Final Technical Report
Research Project T2695, Task 55
Contract Title: CCTV Phase 3

CCTV Technical Report

Phase 3

D.J. Dailey and F.W Cathey
Department of Electrical Engineering
University of Washington
Seattle, Washington 98195

Washington State Transportation Center (TRAC)
University of Washington, Box 354802
University District Building, Suite 535
1107 N. 45th Street
Seattle, Washington 98105-4631

Washington State Department of Transportation
Technical Monitor
Ted Trepanier, State Traffic Engineer

A report prepared for
Washington State Transportation Commission
Department of Transportation
and in cooperation with
U.S. Department of Transportation
Federally Highway Administration

January 2006

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO.	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
WA-RD 635.2		

4. TITLE AND SUBTITLE	5. REPORT DATE
THE AUTOMATED USE OF UN-CALIBRATED CCTV CAMERAS AS QUANTITATIVE SPEED SENSORS —PHASE 3	January 2006
	6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT NO.
Daniel J. Dailey and Frederick W. Cathey	
9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. WORK UNIT NO.
Washington State Transportation Center (TRAC) University of Washington, Box 354802 University District Building; 1107 NE 45th Street, Suite 535 Seattle, Washington 98105-4631	11. CONTRACT OR GRANT NO. Agreement T2695, Task 55
12. SPONSORING AGENCY NAME AND ADDRESS	13. TYPE OF REPORT AND PERIOD COVERED
Research Office Washington State Department of Transportation Transportation Building, MS 47370 Olympia, Washington 98504-7370 Doug Brodin, Project Manager, 360-705-7972	Final Technical Report
	14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

This study was conducted in cooperation with the U.S. Department of Transportation, Federal Highway Administration.

16. ABSTRACT

The Washington State Department of Transportation (WSDOT) has a network of several hundred closed-circuit television (CCTV) traffic surveillance cameras that are deployed for congestion monitoring on the freeways and arterials around Seattle. The goal of the first two phases of this project was to create *algorithms* that would allow these cameras to make continuous quantitative measurements of vehicle speed. In the first two phases, a number of algorithms were developed and tested; the most successful of these was chosen for implementation in this, Phase 3. The goal of this third phase was to implement the algorithms as *prototype software*.

17. KEY WORDS	18. DISTRIBUTION STATEMENT
Closed-circuit television (CCTV), traffic surveillance., congestion monitoring, speed sensors, camera calibration	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22616

19. SECURITY CLASSIF. (of this report)	20. SECURITY CLASSIF. (of this page)	21. NO. OF PAGES	22. PRICE
None	None		

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the views or policies of the Washington State Transportation Commission, Department of Transportation, or Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

Contents

List of Figures

Executive Summary

The Washington Department of Transportation (WSDOT) has a network of several hundred closed-circuit television (CCTV) traffic surveillance cameras deployed on the freeways and arterials around Seattle for congestion monitoring. No camera parameters are known *a priori* for these cameras; WSDOT operators dynamically change their focal length (f), pan angle (θ), and tilt angle (ϕ). The goal of this project was to create algorithms and prototype software to allow these cameras to be used to make continuous quantitative measurements of vehicle speed. However, to use the cameras for quantitative measures, the camera parameters have to be estimated. In order to measure speeds of vehicles from a series of video images it is not necessary to completely calibrate the camera; rather, by using the vanishing point of lines in the image algebraic constraints can be established on the parameters that are sufficient to straighten the image and compute a scale factor for estimating speed.

Chapter Two of this report contains the mathematics that are the intellectual basis for a prototype application to perform the quantitative functions described above. Chapter Three provides details on the calibration process. Chapter Four describes the prototype Java application created with the algorithms developed. The resulting application allows a user to select from a list of cameras, examine the camera view, calibrate the camera, and record speed data to a file. Screen shots of this application appear in Chapter Four.

Chapter 1

Introduction

This report presents a new computer vision approach to traffic speed estimation using uncalibrated highway surveillance cameras. Innovative image processing techniques include the following: (1) use of perspective straightening for image linearization, (2) an autocorrelation technique for lane stripe detection and estimation of the linear pixel/foot scale factor, and (3) a cross correlation technique used to estimate mean traffic speeds and direction of flow.

The Washington Department of Transportation (WSDOT) has a network of several hundred closed-circuit television (CCTV) traffic surveillance cameras deployed on the freeways and arterials around Seattle for congestion monitoring. Approximate location information is provided for each camera, such as highway name, cross-street, and position relative to the highway (north, south, east, west, median, or center), but no calibration parameters are provided. Camera height (h) above ground is unknown, and WSDOT operators can change the focal length (f), pan angle (θ), and tilt angle (ϕ) at will by remote control.

In surveillance situations with favorable geometry, traffic speeds can be measured from a series of video images without the calibration parameters h, f, θ or ϕ, actually being determined(assuming that they are constant for the series). Attention is restricted to cameras that surveil approaching and/or receding traffic: the line of sight must be directed slightly downward and more or less along the highway, rather than across it. Highway lanes and lane boundaries must also be approximately straight in the region of surveillance near the camera. Under these conditions, the highway lines will appear straight in a camera image and will have a recognizable vanishing point.

Given the highway vanishing point, a *straightening* transformation is defined that

maps lines parallel to the highway to vertical image lines. Moreover, the highway-to-image scale factor is constant for each such line. It can be determined by measuring the vertical dimension of an image feature with known length along the highway, such as lane stripe period. Because traffic pixel speed can be estimated by cross correlating lines in successive pairs of straightened images, speeds on the highway can be readily estimated by using the scale factor.

This report covers three main topics. First are presented the mathematics of straightening and its application to camera calibration. Next are presented image processing algorithms for highway vanishing point estimation and image straightening, as well as scale factor and speed estimation. In addition, a software application of these concepts and algorithms that interfaces with the WSDOT camera system is described.

Chapter 2

Theory

This section presents the mathematical theory behind perspective straightening. The theory is based on a standard pin-hole perspective camera model and depends on properties of vanishing points, which are described. The mathematics prove that the composite transformation from highway ground plane coordinates to straightened image coordinates is a *projective shear* preserving vertical lines, as well as, in fact, an *affine shear*, provided the camera has zero roll angle with respect to the ground plane. In the latter case, explicit formulas are given for the camera parameters in terms of the vanishing point coordinates, and the horizontal and vertical highway-to-image scale factors. In practice, only the vanishing point and the vertical scale factor are estimated, since the horizontal factor is not needed for speed estimation. [1]

2.1 Coordinate Systems and Camera Model

This section reviews coordinate system conventions and the pin-hole camera model presented by Cathey and Dailey (2004).

The **highway coordinate system** is a right-handed Cartesian system with origin in the ground plane below the camera. We let $X = (x, y, z)$ denote the coordinate vector for a variable point with respect to this system. The x-axis is parallel to the highway direction, tangent to the road surface and oriented so as to make an acute angle with the camera line of sight (the camera is looking "down" the road). The y-axis is also tangent to the road surface, while the z-axis points down. Thus, sighting down the positive x-axis, the positive y-axis points to the right.

[1] However, see the remark at the end of section 3.2.

The **camera coordinate system** is a right-handed Cartesian system with origin at the camera eyepoint. We let $\bar{X} = (\bar{x}, \bar{y}, \bar{z})$ denote the coordinate vector for a variable point with respect to this system. The positive $\bar{x}$-axis is directed along the camera line of sight. The $\bar{y}$- and $\bar{z}$-axes are oriented parallel to the central row and column of the image surface respectively.

The camera and highway coordinates for a point are related by the following (invertible) affine transformations:

$$X = A\,\bar{X} + B \tag{2.1}$$

$$\bar{X} = A^T\,(X - B), \tag{2.2}$$

where

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \qquad B = \begin{pmatrix} 0 \\ 0 \\ -h \end{pmatrix}. \tag{2.3}$$

A is a rotation matrix (i.e., an orthogonal matrix with determinant = 1). The columns of the matrix A give the highway coordinates for the unit vectors along the camera coordinate system axes. B is the coordinate vector for the camera eye-point in highway coordinates, and h is the perpendicular height of the camera above the road.

The **centered image coordinates** corresponding to a point $\bar{X}$ in the field of view of the camera are defined by a simple projective transformation:

$$\begin{aligned} u &= f\,\bar{y}/\bar{x} \qquad && \text{(positive direction to the right)} \\ v &= f\,\bar{z}/\bar{x} \qquad && \text{(positive direction downward),} \end{aligned} \tag{2.4}$$

where $f > 0$ is a scale factor that depends on the focal length of the camera. The origin of this coordinate system lies on the camera line of sight.

Actual image data are provided as a rectangular array of "pixels" or samples of image intensity. The location of a pixel in the image is specified by its **pixel coordinates**, $(c,\, r)$, the column and row indices of the pixel in the array. We follow the convention that indices start at 0, so the origin of the pixel coordinate system (0, 0) is at the upper left corner of the image. To impose centered image coordinates on an actual image, we need to know the pixel coordinates of the optical center, $(c_0,\, r_0)$ (i.e., where the camera line of sight axis pierces the image). Then uv-coordinates are

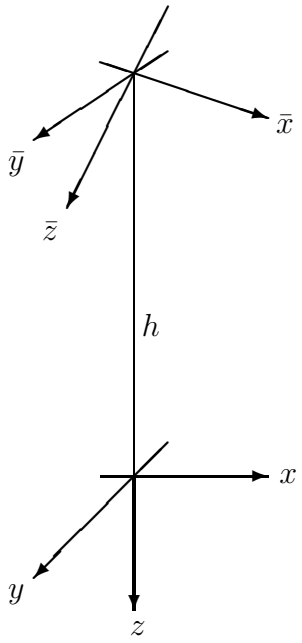

Figure 2.1: Highway and camera coordinate frames.

a simple translation of the cr-coordinates defined by $u = c - c_0$ and $v = r - r_0$. Note that the optical center may not be the center of the image and that the values c_0 and r_0 may not be integral.

The composition of transforms (2.2) and (2.4) constitutes the pin-hole perspective camera model for projecting 3-dimensional space (with highway coordinates) onto a 2-dimensional image plane (with centered image coordinates). In detail,

$$\begin{aligned} u &= f\,\frac{a_{12}x + a_{22}y + a_{32}(z+h)}{a_{11}x + a_{21}y + a_{31}(z+h)} \\ v &= f\,\frac{a_{13}x + a_{23}y + a_{33}(z+h)}{a_{11}x + a_{21}y + a_{31}(z+h)} \end{aligned} . \tag{2.5}$$

Because any rotation matrix is specified by three Euler angles, we see that the model has five independent parameters.

