmenting the test images. Although MRFs are apt for detecting changes in microtexture, an attribute likely to be useful for segmenting alfalfa images, the computational complexity is too great for a real-time system.

8.1 Texture Segmentation Using Gabor Filters

Because the image plane is not parallel to the scene plane (the camera is not looking straight down, but is tilted away from the scene), perspective effects can be expected. The result is that the area, aspect-ratio, and density of texture elements change as depth varies. If a method based on texture filters such as LOG operators is used, it will be necessary to apply filters over a range of scales to capture the change in texture element attributes. This raises an additional consideration: texture attributes indicative of uncut crop at one depth might be indicative of cut crop at another depth. A simple method for avoiding this problem will be discussed later.

Sudan places an additional constraint on the type of texture operator that can be used. Because the crop grows in excess of 6 ft. tall and is not rigid, the stalks move considerably in even moderate winds. The crop, therefore, can be found in any orientation, and, in fact, a mix of orientations can be found in the same image. Thus, it does not make sense to use oriented texture operators.

The texture operator used for segmenting sudan images is the gabor filter [6.13], which is a complex sinusoid modulated by a Gaussian envelope. A 2-D gabor filter is represented by the function [15]:

(EQ 10)

$$\exp\left(-\frac{[(x-x_0)^2+(y-y_0)^2]}{2\sigma^2}\right)\sin(\omega[x\cos(\theta)-y\sin(\theta)]+\varphi)$$

where x_0 and y_0 locate the center of the Gaussian, ω is the frequency of the sinusoid, θ is its orientation, and φ its phase. Gabor [7] demonstrated that the 1-D filter has optimal joint resolution, that is, it demonstrates the optimal trade-off between spatial and frequency domain resolutions. Daugman [4] extended this proof to the 2-D filter.

To meet the constraints imposed by sudan images and the need for real-time operations, 1-D gabor filters at several scales are used to filter the input images. Because orientation is not of extreme importance, the 1-D filters are convolved vertically and horizontally

with the image at each scale. Three scales are used for each orientation, yielding size total filters. These filters were hand-selected, although methods do exist for determining optimal filter characteristics [15-16].

Once a filtered image is produced, the magnitude of the response for each filter is determined, and the Laws Energy [10] for the output of the magnitude operator on each filtered image is computed. Laws Energy for a 7x7 neighborhood of (r,c) is defined as:

(EQ 11)

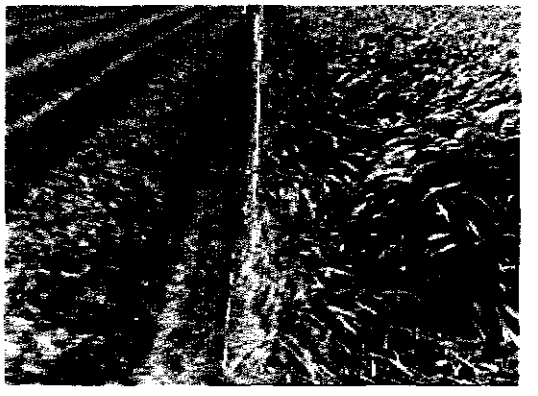
$$E_n(r,c) = \frac{1}{49}\sum_{i=-3}^{3}\sum_{j=-3}^{3}|J_n(r+i,c+j)|$$

where J_n is the value of the image filtered by kernel n.

The output of the energy computation for each filter is then normalized to have unit mean. The normalized, filtered images are then used to compute the cropline. A variety of techniques can used at this juncture, including the step-function method of Ollis' algorithm or a classification scheme such as presented in section 7.0. To account for projective distortion, the images are divided into three sections. The two vertical and two horizontal filters with the most appropriate scale for the depth in the section in question are considered. Experimentation as to which scale of filter provides the strongest response to sudan at the range in question is easily done. This method produces a 4-dimensional feature vector for each position in the image, rather than the 6-dimensional vector resulting from considering the output of every filter. These feature vectors can then be used as the input to the adaptive algorithm described by Ollis. This method can be modified so that each third of the image has its own set of adaptive weights.