2.2 Ground Plane to Image Transformations

The following results are easily derived from equations (2.1) through (2.5) (see, e.g., Cathey and Dailey 2004). The forward and inverse transformations between the highway ground plane ($z = 0$) and the image plane are given by

$$\begin{aligned} u &= f\,\frac{a_{12}x + a_{22}y + a_{32}h}{a_{11}x + a_{21}y + a_{31}h} \\ v &= f\,\frac{a_{13}x + a_{23}y + a_{33}h}{a_{11}x + a_{21}y + a_{31}h} \end{aligned} \tag{2.6}$$

and

$$\begin{aligned} x &= h\,\frac{a_{11}f + a_{12}u + a_{13}v}{a_{31}f + a_{32}u + a_{33}v} \\ y &= h\,\frac{a_{21}f + a_{22}u + a_{23}v}{a_{31}f + a_{32}u + a_{33}v}. \end{aligned} \tag{2.7}$$

These are special cases of 2-dimensional projective transformations. For example, (2.6) can be expressed in the standard form with homogeneous coordinates

$$\begin{pmatrix} \lambda\, u \\ \lambda\, v \\ \lambda \end{pmatrix} = M \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{2.8}$$

where M is the invertible matrix defined by

$$M = \begin{pmatrix} f\,a_{12} & f\,a_{22} & f\,a_{32}\,h \\ f\,a_{13} & f\,a_{23} & f\,a_{33}\,h \\ a_{11} & a_{21} & a_{31}\,h \end{pmatrix}. \tag{2.9}$$

2.3 Highway Vanishing Point

The image coordinates of the vanishing point $(u_\infty,\, v_\infty)$ for lines parallel to the x-axis are found by holding y constant and letting $x \to \infty$ in (2.6):

$$u_\infty = f\,a_{12}/a_{11} \tag{2.10}$$

$$v_\infty = f\,a_{13}/a_{11}. \tag{2.11}$$

The following identities are consequences of the orthonormality of rows of A:

$$\begin{aligned} a_{11}f + a_{12}u_\infty + a_{13}v_\infty &= f/a_{11} \\ a_{21}f + a_{22}u_\infty + a_{23}v_\infty &= 0 \\ a_{31}f + a_{32}u_\infty + a_{33}v_\infty &= 0 \end{aligned} \tag{2.12}$$

and

$$f^2 + u_\infty^2 + v_\infty^2 = (f/a_{11})^2. \tag{2.13}$$

The latter identity is exploited in the analysis of complete camera calibration described in section 2.6.

Using (2.12), we can rewrite the image to highway transformation (2.7) in the form

$$\begin{aligned} x &= h\,\frac{a_{12}(u-u_\infty) + a_{13}(v-v_\infty) + f/a_{11}}{a_{32}(u-u_\infty) + a_{33}(v-v_\infty)} \\ y &= h\,\frac{a_{22}(u-u_\infty) + a_{23}(v-v_\infty)}{a_{32}(u-u_\infty) + a_{33}(v-v_\infty)}. \end{aligned} \tag{2.14}$$

Letting $m = (u-u_\infty)/(v-v_\infty)$ denote the slope of the line passing through $(u,\, v)$ and $(u_\infty,\, v_\infty)$, and letting $\xi = 1/(v-v_\infty)$ results in

$$x = h\,\frac{a_{12}\,m + (f/a_{11})\,\xi + a_{13}}{a_{32}\,m + a_{33}} \tag{2.15}$$

$$y = h\,\frac{a_{22}\,m + a_{23}}{a_{32}\,m + a_{33}}. \tag{2.16}$$

Conversely,

$$m = -\frac{a_{33}\, y - a_{23}\, h}{a_{32}\, y - a_{22}\, h} \tag{2.17}$$

$$\xi = \frac{a_{11}}{h\, f}\,(a_{32}\, m + a_{33})\, x - \frac{a_{11}}{f}\,(a_{12}\, m + a_{13}) \tag{2.18}$$

$$= -\frac{a_{11}}{f}\ \frac{A_{11}\, x + A_{21}\, y + A_{31}\, h}{a_{32}\, y - a_{22}\, h}, \tag{2.19}$$

where A_{i1} is the cofactor of a_{i1} in the matrix A,

$$\begin{aligned} A_{11} &= a_{22}\, a_{33} - a_{23}\, a_{32} \\ A_{21} &= a_{13}\, a_{32} - a_{12}\, a_{33} \\ A_{31} &= a_{12}\, a_{23} - a_{22}\, a_{13}. \end{aligned} \tag{2.20}$$

Formulas (2.18) and (2.19) follow directly from (2.15) and (2.17) and the fact that

$$a_{32}\, m + a_{33} = \frac{-h\, A_{11}}{a_{32}\, y - a_{22}\, h}. \tag{2.21}$$

Note that equations (2.16) and (2.17) show that each line through the vanishing point (m = constant) corresponds to a highway line parallel to the x-axis (y = constant). Equations (2.15) and (2.19) show that for each line through the vanishing point (m = constant), there is a linear dependence between x and ξ. Evidently the transformation $(u,\, v) \rightarrow (m,\, \xi)$ is a "straightening" transformation, but this is not the one exploited in this paper.

2.4 No Camera Misalignment

Consider the ideal situation [2] in which the road surface is level and there is no camera misalignment (the camera frame has no roll with respect to local level). Let the camera line-of-sight point downward from the local level with an angle $\phi < 0$ and let the vertical plane containing the line-of-sight make angle θ with the highway x-axis, where the sign of θ is determined by the right-hand rule. Then the rotation matrix A is given by

$$A = \begin{pmatrix} \cos\phi\cos\theta & -\sin\theta & \sin\phi\cos\theta \\ \cos\phi\sin\theta & \cos\theta & \sin\phi\sin\theta \\ -\sin\phi & 0 & \cos\phi \end{pmatrix} \tag{2.22}$$

[2] The results hold in the more general case when the camera y'-axis is parallel to the highway $x\,y$-plane (not necessarily local level), with appropriate modifications to the definitions of ϕ and θ.

Note that $a_{32} = 0$. This is a necessary and sufficient condition for the "no misalignment/zero roll" situation.

Formulas (2.10), (2.11), and (2.13) become

$$u_\infty = -f \sec\phi \tan\theta \tag{2.23}$$

$$v_\infty = f \tan\phi \tag{2.24}$$

$$f^2 + u_\infty^2 + v_\infty^2 = (f \sec\phi \sec\theta)^2. \tag{2.25}$$

By using (2.23) and (2.24), we can solve for ϕ and θ in terms of f, u_∞, and v_∞. Indeed, setting $w = \sqrt{f^2 + v_\infty^2}$, we have the following formulas for the sines and cosines of ϕ and θ:

$$\cos\phi = \frac{f}{w} \qquad \sin\phi = \frac{v_\infty}{w} \tag{2.26}$$

$$\cos\theta = \frac{w}{\sqrt{w^2 + u_\infty^2}} \qquad \sin\theta = \frac{-u_\infty}{\sqrt{w^2 + u_\infty^2}}. \tag{2.27}$$

Hence,

$$A = A(f, u_\infty, v_\infty) = \begin{pmatrix} \frac{f}{\sqrt{w^2+u_\infty^2}} & \frac{u_\infty}{\sqrt{w^2+u_\infty^2}} & \frac{v_\infty}{\sqrt{w^2+u_\infty^2}} \\ \frac{-u_\infty f}{w\sqrt{w^2+u_\infty^2}} & \frac{w}{\sqrt{w^2+u_\infty^2}} & \frac{-u_\infty v_\infty}{w\sqrt{w^2+u_\infty^2}} \\ -\frac{v_\infty}{w} & 0 & \frac{f}{w} \end{pmatrix} \tag{2.28}$$

2.5 Straightening

For certain image processing applications, such as lane stripe detection and traffic speed estimation, it is desirable to "straighten" a given region of interest by applying a projective transformation like that defined by (2.7). A straightened image corresponds to a sheared top-down view of the highway.

A straightening transformation, $(u, v) \to (x', y')$, is defined by using (2.7), but with suitable fictitious parameter values f', h', and a'_{ij} substituted for the true but unknown values. It is assumed that the vanishing point (u_∞, v_∞) for the highway lines is known[3] and a zero-roll rotation matrix is used, $A' = A(f', u_\infty, v_\infty)$, as defined in section 2.4. Here, f' is determined from (2.24) so as to correspond to the nominal depression angle ϕ' (say $-10°$). The value for h' can be arbitrary at this point. (In section 3.1.3 of image processing algorithms, h' is selected to produce a straightened image with a specific width.)

[3] An algorithm for computing the vanishing point is given in section 3.1.2.

The formulas for the forward and inverse transformations are analogous to (2.6) and (2.7):

$$\begin{aligned} u &= f' \, \frac{a'_{12}x' + a'_{22}y'}{a'_{11}x' + a'_{21}y' + a'_{31}h'} \\ v &= f' \, \frac{a'_{13}x' + a'_{23}y' + a'_{33}h'}{a'_{11}x' + a'_{21}y' + a'_{31}h'} \end{aligned} \tag{2.29}$$

$$\begin{aligned} x' &= h' \, \frac{a'_{11}f' + a'_{12}u + a'_{13}v}{a'_{31}f' + a'_{33}v} \\ y' &= h' \, \frac{a'_{21}f' + a'_{22}u + a'_{23}v}{a'_{31}f' + a'_{33}v}. \end{aligned} \tag{2.30}$$

Like the true highway-to-image transformations, lines parallel to the x'-axis correspond to image lines through the vanishing point (u_∞, v_∞). It follows that when the straightening transform (2.30) is composed with the true highway-to-image transformation (2.6), lines parallel to the x-axis correspond to lines parallel to the x'-axis. The discussion below proves that the points on each pair of corresponding lines are linearly related and that the composite highway to straightened image transform is in fact affine, provided that the camera is not misaligned. These properties are what make straightening a useful tool and allow for meaningful use of correlation techniques on a straightened image.

Theorem 1. *(a) The composition of the true highway-to-image transformation* (2.6) *and the straightening transformation* (2.30) *is a projective shear of the form*

$$x' = \frac{c_{11}\, x + c_{12}\, y + c_{13}}{c_{32}\, y + c_{33}} = r_1(y)\, x + r_0(y) \tag{2.31}$$

$$y' = \frac{c_{22}\, y + c_{23}}{c_{32}\, y + c_{33}} \qquad = r_2(y) \tag{2.32}$$

for certain constants c_{ij} and where each $r_i(y)$ is a fractional linear form in y. It follows that the "vertical scale factor" $r_1(y)$ is independent of x on any highway line.

(b) If the camera is not misaligned (i.e., has zero roll angle with respect to the ground plane), then $r_1(y)$ is constant and $r_2(y)$ and $r_0(y)$ are linear. In this case, the transformation from highway to straightened image coordinates is an affine shear of the form

$$\begin{aligned} x' &= \beta_{11}\, x + \beta_{12}\, y + \beta_{13} \\ y' &= \qquad\quad \beta_{22}\, y + \beta_{23} \end{aligned} \tag{2.33}$$

for certain constants β_{ij}. The scale factors β_{11} and β_{22} are given by

$$\beta_{11} = \frac{h'\, f'\, a_{11}\, a_{33}}{h\, f\, a'_{11}\, a'_{33}}, \qquad\qquad \beta_{22} = \frac{h'\, a'_{22}\, a_{33}}{h\, a_{22}\, a'_{33}}. \tag{2.34}$$

Proof. (a) Because the composition of two projective transformations is projective, the transformation from the xy-plane to the $x'y'$-plane must have the form

$$(x',\, y') = \left(\frac{c_{11}\, x + c_{12}\, y + c_{13}}{c_{31}\, x + c_{32}\, y + c_{33}},\ \frac{c_{21}\, x + c_{22}\, y + c_{23}}{c_{31}\, x + c_{32}\, y + c_{33}}\right)$$

for certain constants c_{ij}. In this case, the transformation maps lines that are parallel to the x-axis to lines that are parallel to the x'-axis. Because $x' \to \infty$ as $x \to \infty$, it follows that $c_{31} = 0$, and because y constant implies y' constant, it also follows that $c_{21} = 0$. Hence,

$$r_0(y) = \frac{c_{12}y + c_{13}}{c_{32}y + c_{33}}, \qquad r_1(y) = \frac{c_{11}}{c_{32}y + c_{33}}, \qquad r_2(y) = \frac{c_{22}y + c_{23}}{c_{32}y + c_{33}}.$$

There is a more constructive method to derive (a) which leads directly to a proof of (b). For the straightening transformation there are equations analogous to (2.15) and (2.16),

$$x' = \frac{h'}{a'_{33}}\,(a'_{12}\, m + \frac{f'}{a'_{11}}\,\xi + a'_{13}) \tag{2.35}$$

$$y' = \frac{h'}{a'_{33}}\,(a'_{22}\, m + a'_{23}). \tag{2.36}$$

Then (a) follows by substituting for m and ξ using (2.17) and (2.18). Explicit expressions for $r_0(y)$, $r_1(y)$, and $r_2(y)$ are given by

$$r_0(y) = \frac{h'}{a'_{33}}(a'_{12}\, m + a'_{13} - \frac{f'\, a_{11}}{f\, a'_{11}}(a_{12}\, m + a_{13})) \tag{2.37}$$

$$r_1(y) = \frac{h'\, f'\, a_{11}}{h\, f\, a'_{11}\, a'_{33}}(a_{32}\, m + a_{33}) \tag{2.38}$$

$$r_2(y) = \frac{h'}{a'_{33}}\,(a'_{22}\, m + a'_{23}) \tag{2.39}$$

where

$$m = -\frac{a_{33}\, y - a_{23}\, h}{a_{32}\, y - a_{22}\, h}. \tag{2.40}$$

(b) If the camera coordinate frame has zero roll angle with respect to the highway coordinate frame, then $a_{32} = 0$. Then m is a linear function of y, resulting in

$$r_1(y) = \frac{h'\, f'\, a_{11}\, a_{33}}{h\, f\, a'_{11}\, a'_{33}} \qquad \text{constant} \tag{2.41}$$

$$r_2(y) = \frac{h'\, a'_{22}\, a_{33}}{h\, a_{22}\, a'_{33}}\; y + \frac{h'\, a'_{22}}{a'_{33}}\,(a_{22}\, a'_{23} - a_{23}\, h) \qquad \text{linear in } y. \tag{2.42}$$

□

2.6 Complete Camera Calibration

A consequence of Theorem (1)(b) is the following calibration result.

Theorem 2. *Assume the vanishing point (u_∞, v_∞) of the highway lines is known and that a straightening transformation has been defined as above. Assume that the camera orientation has zero roll angle with respect to the highway plane so that Theorem* (1) *(b) holds true. Finally, assume that the scale factors β_{11} and β_{22} are known. (For example, β_{11} may be determined as the ratio of image lane stripe period to actual lane stripe period, and β_{22} as the ratio of image lane width to actual lane width.) Then it is possible to completely calibrate the camera. That is, we can determine the focal length, f, height, h, of the camera above the road, and the camera orientation angles θ and ϕ:*

$$f = \sqrt{\frac{1}{2}\,(d^2 - 2(u_\infty^2 + v_\infty^2) + d\,\sqrt{d^2 - 4u_\infty^2}\,)} \tag{2.43}$$

$$\phi = \arctan\frac{v_\infty}{f} \tag{2.44}$$

$$\theta = \arcsin\frac{-u_\infty}{f^2 + u_\infty^2 + v_\infty^2} \tag{2.45}$$

$$h = \frac{h'\, a'_{22}}{\beta_{22}\, a'_{33}}\,\frac{\cos\phi}{\cos\theta} \tag{2.46}$$

where

$$d = \frac{\beta_{22}}{\beta_{11}}\,\frac{f'}{a'_{11}\, a'_{22}}. \tag{2.47}$$

Proof. From (2.34) it follows that

$$\frac{f}{a_{11}\,a_{22}} = \frac{\beta_{22}}{\beta_{11}}\,\frac{f'}{a'_{11}\,a'_{22}} = d \tag{2.48}$$

where the right hand side is known. Using (2.28), produces

$$\frac{f^2 + u_\infty^2 + v_\infty^2}{\sqrt{f^2 + v_\infty^2}} = d.$$

This leads to a quadratic equation in f^2 with two possible solutions

$$f^2 = \frac{1}{2}\,(d^2 - 2(u_\infty^2 + v_\infty^2) \pm d\,\sqrt{d^2 - 4u_\infty^2}\,). \tag{2.49}$$

We must choose the solution with the positive root in order to preserve the identity (2.13). This can be seen as follows. First, factor d out of the square root in (2.49) and rearrange the result to obtain

$$f^2 + u_\infty^2 + v_\infty^2 = \frac{1}{2}\,d^2\,(1 \pm \sqrt{1 - 4(u_\infty/d)^2}\,). \tag{2.50}$$

Using equations (2.6), (2.10), and (2.28) results in

$$\frac{u_\infty}{d} = a_{12}a_{22} = -\sin\theta\,\cos\theta = -\frac{1}{2}\sin 2\theta,$$

and hence

$$\sqrt{1 - 4(u_\infty/d)^2} = \cos 2\theta \quad (\text{where} \quad |\theta| < \pi/2).$$

Thus, equation (2.50) becomes

$$f^2 + u_\infty^2 + v_\infty^2 = \frac{1}{2}\,d^2\,(1 \pm \cos 2\theta\,).$$

Because $\frac{1}{2}\,d^2\,(1+\cos 2\theta\,) = (d\,\cos\theta)^2$ and $d\,\cos\theta = d\,a_{22} = f/a_{11}$ by (2.6), the desired identity (2.13) holds if we choose the + sign in (2.49). (If we choose the solution with the − sign, then (2.13) holds only if $\theta = \pm 45°$, in which case the two solutions are the same.)

Now that f is known, (2.26) through (2.28) can be used to compute the orientation angles ϕ, θ and the matrix coefficients a_{ij}, and (2.34) can be used to compute h. □

Chapter 3

Image Processing Algorithms

This section describes image processing algorithms for estimating the highway line vanishing point, straightening an image, and estimating the vertical image-to-highway scale factor. An application of these algorithms is referred to as the "calibration process." This chapter also describes algorithms for estimating traffic speed given a successful calibration.

Throughout this section let $\{I_\ell,\ \ell = 1, \ldots, L\}$ denote a sequence of time-tagged monochromatic (i.e., grayscale) images.

3.1 Calibration Process

There are four main phases in the calibration process: (1) highway line detection, (2) vanishing point estimation, (3) construction of image straightening *warp operator*, and (4) computation of the highway-to-image scale factor.

It is possible for the calibration process to fail at some stage. For example, the line detection algorithm may fail to detect enough highway lines, the vanishing point estimation algorithm may fail to converge and hence no warping operation can be defined, or the stripe detection algorithm may fail. In some cases these failures can be eliminated by adjusting certain control parameters.

3.1.1 Highway Line Detection

Highway line detection consists of thresholding an edge-detected background image and then applying a modified Hough/Radon transform.

1. *Compute background:* The "background map," $B = \frac{1}{L}\sum I_\ell$, is the average of the image sequence. Averaging has the effect of suppressing the traffic and enhancing the road lines and stripes.

2. *Compute gradient:* Compute the background image gradient, $G = (\partial_u B, \partial_v B)$ by convolving B with the Sobel kernels [1] K_u and K_v given by

$$K_u = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \quad K_v = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

   Thus, $\partial_u B = B * K_u$ gives an estimate at each point of the rate of change of intensity in B in the u direction (to the right). Similarly, $\partial_v B = B * K_v$ is an estimate of the rate of change of intensity in the v direction (downward).

   Compute the magnitude map $M = |G| = \sqrt{(\partial_u B)^2 + (\partial_v B)^2)}$ and the angle map $\Theta = \arg(G)$. Here, $-180 \leq \Theta(u, v) < 180$ is the angle that the gradient vector at (u, v) makes with the u-axis.

   Typically, $M(u, v)$ is "large" on the edges of high contrast features, in which case the gradient is perpendicular to the edge. The purpose of the next two steps is to determine a "largeness" threshold for detecting highway edges.

3. *Crop magnitude:* In this step, a rectangular subimage M'(the region of interest (ROI)) is extracted in the interior of M. The boundary of this subimage is chosen so that unwanted edge artifacts are excluded that could adversely affect the threshold level computed in the next step. (The idea here is that too many high magnitude non-highway edges give rise to a threshold that is too high to detect highway lines.) In this case, the strong text edges that appear at the top of M and the artificial vertical line on the left produced by the black stripe[2] ever present in our images are excluded. (See Figure 3.1.) Also, since we have no interest in detecting near horizontal edges, we zero the magnitude at any point in M' where the gradient angle is near $\pm 90°$. (We use a nearness threshold of $22.5°$.)

[1] These kernels often appear in the literature rotated by 180° degrees. This is evidently due to some confusion as to whether they are actually convolved with the image or simply applied as masks.

[2] This is a "feature" of the video capture card.

4. *Edge detect (threshold):* Compute an "optimal" threshold level for the cropped magnitude map M' by using Otsu's algorithm (Otsu 1979). This threshold divides the pixels into two classes in a way that maximizes their "between-class variance." (See also Fukunaga 1972.) Next, compute a thresholded image T, where a point in T has value 0 or 1 depending on whether the corresponding point in M' is above or below the threshold level.

Figure 3.1: Background image on the left, thresholded edge image on the right.

5. *Hough (Radon) transform:* Apply a modification of the standard Hough transform (Leavers 1992) which is described below. First, the theory of the standard Hough transform, sometimes referred to as the discrete Radon transform is reviewed.