A simpler method for picking out the cropline has proven to be sufficient to pick out croplines from filtered images. Sudan has such pronounced texture as to allow for simply picking out the largest step in texture energy without use of adaptive weights. An example cropline computed in this manner is given in Figure 14. The benefit of this simpler method is increased speed.

FIGURE 14. Cropline Found Using Texture Filters



9.0 Results

Five data sets were used to compare the accuracy of the algorithms, with each data set representing a unique crop, field, or environmental factor. Each data set consists of 20 images, producing a total of 100 images. The five data sets are: 1) alfalfa on flats; 2) alfalfa on beds; 3) sudan on beds; 4) sudan on flats; 5) sudan on flats in high winds. The last data set was selected to illustrate factors other than texture and color which may affect segmentation.

Accuracy is measured as the deviation of the computed croplines from a hand-selected ideal cropline. Four measures are given for each data set: average error in pixels from the computed croplines to the ideal croplines, the percent of computed cropline points that fall within 10 pixels of the ideal, average error in meters from the projection of the computed croplines onto the scene and the projection of the ideal cropline, and the standard deviation of this measure. The latter two measures provide a sense of how useful the computed croplines are for accurately steering the vehicle.

The first data set is a subset of that used by Ollis [11]. The modified adaptive algorithm produces the best results, although the model-based method also produces a substantial

number of cropline points. The texture method is not considered here nor for the next data set, because its use is meant primarily for images with high texture information.

TABLE 1. Performance of Color Segmentation Algorithms on Test Set from [11] (Alfalfa on flats)

Algorithm	Avg. Error (in pixels)	% within 10 pixels	Avg. Error (in meters)	Standard Deviation (in meters)
Adaptive	17.52	70.6	0.36	1.57
Modified Adaptive	14.27	78.3	0.31	1.42
Model-based	25.31	65.3	0.62	1.85
Color edge	51.8	54.8	0.92	1.69

The second data set is a series of images taken in El Centro while cutting in alfalfa fields on beds. The adaptive and color edge routines provide similar average error, but the adaptive provides a greater number of useful cropline points.

TABLE 2. Performance of Color Segmentation Algorithms on Images of Alfalfa on Beds

Algorithm	Avg. Error (in pixels)	% within 10 pixels	Avg. Error (in meters)	Standard Deviation (in meters
Adaptive	9.6	64.0	0.19	0.19
Modified Adaptive	8.9	67.2	0.17	0.18
Model-based	20.0	49.0	0.64	2.26
Color Edge	12.9	42.7	0.38	0.56

The performance of the modified adaptive algorithm that considers only the region of the image where the crop and first cut region are expected to appear is provided as a point of comparison for images of sudan on beds. The best performance according to both measures is the model-based method.

Contrary to expectations, the texture method performs poorly. This is due largely to the edge of the crop being flattened by the harvester's cutter on the previous pass, suppress-

ing texture information for this section of crop (which takes up about 20 pixels at its widest), and to a substantial shadow in the bottom of the image.

TABLE 3. Performance of Algorithms on Images of Sudan on Beds

Algorithm	Avg. Error (in pixels)	% within 10 pixels	Avg. Error (in meters)	Standard Deviation (in meters)
Adaptive	33.9	42.6	1.41	4.08
Modified Adaptive	12.6	76.8	0.74	3.76
Model-based	10.9	86.5	0.67	3.25
Color Edge	36.7	69.3	0.53	1.64
Texture	18.3	80.4	0.30	0.39

The texture method is the clear choice for the fourth data set. The others do not provide enough useful cropline points to be able to correctly discard outliers.

TABLE 4. Performance of Algorithms on Images of Sudan on Flats (Set 1)

Algorithm	Avg. Error (in pixels)	% within 10 pixels	Avg. Error (in meters)	Standard Deviation (in meters)
Adaptive	81.2	0.8	2.85	2.96
Modified Adaptive	33.1	2.9	0.73	1.83
Model-based	37.4	1.8	0.97	2.06
Color Edge	48.3	1.8	0.78	0.88
Texture	28.8	40.6	0.43	0.40

The final data set demonstrates the difficulty of finding an appropriate algorithm for navigating through an entire field. It is drawn from images of the same field as the fourth data set, but from a different pass. During this pass, strong winds blew the 5 ft.+ crop wildly, obscuring the cropline.