   Recall that every straight line in the uv-plane can be expressed uniquely in the normal form

   $$p = u \cos\theta + v \sin\theta \tag{3.1}$$

   where $-90° \leq \theta < 90°$ is the angle of a unit normal vector pointing into the right half-plane, and $-\infty < p < \infty$ is the signed distance from the line to the origin. (See Figure 3.2.)

   The domain of a Hough map, H, consists of a discrete selection of points (θ, p). p is assumed to be an integer, $-w/2 \leq p < w/2$, where w is the width of T, and the angle discretization step size is assumed to be $d\theta = 180/w$. This gives a $w \times w$ square grid in the $\theta\, p$-plane. The value $H(\theta, p)$ is the number of nonzero

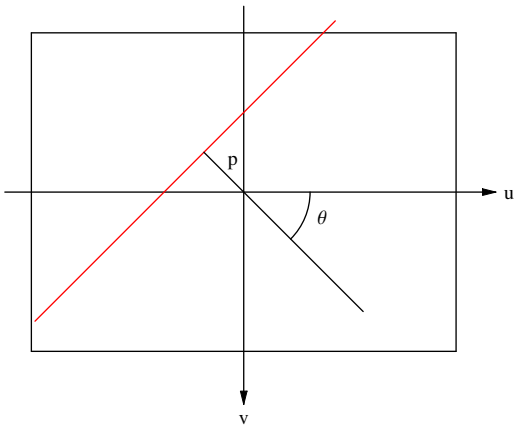

Figure 3.2: A line with parameters $\theta = 45°$ and $p < 0$.

points in T that satisfy the rounded version of (3.1):

$$H(\theta,\, p) = \sum_{(u,\, v)\in\ell} T(u,\, v) \tag{3.2}$$

$$\ell = \{(u,\, v) : p = \lfloor u\, \cos\theta + v\, \sin\theta + 1/2 \rfloor\}. \tag{3.3}$$

Large Hough counts indicate the presence of long lines. The Hough procedure for computing (3.1) is the following: for each point $(u,\, v)$ for which $T(u,\, v) = 1$ and for each discrete value of θ compute $p = \lfloor u\, \cos\theta + v\, \sin\theta + 1/2 \rfloor$ and increment the count $H(\theta,\, p)$.

A modification of the standard Hough algorithm is made that takes into account the image gradient angle. It produces two maps: $H_1(\theta,\, p)$ for "leading edge" lines and $H_2(\theta,\, p)$ for "trailing edge" lines. The reason for doing this is that each highway boundary line is usually edge detected as a "ribbon" (i.e., two sets of points with approximately opposite gradients). The matrix shown below is a portion of the masked gradient angle shown in Figure 3.3:

$$\begin{pmatrix}
128 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
124 & 123 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 122 & 123 & 124 & 0 & 0 & 0 & 0 \\
-56 & 0 & 121 & 123 & 127 & 0 & 0 & 0 \\
-56 & -57 & -55 & 0 & 124 & 122 & 122 & 0 \\
0 & 0 & -55 & -56 & 0 & 0 & 120 & 123 \\
0 & 0 & 0 & -53 & -52 & 0 & 0 & 120 \\
0 & 0 & 0 & 0 & -52 & -56 & -56 & 0
\end{pmatrix}$$

Reading from left to right shows that when the gradient magnitude rises above threshold, the gradient points upward to the right at an angle near $-55°$ (the leading edge of the stripe). It then points downward to the left at an angle near $125°$ (the trailing edge of the stripe) before the magnitude falls below threshold. Because of discretization errors, points on both sides of a ribbon can contribute to the same Hough count, indicating a line that is not actually present in the image. The following modification to the Hough algorithm prevents this undesirable feature.

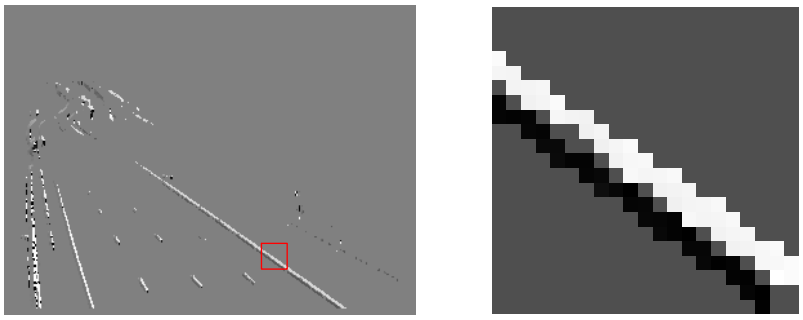

Figure 3.3: Point-wise product of T and Θ on the left, close up of smaller region on right.

Define the "leading edge" value $H_1(\theta, p)$ as in (3.2) but additionally require that the points on the line (3.3) have a gradient angle near θ:

$$|\Theta(u, v) - \theta \mod 360| < \Delta\theta$$

(say $\Delta\theta = 22.5°$). Similarly, define the "trailing edge" $H_2(\theta, p)$ in terms of the opposite gradient angle,

$$|\Theta(u, v) + 180 - \theta \mod 360| < \Delta\theta.$$

Highway boundary lines should register at least once in each Hough map with approximately the same line parameters.

6. *Line detect:* For each of the two Hough maps, find the set of lines parameters, (θ, p) for which the Hough count is greater than a specified threshold (say $H_i(\theta, p) > 100$ for an image of half height 120 pixels). Order the set lexicographically and partition it into subsets of angular width no greater than a

specified limit (say 5°). Each subset may contain more that one line as a result of discretization, pixel size, road curvature effects, and other errors. Select the line with greatest count from each subset. Figure 3.4 shows a leading edge Hough map and the detected lines.

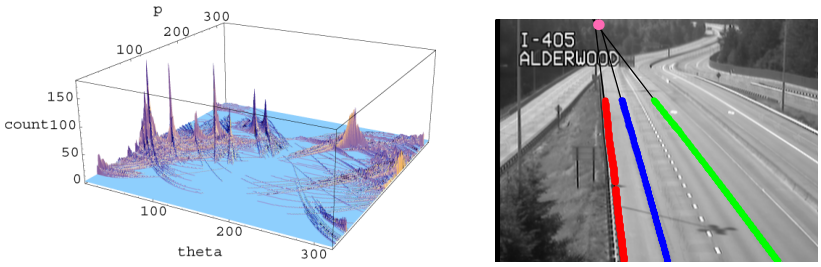

Figure 3.4: A 3-D leading edge Hough plot on left, detected lines on right.

It is possible that in some surveillance scenes long lines in the image will be detected that are not highway lines. For example, a tall light pole in front of the camera may be detected. Sometimes undesirable divergent ramp lines are also detected. By limiting the Hough domain θ values, these lines may be suppressed. Spurious lines are also removed with a sub-sampling technique used in the vanishing point estimation step described below.

3.1.2 Vanishing Point Estimation

The vanishing point (common intersection) of the highway lines is estimated by using a linear least squares method. (Compare with Masoud et al. 2001.) The maximum RMS error in the solution must not exceed a specified threshold, and some constraints are imposed on the number of lines and their position in the image.

1. *Constraints.* At least three lines are required,

$$\{p_i = u\,\cos\theta_i + v\,\sin\theta_i,\quad i = 1, \ldots m\} \qquad m \geq 3. \tag{3.4}$$

Initially, all of the lines detected in the preceding step are used. In addition, the intersection of these lines with the horizontal baseline of the ROI, $v = v_{\max}$, must span an interval of length at least one third of the width of the ROI. (Note that the u-coordinates of the intersection points have the form

$u_i = p_i \sec\theta_i - v_{\max} \tan\theta_i$, so that the span interval has a length equal to $\max\{u_i\} - \min\{u_i\}$.)

2. *Least Squares.* Write the system of equations (3.4) in matrix form, $b = M\,w$, where

$$b = \begin{pmatrix} p_1 \\ \vdots \\ p_m \end{pmatrix} \qquad M = \begin{pmatrix} \cos\theta_1 & \sin\theta_1 \\ \vdots & \vdots \\ \cos\theta_m & \sin\theta_m \end{pmatrix} \qquad w = \begin{pmatrix} u \\ v \end{pmatrix}. \tag{3.5}$$

Then, the least squares solution to the system is given by

$$w = M^T\,M\,M^T\,b, \tag{3.6}$$

and the RMS error is

$$\epsilon = \frac{|M\,w - b|}{\sqrt{m}}. \tag{3.7}$$

If this error is less than a specified threshold (say two pixels), then the solution is assumed to be an estimate of the true vanishing point.

3. *Recourse.* If the error is too large, and there are more than three lines, a subsampling procedure is used. Select a sample size, k, at which $3 \le k < m$. For each subset of lines of size k[3] try to solve for the vanishing point by using the procedure described above (steps 1 and 2). If solutions exist, choose one based on some criterion, such as least RMS error or largest baseline span. If no solutions are found, then try again with a smaller value of k. If this is not possible, the calibration process fails.

4. *Refinement of Solution.* The line parameters contributing to a solution found in step 2 lie on the discrete grid that is the Hough domain. As suggested by Leavers (1992), the parameter values can be improved by a line fitting technique. Leavers (1992) suggested the following procedure: given a line with parameters $(\theta,\, p)$ in the Hough domain, (1) collect a band of points around this line, (2) rotate the band so it is near horizontal, (3) compute an ordinary least squares line fit, and (4) use the result to correct $(\theta,\, p)$.

[3]Generating subsets of size k from a set of size m amounts to generating increasing subsequences $(i_1,\, i_2, \ldots, i_k)$ of length k from the sequence $(1,\, 2, \ldots, m)$. This is easy to do recursively, starting with $(1,\, 2, \ldots, k)$. Indeed, having generated $(i_1,\, i_2, \ldots, i_k)$, the next subsequence in the lexicographical ordering is obtained as follows: find the last subscript $j \le k$ such that $i_j < m - k + j$ and form the subsequence $(i_1, \ldots, i_{j-1},\, i'_j, \ldots, i'_k)$, where $i'_j = i_j + 1$, $i'_{j+1} = i'_j + 1, \ldots$.

Here, a more direct method is used. For each selected line (θ_i, p_i), form the "band" b_i of above-threshold points (u, v) such that $|u \cos\theta_i + v \sin\theta_i - p_i| < \Delta p$ (say $\Delta p = 3$) and such that the gradient angle $\Theta(u, v)$ is consistent with the Hough image in which the line was found. Using the method of *Total Least Squares* (see Appendix A), find the line with parameters $(\tilde{\theta}_i, \tilde{p}_i)$ that minimizes the sum of squares of "true" perpendicular distances

$$J(\theta, p) = \sum_{(u,v)\in b_i} (u \cos\theta + v \sin\theta - p)^2. \tag{3.8}$$

Now compute a final estimate for the vanishing point (u_∞, v_∞) in terms of the $(\tilde{\theta}_i, \tilde{p}_i)$'s by using the least squares method described in step 2 above.

3.1.3 Straightening Warp Operator

At this stage our objective is to define an operator that can "warp" an image region inscribed in a triangle with the apex at the vanishing point onto a prescribed rectangle. Let $p_\infty = (u_\infty, v_\infty)$ denote the vanishing point estimate computed in the previous step. Let $v = v_{\max}$ define the horizontal baseline of the ROI defined in the cropping step discussed above, and let $p_1 = (u_1, v_{\max})$ and $p_2 = (u_2, v_{\max})$, two points on this baseline such that $u_1 < u_2$ (for example, the intersections with the baseline of two extreme lines found in section 3.1.2). The objective is to warp a trapezoidal region inscribed in the bottom of the triangle with vertices p_∞, p_1, p_2 onto a rectangle that has a prescribed width W and height H. (See Figure 3.5.) Generally, the width should be $W \approx u_2 - u_1$ and the height, H, should be 512 or 1024.[4]

The definition of the warp operator proceeds as follows.

1. *Straightening transformation.* Define a straightening transformation as described in section (2.5). Choose a nominal depression angle (say $\phi' = -10°$) and set the corresponding focal length $f' = v_\infty \cot\phi'$ and orientation matrix $A' = A(f', u_\infty, v_\infty)$. Initially, set the height parameter $h' = 1$. Apply transformation (2.30) to the baseline vertices p_1 and p_2 to obtain corresponding points $q_1 = (x_1', y_1')$ and $q_2 = (x_2', y_2')$. Redefine the height parameter to be

[4]The correlation methods used later are based on an implementation of the Discrete Fourier transform that requires data size to be power of 2.

$h' = W/(y'_2 - y'_1)$. This completes the definition of the straightening transformation.

2. *Rectangle.* Recompute the coordinates of the points q_1 and q_2. It follows now that $y'_2 - y'_1 = W$. Let $y'_{\min} = y'_1$ and $y'_{\max} = y'_2$. Set $x'_{\min}$ to the maximum of x'_1 and x'_2 and set $x'_{\max} = x'_{\min} + H$. Then the rectangle with corners

$$(x'_{\min},\, y'_{\min}),\ (x'_{\min},\, y'_{\max}),\ (x'_{\max},\, y'_{\min}),\ (x'_{\max},\, y'_{\max}) \tag{3.9}$$

has the prescribed dimensions. Also, the inverse of the straightening transform (2.29) maps this rectangle onto a trapezoid inscribed in the triangle.

3. *Warp operator.* The straightening warp operator is defined as follows. Let I denote a source image and initialize an empty destination image S_I of width W and height H. For each point in the domain of S_I with pixel coordinates $(c,\, r)$, form the point

$$(x',\, y') = (x_{\max} - r,\, y_{\min} + c), \tag{3.10}$$

which lies in the rectangle (3.9). Use the inverse of the straightening transform (2.29) to map this to a point $(u,\, v)$ in the domain of I. Define $S_I(c,\, r)$ by the formula

$$S_I(c,\, r) = I(u,\, v). \tag{3.11}$$

Use bilinear interpolation (see Ballard and Brown 1982) to compute the right hand side since u and v are not integral. Figure 3.6 is a sample straightened background image.

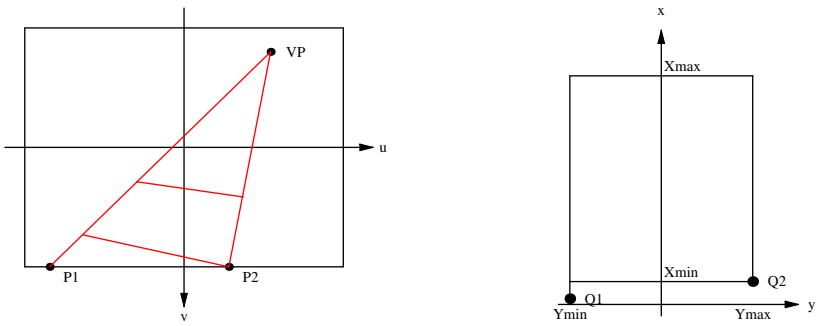

Figure 3.5: Warp geometry.

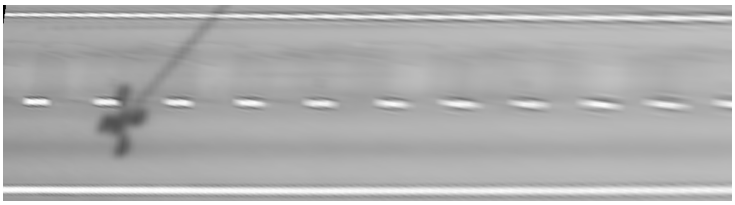

Figure 3.6: A straightened background image (rotated 90°).

3.1.4 Stripe detection and scale factor

The final stage of the calibration process is estimation of the vertical highway-to-image scale factors. For simplicity, we assume that there is no camera misalignment, and, hence, by Theorem (1)(b) the scale factor is constant across the image.

The process searches a straightened background image for highway stripes, estimates the image stripe period, and depends on *a priori* knowledge of true stripe periods to estimate the scale factor β_{11}. The algorithm has the following steps.

1. *Straighten background.* Rescale the background map, B, so that the intensities lie in the unit interval $[0, 1)$ and compute the straightened background image, S_B, by using the warp operator described above. Use column-row $(c,\, r)$ pixel coordinates for points in S_B, where $0 \leq c < W$ and $0 \leq r < H$.

2. *Autocorrelate.* For each column c of S_B, compute the autocorrelation

$$C_c(k) = \sum_r (S_B(c,\, r+k) - \mu_c)\,(S_B(c,\, r) - \mu_c) \tag{3.12}$$

$$= \sum_r S_B(c,\, r+k)\, S_B(c,\, r) - H\,\mu_c, \tag{3.13}$$

   where

$$\mu_c = \frac{1}{H} \sum_r S_B(c,\, r) \tag{3.14}$$

   is the mean value for column c.[5] The summation in (3.13) is circular in the sense that row index $r + k$ is taken to be modulo H. The reason that the autocorrelation is computed is that it is useful for detecting periodic structure. Indeed, (3.13) should be regarded as the inner product of the data in column c

[5] We use a Fast Fourier Transform (FFT) technique to compute the autocorrelation. First FFT the column and set the amplitude at frequency 0 equal to 0 (which effectively removes the mean). Next, square the amplitude at each frequency and inverse FFT the result. This gives the autocorrelation.

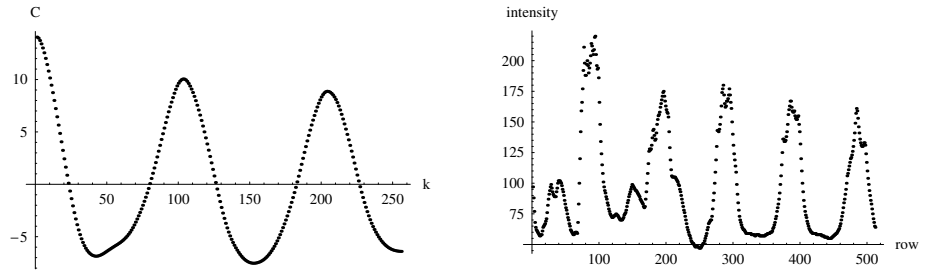

Figure 3.7: Autocorrelation on the left, line of stripes on right.

with a copy of itself shifted upward k steps. If the data are periodic with period K, then $C_c(k)$ will assume its maximum value exactly when k is a multiple of K. (This follows from the Schwartz inequality for inner products.)

Because C_c is symmetric about the midpoint, k is limited to the interval $0 \leq k < H/2$. Figure 3.7 illustrates the autocorrelation for a line in a straightened background where there are lane stripes.

3. *Find maxima.* For each column c, $C_c(k)$ is maximum when $k = 0$. If enough stripes lie on column c, then the stripe period can be measured by locating the value of the shift $k = k_{max}(c)$ corresponding to the first local maximum value beyond $k = 0$. The following stripe detection criterion is used: values $C_c(k)$ must first decrease to a local minimum below some negative threshold (say -2), then increase to a local maximum above some positive threshold (say 2), and then decrease again to a local minimum below the negative threshold.

4. *Image stripe period.* Adjacent columns that satisfy the condition above are grouped together and deemed to represent a line of stripes on the road. The image stripe period is set for this line equal to the shift $k_{max}(c)$, which gives the largest value for $C_c(k_{\max}(c))$. If there is more than one line of stripes, one of them with a period corresponding to the strongest autocorrelation value is selected.

5. *Scale factor.* The vertical highway-to-image scale factor β_{11} is obtained by dividing the image stripe period by the known true stripe period.

3.2 Traffic Speed Estimation

By using cross correlation of vertical line intensities in successive pairs of straightened images, moving traffic can be detected and speed and direction of motion (receding from or approaching toward the camera) can be measured. By using the calibrating scale factor, line by line measurements of traffic speed are obtained. Clustering this line data gives average lane speeds. Note that this method of speed estimation does not require the tracking of individual vehicles, and, hence, does not generally suffer from the splitting/merging target occlusion problems faced by a vehicle tracker when traffic is congested.

Usually, a sequence is collected of about 20 frames grabbed at rate of 5 or 10 Hz. The sequence may be the same one used for the calibration process or one collected at some later time. In the latter case, the camera focal length and orientation have to be checked for changes since the time of the calibration. To do this, a reference copy, T_0, of the threshold image computed at calibration time is saved. Then, the threshold image, T, of the new image sequence is computed and matched with T_0; if

$$\frac{T \cdot T_0}{|T|\,|T_0|} < \frac{1}{2}, \tag{3.15}$$

we must try to recalibrate (or start over with a different camera selection).

Now, given a sequence of time-tagged images $\{I_\ell\}$ and a background B, the following operations are performed for each subscript ℓ.

1. *Straighten.* Form normalized images $I'_\ell = I_\ell - B$ and $I'_{\ell+1} = I_{\ell+1} - B$ with intensity values rescaled to lie in the unit interval $[0, 1)$. Apply the warp operator to get straightened images $S_\ell = S_{I'_\ell}$ and $S_{\ell+1} = S_{I'_{\ell+1}}$. A pair of straightened images (no normalization) are shown in Figure 3.8.