TABLE 5. Performance of Algorithms on Images of Sudan on Flats (Set 2: High Winds)

Algorithm	Avg. Error (in pixels)	% within 10 pixels	Avg. Error (in meters)	Standard Deviation (in meters)
Adaptive	157.6	2.4	3.63	4.56
Modified Adaptive	85.25	4.7	2.90	5.41
Model-based	83.34	4.3	2.08	3.73

TABLE 5. Performance of Algorithms on Images of Sudan on Flats (Set 2: High Winds)

Algorithm	Avg. Error (in pixels)	% within 10 pixels	Avg. Error (in meters)	Standard Deviation (in meters)
Color Edge	129.8	6.0	2.73	3.70
Texture	72.2	10.6	0.99	0.88

The speed of the algorithms is compared on a Pentium III 500 Mhz single-processor computer. Each algorithm is easily parallelized, enabling the slower algorithms to reach the 4 Hz. minimum without the use of expensive hardware.

The cycle rate of the color edge method is computed running the algorithm with 20x20 windows over every pixel in the image. This produces highly redundant information, because adjacent sets of windows have 95% overlap. A sparser edge intensity image can be produced by applying windows with no row overlap. This speeds up the algorithm by approximately a factor of 20. Furthermore, once the initial cropline is found, it is not necessary to process the whole image in subsequent iterations. Because finding a color edge does not rely on information outside of the two 20x20 windows, the algorithm can focus in on a relatively small region of the image once an initial cropline has been found. A 100 pixel window around the previous cropline is more than adequate to find the new cropline. This yields an additional speed-up of over 6 times (the images in this example have 640 columns). With these two speed-ups, the color edge routine runs at the minimum 4 Hz. without the need for parallelization.

TABLE 6. Algorithm Cycle Rate in Hz.

Algorithm	Cycle Rate
Adaptive	30 Hz.
Modified Adaptive	30 Hz.
Model-based	4 Hz.
Color Edge	0.033 Hz.
Texture	2 Hz.

10.0 Conclusions

The results from the data sets, added to those of Ollis [11], demonstrate that a single technique relying exclusively on RGB or texture information is insufficient to accurately segment images arbitrary field types. What is needed are techniques specifically tuned for specific field types. The results from the above experiments indicate possible combinations of techniques for specific crop and field types.

The modified adaptive algorithm provides the best results for alfalfa, whether on flats or beds. There are drawbacks, however, associated with each type of modified adaptive:

the algorithm either requires the probable cropline curvature to be known a priori or is still susceptible (though to a lesser degree than the original version) to spurious steps in the image. It would be helpful to have both an estimate of the curvature and a sense of what portion of the image can be discarded to overcome these limitations. The color edge detector can be of use in providing this information. Linking edges as described in section 6.0 offers a set of possible croplines. Some of these croplines will be spurious ones, a result of the color difference between windrow and cut crop region. Even these spurious boundaries are useful in estimating the curvature of the cropline, because the curvature of the windrow from the previous cut matches that of the current (true) cropline. This is due to the fact that this cropline was made in the act of creating the windrow. The color edge detector also can give a rough estimate of how much of the image can be discarded. Although the color edge detector returns an estimate of the true cropline, the results indicate that this cropline can have considerable error. This error is often due to bare patches of crop along the cropline. The boundary between windrow and cut crop, however, tends to be more accurate, because harvesters form fairly uniform windrows. This is the boundary that is of greatest importance for deciding how much of the image can be discarded, since the goal is to prevent spurious boundaries along the windrow.

Sudan fields present greater problems, as will any field of tall crop. The color-based techniques failed in sudan on beds because they picked the boundary between bent over uncut crop and cut crop as the true cropline. Error in the texture method was largely due to the presence of strong texture in the cut crop sections. More sensitive texture methods can conceivably differentiate between the texture in the cut crop sections and the crop itself, but this is to ignore a larger problem. The images of sudan on beds were fairly ideal because of the lack of wind. This cannot be counted on. Tall crop in high winds can completely obscure the cropline. Unfortunately, how the cropline is obscured is highly unsystematic. Winds range from highly directional (blowing the crop over to one side) to swirling, causing the 7 ft. crop to swirl around. The devastating affect of this can be seen in the results on sudan in high winds. None of the techniques comes close to identifying the true cropline, and steering the harvester based on their outputs would be a random walk.