2. *Cross correlate.* For each column index c, compute the cross-correlations

$$CC_c(k) = \sum_r (S_\ell(c,\, r+k) - \mu_{c,\,\ell})\,(S_{\ell+1}(c,\, r) - \mu_{c,\,\ell+1}) \tag{3.16}$$

$$= \sum_r S_\ell(c,\, r+k)\, S_{\ell+1}(c,\, r) - H\,\mu_{c,\,\ell}\,\mu_{c,\,\ell+1}, \tag{3.17}$$

where $\mu_{c,\,\ell}$ and $\mu_{c,\,\ell+1}$ are the means for the c^{th} columns of S_ℓ and $S_{\ell+1}$, respectively.[6]

[6] As in the previous section, an FFT technique is used to compute (3.17). Apply the FFT to

The reason the cross correlation is computed is that it is useful for detecting translation in structure between image frames. Indeed, (3.17) should be regarded as the inner product of the data in column c of S_ℓ shifted upward k steps with the corresponding unshifted column of $S_{\ell+1}$. Figure 3.9 at the left shows the peak in the cross-correlation function that identified the distance traveled between frames shown in figure 3.8. In the right of figure 3.9, rows from the sequential images are aligned by the offset identified from the cross-correlation function, demonstrating the similarity of the signals.

3. *Threshold.* For each column index c, find the shift $k = k_{max}(c)$ giving the maximum cross-correlation value. Threshold this array of numbers, that is, form the list of pairs $\{D_c = (c,\, k_{max}(c))\}$ such that the correlation value $CC_c(k_{\max}(c))$ is above a given threshold. (A nominal threshold of 6 is used but may be lowered if the contrast is poor.) Each D_c represents a detection of traffic motion in the corresponding column. If $k_{\max}(c) < H/2$, the motion is upward in the image, and, hence, away from the camera; otherwise, motion is toward the camera.

4. *Cluster.* Order the list of detections by column index and partition into clusters according to the following criteria: two consecutive detections D_c and D'_c are in the same cluster if and only if $c' - c$ and the shift difference $|k_{\max}(c') - k_{\max}(c)|$ are both small. (smallness limits of 5 and 10 are used respectively.) To help reduce false alarms, any clusters that are small (say fewer than 5 detects) are discarded. Remaining clusters correspond to a lane of traffic. However, there may be more than one cluster per lane.

each of the two corresponding columns, zero the amplitudes at frequency 0, multiply the first by the complex conjugate of the second, and then inverse FFT the result. This gives the cross-correlation.

Figure 3.8: Sequence of straightened images, frame 6 on left and frame 7 on right. Interframe time is 0.5 seconds.

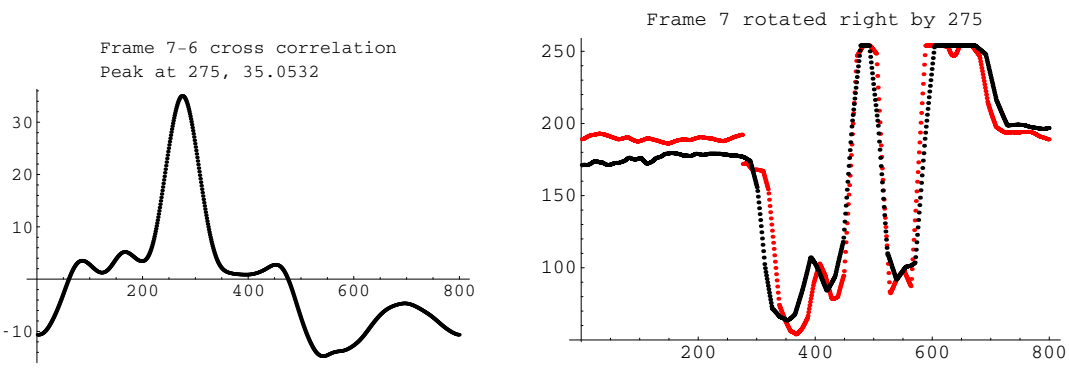

Figure 3.9: Cross-correlation to estimate travel distance on the left, cross-section from frame 7 aligned with cross-section from frame 6 on the right.

5. *Report.* Finally, form a *cluster report* for each cluster of detections. Each report includes the following:

t	time-tag of image ℓ
Δt	time difference between images ℓ and $\ell+1$
n	number of detections in the cluster
w	cluster width $(c_{\max} - c_{\min})$
$\bar{c}$	column mean
$\bar{k}$	mean of the $k_{\max}$-values
$\sigma^2(k)$	variance of the $k_{\max}$-values
v	mean speed
$\sigma^2(v)$	variance of speed

Speed is computed as follows. Let $\alpha = 1/\beta_{11}$ denote the vertical image-to-highway scale factor. If $\bar{k} < H/2$, set $v = \alpha\,\bar{k}/\Delta t$; otherwise, set $v = \alpha\,(\bar{k} - H)/\Delta t$. In the latter case, the speed is negative, indicating motion toward the camera. In either case, let $\sigma^2(v) = \alpha^2\,\sigma^2(k)/\Delta t^2$.

This concludes the processing for a pair of successive images.

The processing described above is performed for all image pairs I_ℓ and $I_{\ell+1}$, $\ell = 1, 2, \ldots, L-1$, and a list is formed of all reports generated at each stage. Also

generated is a summary report of overall speed statistics for approaching and receding traffic. These statistics consist of the total detection count, n_T, overall mean speed, v_T, and variance $\sigma^2(v_T)$. These are computed with a recursive procedure applied to the list of reports. It is not necessary to collect up all detections generated from all pairs in the image sequence to compute the overall statistics. The recursion depends on the following combination rule.

Lemma 1. *Let A be a set of numbers of size m, mean μ, and variance σ^2. Let A_1, A_2 be a partition of A such that A_1 has size m_1, mean μ_1 and variance σ_1^2, and A_2 has size m_2, mean μ_2, and variance σ_2^2. Let $\lambda_1 = m_1/m$ and $\lambda_2 = m_2/m$. Then*

$$\begin{aligned} m &= m_1 + m_2 \\ \mu &= \lambda_1\,\mu_1 + \lambda_2\,\mu_2 \\ \sigma^2 &= \lambda_1\,\sigma_1^2 + \lambda_2\,\sigma_2^2 + \lambda_1\,\lambda_2\,(\mu_1 - \mu_2)^2. \end{aligned}$$

To compute the overall statistics for receding traffic (approaching traffic is handled in a similar way), let P_r be the set of reports with positive speed. Suppose there are k such reports, and form the list of triples $(n_j,\, v_j,\, \sigma^2(v_j))$, $j = 1, \ldots, k$.

If $k = 1$, let $(n_T,\, v_T,\, \sigma^2(v_T)) = (n_1,\, v_1,\, \sigma^2(v_1))$, and that is all. Otherwise, recursively compute $(n'_T,\, v'_T,\, \sigma^2(v'_T))$, the overall statistics corresponding to the first $k-1$ reports, and apply the lemma:

$$\begin{aligned} \lambda_k &= n_k/(n'_T + n_k) \\ \lambda' &= n'_T/(n'_T + n_k) \end{aligned}$$

$$\begin{aligned} n_T &= n'_T + n_k \\ v_T &= \lambda'\,v'_T + \lambda_k\,v_k \\ \sigma^2(v_T) &= \lambda'\,\sigma^2(v'_T) + \lambda_k\,\sigma^2(v_k) + \lambda'\,\lambda_k\,(v'_T - v_k)^2 \end{aligned}$$

3.2.1 Accuracy Considerations

The standard deviation of traffic speed given in a cluster report is $\sigma(v) = \alpha\,\sigma(k)/\Delta t$, where $\alpha = 1/\beta_{11}$ is the image-to-highway scale factor. If there is no error in α or Δt, then this may be regarded as the accuracy of the reported speed. Assuming this for the moment, and using typical values $\Delta t = .25$ sec, $\sigma(k) < 1$ pixel, and $\alpha < 1$ ft/pixel, then $\sigma(v) < 4$ ft/sec < 3 mph.

In general, for a point moving on the highway (say receding from the camera) differential calculus gives the formula

$$\begin{aligned} d\,v = d(\alpha\,\frac{\bar{k}}{\Delta t}) &= \frac{\bar{k}}{\Delta t}\,d\,\alpha + \alpha\,(\frac{\Delta t\,d\,\bar{k} - \bar{k}\,d\,\Delta t}{\Delta t^2}) \\ &= (\frac{d\,\alpha}{\alpha} - \frac{d\,\Delta t}{\Delta t})\,v + \alpha\,\frac{d\,\bar{k}}{\Delta t} \\ &= (\frac{d\,L}{L} - \frac{d\,\ell}{\ell} - \frac{d\,\Delta t}{\Delta t})\,v + \alpha\,\frac{d\,\bar{k}}{\Delta t} \end{aligned} \tag{3.18}$$

where $\alpha = L/\ell$, L is the stripe period on the highway, and ℓ is the image stripe period. A first order estimate of the variance is

$$\sigma^2(v) = \left((\frac{\sigma(L)}{L})^2 + (\frac{\sigma(\ell)}{\ell})^2 + (\frac{\sigma(\Delta\,t)}{\Delta\,t})^2\right)\,v^2 + (\alpha\,\frac{\sigma(k)}{\Delta\,t})^2. \tag{3.19}$$

Assume that the stripe period, L, is approximately 40 feet with an error on the order of 1 foot; assume that the image stripe period, ℓ, is greater than 40 pixels with an error on the order of one pixel; and assume that the timing error is less than five milliseconds. Also, assume as before that $\Delta t = .25$ sec and $\sigma(k) < 1$ pixel. Then

$$\begin{aligned} \sigma(v) &< \sqrt{\left((\frac{1}{40})^2 + (\frac{1}{40})^2 + (\frac{.005)}{.25})^2\right)\,v^2 + 3^2} \\ &< .04\,v + 3 \ \text{ mph.} \end{aligned} \tag{3.20}$$

This result is for speed in the highway plane: the vertical image-to-highway scale factor α is measured at ground level. However, the cross-correlation lags, $\bar{k}$, may correspond to vehicle points that are several feet above the road. Indeed, during daylight hours the correlation lags generally correspond to features such as the bumper, hood, or roof that may be 2 to 5 feet above the road, rather than shadows cast on the road. It may, therefore, be necessary to introduce a correction for α that corresponds to an appropriate height above the road.

This correction can be made with an estimate of the height, h, of the camera above the road. Indeed, because α is directly proportional to h, as shown in the first formula of (2.34), we can write $\alpha = c\,h$, where c is independent of h. If a scale factor α' appropriate for an elevation δh above the road is necessary, then let $\alpha' = c\,(h - \delta h) = \alpha\,(1 - \delta h/h)$. By using α instead of α' speed is overestimated with a relative error of

$$\frac{v - v'}{v'} = \frac{\alpha\,\bar{k}/\delta t - \alpha'\,\bar{k}/\delta t}{\alpha'\,\bar{k}/\delta t} = \frac{\delta h/h}{1 - \delta h/h}. \tag{3.21}$$

With $\delta h = 3$ feet and $h = 50$ feet, this relative error is approximately 6 percent.

Chapter 4

Application

A distributed application has been developed in Java for traffic video image acquisition, camera calibration, and speed estimation. A server program, *Capture*, (running in the WSDOT local network) has access to the CCTV camera video switch that grabs and serves video image sequences. A client application program, *AutoCalSpd*, (in the University of Washington ITS local network) obtains and processes the image sequences. Communication between server and client is through a "controlled proxy" program, which prevents unauthorized users from accessing the server.

This section discusses the implementation of the image processing client application *AutoCalSpd*. The application has two main parts: an *image processor* that performs the calculations for calibration and speed estimation and a *graphical user interface* that allows camera selection, parameter tuning, and visibility into the various image processing stages. There is an interactive mode of operation in which the user supervises the calibration, and there is an automatic mode in which speed data are continuously recorded into a file.

4.1 Graphical User Interface

The user interface is implemented with the Java Swing GUI toolkit. It consists of three principle components: a control panel, an image display desktop panel, and a camera selector panel. A screenshot of the GUI is shown in Figure 4.1.

When the application is launched, a default camera is selected (# 12 at I-5 and NE 45^{th} street) and shown highlighted in the camera selector panel. A 320×240 snapshot image of the field of view is shown in the display panel. All calibration

Figure 4.1: Screen shot of the CameraView application.

parameters are set at their default values and the "try calibrate" toggle is set, but no calibration is attempted until the "load images" button is pressed. The "try speed" toggle is disabled because no image-to-highway scale factor is known yet. The user may switch cameras, increase the image size to 640×480, and preview images at any time. For calibration purposes, a camera should be selected with the following properties:

- The camera should surveil approaching and/or receding traffic, that is, it should point downward and along the highway rather than across it.
- The view should show straight highway lines in the bottom half of the image as well as lane stripes.
- The view should be mostly unobstructed by overpasses, large overhead signs, or divergent lanes of traffic.[1]

Under these conditions a successful calibration is likely.

To effect a calibration, the user selects the image size, the number of images to acquire, and the frame rate, and then presses the "load images" button. A request is sent to the *Capture* server program with the three aforementioned parameter values, as well as other (currently uneditable) values for brightness, contrast, compression type, and color model (grayscale). The response from the server is a sequence of time-tagged images, or frames, the first of which is presented in a "captured frames" viewer in the image display desktop. The user can cycle through the images by using "spinner" buttons attached to the bottom of viewer. Because the "try calibrate" toggle button is selected, a calibration process is attempted. The user may view images produced at various stages of the process by pressing labeled buttons under the desktop. This is useful for confirming the validity of a calibration or diagnosing and maybe correcting a failure.

If the calibration fails, an error message pops up indicating where the failure occurred: either no vanishing point was found or no stripes were detected. The desktop may be used as a diagnostic tool in these cases. Visibility into the image

[1]This condition may be relaxed in some cases by editing certain control parameters during the calibration process, for example moving the baseline of the ROI further up into the image, narrowing the angular limits of the Hough transform, or increasing the height of the straightened image.

processing stages prior to the failure point may suggest parameter changes that could lead to a successful calibration. For example, if the vanishing point could not be found, it is useful to view the "lines" image, which shows detected lines superimposed on the background image. If too many or too few lines are shown, the user may edit the Hough parameters and force a reactivation of the calibration process starting at the Hough stage. If stripes could not be found, it is useful to view the "straightened background" image. This may show that not enough stripes are present, in which case the length of the straightened image may be increased and the calibration process reactivated beginning with the straightening step. If the stripes are faint, the stripe detection thresholds may be lowered and the calibration process reactivated beginning with the stripe detection step.

If the calibration process does not fail, a status message under the desktop indicates "successful calibration," and the "try speed" toggle button is enabled. However, before activating a speed computation, the user should view the "stripes" image to double check that road stripes were actually found rather than some other periodic structure such as construction barrels. (Also, if an insufficient number of image frames are collected, traffic can appear as a periodic structure in the background.)

Selecting "try speed" starts a speed estimation process using default values for the correlation threshold and the stripe period (40 feet). When this process completes, a status message under the desktop indicates "speeds computed," and the current approaching and receding traffic speed estimates are displayed also just below the desktop. If no speeds are shown, then either there is no traffic (which can verified by spinning through the "captured frames" viewer) or the cross-correlation threshold is too high. The user can edit this parameter and reactivate the speed computation. If the speeds appear unreasonable, then the default stripe period may be wrong (some highway stripes are spaced at 12- and 15-foot intervals).

Once satisfied with the camera calibration and speed parameter settings, the user may continue interactively to load images, and as long as "try speed" is selected, speeds will be computed. The automatic mode of operation may be enabled by pressing the "record" button. Then the program will repeatedly load images, compute speed reports, and append them to a file. As indicated in section 3.2, a test is performed on each cycle to determine if the camera calibration (scale factor and

straightening transformation) is still valid. If not, a popup alerts the user that action needs to be taken: either select "try calibrate" or select a new camera and start over.

4.2 Image Processor

The image processor is the heart of the *AutoCalSpd* application. It consists of an image processing *graph* (see Figure 4.2) whose nodes implement the various algorithms described in section 3 and methods for manipulating the graph. In particular, the processor manages the loading of source images and the *rendering* of the graph according to the controls and parameter settings imposed by the user. The processor can be in one of two states, *calibrate* or *compute speed*, and regulates rendering of the graph accordingly.

The image processor is implemented in Java and relies heavily on the Java Advanced Imaging (JAI) application programming interface.[2] The JAI provides a set of basic image processing operators and a framework for defining and registering custom operators. Every operator stores an operation name, a *ParameterBlock* containing sources and parameters, and a *RenderingHints*, which contains image rendering hints such as image size and format.

Programming in JAI generally involves constructing an image processing chain, or more generally, a directed acyclic graph (DAG) whose nodes are operators. This is useful in that a chain or DAG may be manipulated dynamically and rendered multiple times. Thus, for example, the same chain of operations may be applied to different images, or the parameters of certain operations in a chain may be modified interactively. It is important to note that image rendering adheres to the *pull* model, that is, a node is rendered only when there is a request for actual pixel data.

4.2.1 The DAG

Figure 4.2 shows the basic DAG for the image processor. Operators at levels (1) through (5) are provided with the JAI distribution, while operators at levels (6) through (10) are custom. The graph is constructed during the initialization phase of the program. The operators are created and linked together in top down, left to

[2]On-line documentation for the JAI is available from Sun. See *Programming in Java Advanced Imaging*. http://docs.sun.com/app/docs/doc/806-5413-10

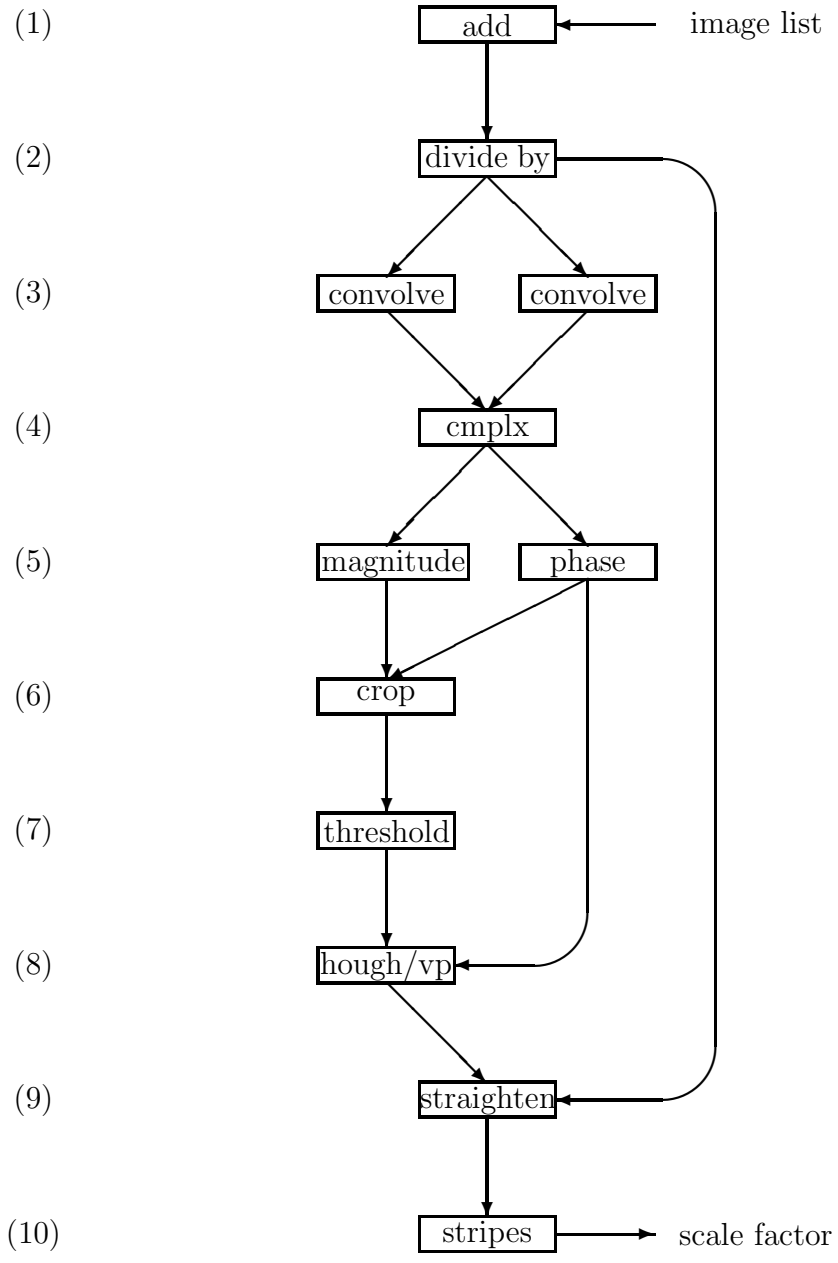

Figure 4.2: Image processing graph.

right order with default parameter settings. Once the DAG has been constructed, the user can request the processor to load source image data or make changes to various operator parameters.

Operators (1) and (2) taken together implement the "background" algorithm discussed in section 3.1.1. The image layout parameter for the "add" operation specifies that the input data type (byte) should be cast to type (double) so that the correct sum is computed. The parameter for the "divide by" operator must be set to the length of the input image list.

Operators (3) through (5) implement the algorithm for computing the gradient in polar form. Parameters for the two "convolve" operators are the appropriate Sobel masks for computing the gradient components in Cartesian form. The "cmplx" operator simply combines the two components into a single complex image, while the "magnitude" and "phase" operators compute the polar components of the gradient.

The "crop" operator (6) implements the "crop magnitude" algorithm described in section 3.1.1. Parameters are the bounds of the rectangular ROI and the gradient angle threshold.

The "threshold" operator (7) implements the "edge detect (threshold)" algorithm described in section 3.1.1. Recall that the threshold level is computed automatically with Otsu's method. The parameters for this operator consist of two switches: one to enable double thresholding and one to enable non-maximal suppression (See Sonka et al. 1999.). Both of these switches are currently off.