The problem of shadows was not addressed here, and it is a formidable one for cropline following. Even shorter crops such as alfalfa cast long enough shadows to require that the harvester not follow the shadow (if the shadow is only a few inches wide, it would be reasonable to follow it). As the harvester passes through crop, it pushes uncut crop over, resulting in a shadow in the next pass that falls across the crop to the cut regions. A segmenter would have to discriminate between shadowed crop and cut regions. This is exceptionally difficult when cutting on beds, because longer crop stalks are left behind in cut regions, which then appear similar in RGB space to uncut crop. Although the errors caused by shadows tend to be systematic, deviating from true cropline points by a consistent distance, the difficulty lies in identifying this distance. Once identified, it can be easily taken into account. One possible solution is to use vision guidance as a form of cruise control, and allow the machine operator to adjust the detected cropline to compensate for shadowing.

The methods presented in this work have the greatest chance of success in fields of thick, short crop, which casts short shadows and has few bare spots. Additional sensing technologies will be required to enable accurate cropline following in fields of tall crop.

11.0 References

[1] Billingsley, J. and Schoenfisch, M. Vision-Guidance of Agricultural Vehicles. Autonomous Robots. 2. pp. 65-76 (1995).

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

[3] Daily, M. Color Image Segmentation Using Markov Random Fields. Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1989, pp. 304-312.

[4] Daugman, J. G. Two-dimensional Spectral Analysis of Cortical Receptive Field Profiles. Vision Research (20), 1980, pp. 847-856.

[5] Duda, R. and Hart, P. Pattern Classification and Scene Analysis. 1973, J. Wiley & Sons, pp. 114-118.

[6] Fogel, I. and Sagi, D. Gabor Filters as Texture Discriminator. Biological Cybernetics (61), 1989, pp. 103-113.

[7] Gabor, D. Theory of Communication. J. Inst. Elec. Eng. (93), 1946, pp. 429-457.

[8] Gerrish, John et al. Transactions of the Society of Automotive Engineers. 1987, paper 861455.

[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] Laws, K. Rapid Texture Identification. Proceedings of the SPIE Conference on Image Processing for Missile Guidance, 1980, pp. 376-380.

[11] Ollis, M. Perception Algorithms for a Harvesting Robot. Doctoral Thesis. CMU-RI-TR-97-43 August 1997.

[12] Ollis, M. and Stentz, A. First Results in Crop Line Tracking. Proceedings of IEEE Conference on Robotics and Automation (ICRA '96), Minneapolis, MN April 1996, pp. 951-956.

[13] Ollis, M. and Stentz, A. Vision-Based Perception for an Autonomous Harvester. Proceedings of the IEEE/RSJ International Conference on Intelligent Robotic Systems September, 1997.

[14] Pilarski, T., Happold, M., Pangels, H., Ollis, M., Fitzpatrick, K., and Stentz, A. The Demeter System for Automated Harvesting. Proceedings of the 8th International Topical Meeting on Robotics and Remote Systems. April, 1999.

[15] Turner, M. R. Texture Discrimination by Gabor Functions. Biological Cybernetics (55), 1986, pp. 71-82.

[16] Weldon, T. and Higgins, W. Design of Multiple Gabor Filters for Texture Segmentation. IEEE Int. Conf. Acoust., Speech, Sig. Proc., vol. IV, May 1996, pp. 7-10.

[17] Weldon, T., Higgins, W. and Dunn, D. Efficient Gabor Filter Design for Texture Segmentation. Pattern Recognition 29:12, 1996, pp. 2005-2015.

[18] Zhang, Q., Reid, J. and Noguchi, N. Agricultural Vehicle Navigation Using Multiple Guidance Sensors. Tech. Report UILU-ENG-99-7013.