The "hough/vp" operator (8) implements the algorithms for computing Hough maps and the highway vanishing point detection algorithm of section 3.1.2. Parameters consist of the Hough threshold, two domain angle limits, and a "no verticals" switch. This operator is unique in that its imagery is not used downstream. It computes the vanishing point and baseline needed by the "straighten" operator, which is its sink.

The "straighten" operator (9) constructs the straightening warp discussed in section 3.1.3. This depends on the vanishing point and baseline computed above. The warp is applied to the background image computed at level (2). Parameters are the depression angle, ϕ, and the height of the straightened image (512 or 1024). Because image data are stored in row major order and autocorrelation is to be performed on

columns, the image is rotated 90° for ease of data access.

The "stripes" operator (10) implements the autocorrelation algorithm for determining the scale factor discussed in section 3.1.4. Parameters are the upper and lower autocorrelation thresholds for stripe detection.

4.2.2 Managing the DAG

The image processor supports a number of methods for user interaction with the DAG.

1. *loadImages.* This method sends a user-specified request to the image server for raw image data. The response includes time-tagged byte arrays of gray-scale image data. These data are then used to construct JAI image data structures called *TiledImages*. The list of images is set as the source for the background "add" operator, and its length is set as the parameter for the background "divide by" operator. No image rendering is performed as a result of these changes. However, all renderings and associated computed values in the graph are cleared.

2. *setParameters.* Methods are provided for changing the parameters for various operator nodes. No image rendering is performed, but the current rendering and any computed values are cleared. In addition, renderings and computed values associated with downstream nodes are cleared.

3. *calibrate.* This method implements the calibration process described in section 3.1. It starts by accessing the vanishing point and baseline from the Hough operator. This forces a rendering of nodes up to and including the Hough node, if they have not already been rendered. If the vanishing point could not be computed, an exception is thrown and caught by the GUI. Next, the calibrate method accesses the image stripe period from the stripes operator. This forces computation of the straightening warp transformation, if it has not already been computed, and rendering of the straightened background image, if it has not already been rendered. An exception is thrown if the image stripe period could not be computed. If the calibration process is successful, the calibrating scale factor is computed. The straightening warp is saved for later use, as well as copies of the background and threshold images.

4. *computeSpeed.* This method implements the algorithm discussed in section 3.2 and is only invoked if calibration has been successful. It first gets the rendering from the threshold operator and computes the expression (3.15). This forces a rendering of the nodes up to and including the threshold node, if they have not already been rendered. If the inequality holds, then a recalibration is required and an exception is thrown. Otherwise, speeds are computed by using the straightening warp, background image, and the scale factor computed above.

References

[1] Ballard D. H. and Brown C. M., *Computer Vision*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982

[2] Cathey F.W. and Dailey D.J., *One-Parameter Camera Calibration for Traffic Management Cameras*, Proceedings of the IEEE 7th International Conference on Intelligent Transportation Systems, Washington D.C., 4-6 October, 2004

[3] Fukunaga K., *Introduction to Statistical Pattern Recognition*, Academic Press, 1972

[4] Leavers V. F., *Shape detection in computer vision using the Hough transform*, Springer Verlag, London, New York, 1992

[5] Masoud O, Rogers S., Papanikolopoulos, N.P., *Monitoring Weaving Sections*, ITS Institute, Univ. Minnesota, Minneapolis, CTS 01-06, Oct. 2001

[6] Otsu N., *A threshold selection method from gray-level histograms*, IEEE Transactions Systems, Man, and Cybernetics, Vol. SMC-9, 1979.

[7] Sonka M., Hlavac V. and Boyle R., *Image Processing, Analysis, and Machine Vision* PWS Publishing, Brooks/Cole Publishing Company, Pacific Grove, California, 1999

[8] Van Huffel S. and Vandewalle J., *The Total Least Squares Problem: Computational Aspects and Analysis*, Philadelphia: SIAM, 1991

Appendix A

Total Least Squares Formula

This appendix presents a simple algebraic closed form solution for the 2-dimensional Total Least Squares problem. As noted in section 3.1.1, this problem arises in connection with line parameter refinement after detection with the Hough/Radon transform. This result is apparently new; solutions given in the literature generally involve singular value matrix decompositions. (See Van Huffel and Vandewalle 1991.)

We are given a list of data points $z_j = (x_j, y_j),\ j = 1 \ldots n$, and we seek an equation for the line of *Total Least Squares* (i.e., the line ℓ that minimizes the sum of squares of "true" distances)

$$J(\ell) = \sum_{j=1}^{n} dist(z_j,\, \ell)^2. \tag{A.1}$$

The *normal* equation for any line ℓ is given by

$$p = x\cos\theta + y\sin\theta, \tag{A.2}$$

where $-\pi/2 \leq \theta\ < \pi/2$ and $-\infty < p < \infty$. Let $\ell^\perp$ denote the oriented line through the origin with direction $v_\theta = (\cos\theta,\, \sin\theta)$ and let $z = (x,\, y)$ denote an arbitrary point in the plane. Then the inner product $z \cdot v_\theta$ is the signed distance from the origin to the orthogonal projection of z on $\ell^\perp$. In particular, $|z \cdot v_\theta - p| = |x\cos\theta + y\sin\theta - p|$ is the distance from $z = (x,\, y)$ to ℓ. (See Figure A.1.)

In terms of the line parameters θ, p, the function (A.1) assumes the form

$$J(\theta,\, p) = \sum_{j=1}^{n} (x_j\cos\theta + y_j\sin\theta - p)^2 \tag{A.3}$$

A closed form for the minimizer of this function is given by the following theorem.

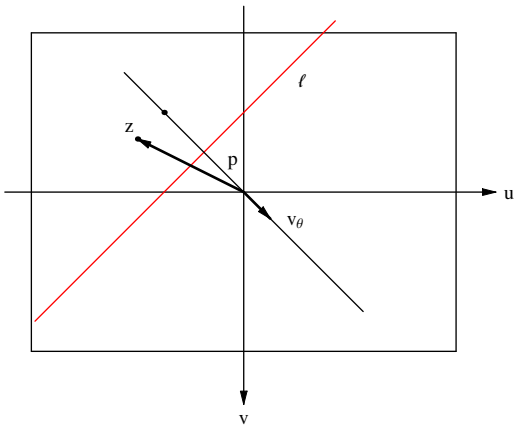

Figure A.1: Distance from z to ℓ is $|z \cdot v_\theta - p|$.

Theorem 3. *Using the notation above, let*

$$(\bar{x}, \bar{y}) = \frac{1}{n} \sum_{j}^{n} (x_j, y_j)$$

be the data centroid, and define

$$A = \frac{1}{2} \sum_{j=1}^{n} ((x_j - \bar{x})^2 - (y_j - \bar{y})^2) \tag{A.4}$$

$$B = \sum_{j=1}^{n} (x_j - \bar{x})(y_j - \bar{y}) \tag{A.5}$$

$$r = \sqrt{A^2 + B^2}. \tag{A.6}$$

If $r > 0$, then (A.1) *has the unique minimizer given by*

$$\cos\theta = \sqrt{\frac{r - A}{2r}} \tag{A.7}$$

$$\sin\theta = \pm\sqrt{\frac{r + A}{2r}} \tag{A.8}$$

$$p = \bar{x}\,\cos\theta + \bar{y}\,\sin\theta \tag{A.9}$$

where the $\pm$ sign in (A.8) *is chosen opposite to the sign of B. If $r = 0$, then* (A.9) *minimizes* (A.1) *for any value of θ.*

Proof. To simplify things, translate the coordinate system so that the origin is at the data centroid. Let $u_j = x_j - \bar{x}$ and $v_j = y_j - \bar{y}$ for $j = 1 \ldots n$. Then equation (A.3)

becomes

$$\begin{aligned} J(\theta,\, p) &= \sum_{j=1}^{n} (u_j \cos\theta + v_j \sin\theta + \bar{x}\cos\theta + \bar{y}\sin\theta - p)^2 \\ &= \sum_{j=1}^{n} (u_j \cos\theta + v_j \sin\theta)^2 + n\,(\bar{x}\cos\theta + \bar{y}\sin\theta - p)^2. \end{aligned} \tag{A.10}$$

(The term linear in p cancels out since $\sum_j u_j = 0 = \sum_j v_j$.) Clearly, for any fixed value of θ, the value $J(\theta,\, p)$ is a minimum when

$$p = p_\theta = \bar{x}\cos\theta + \bar{y}\sin\theta. \tag{A.11}$$

It suffices therefore to find θ that minimizes

$$\begin{aligned} J_0(\theta) &= \sum_{j=1}^{n} (u_j \cos\theta + v_j \sin\theta)^2 \\ &= \sum_{j=1}^{n} u_j^2 \cos^2\theta + \sum_{j=1}^{n} v_j^2 \sin^2\theta + 2\sum_{j=1}^{n} u_j v_j \cos\theta \sin\theta. \end{aligned} \tag{A.12}$$

Using the trigonometric identities

$$\cos^2\theta = \frac{1}{2}(1 + \cos 2\theta)$$

$$\sin^2\theta = \frac{1}{2}(1 - \cos 2\theta)$$

$$\sin 2\theta = 2\sin\theta\cos\theta,$$

we rewrite equation (A.12) in the form

$$J_0(\theta) = A\cos 2\theta + B\sin 2\theta + C, \tag{A.13}$$

where

$$A = \frac{1}{2}\sum_{j=1}^{n}(u_j^2 - v_j^2), \qquad B = \sum_{j=1}^{n} u_j v_j, \qquad C = \frac{1}{2}\sum_{j=1}^{n}(u_j^2 + v_j^2). \tag{A.14}$$

Let $(r,\, \theta)$ be the polar coordinates for $(A,\, B)$

$$(A, B) = r(\cos\phi,\, \sin\phi),, \tag{A.15}$$

where

$$r = \sqrt{A^2 + B^2} \quad \text{and} \quad 0 \le \phi < 2\pi. \tag{A.16}$$

If $r = 0$, then any value for θ minimizes $J_0(\theta) = C$, so assume $r > 0$. By using the double angle formula for cosines, equation (A.13) becomes

$$J_0(\theta) = r\cos(\phi - 2\theta) + C. \tag{A.17}$$

Setting $\theta = \phi/2 - \pi/2$ yields the minimum value $J_0(\theta) = C - r$.

Since ϕ is the unique value in the interval $[0, 2\pi)$ such that (A.15) holds, so θ is the unique value in the interval $[-\pi/2, \pi/2)$ that minimizes J_0. For this value of θ we have

$$\cos\theta = \quad \sin(\phi/2) = \quad \sqrt{\frac{1}{2}(1 - \cos\phi)} = \quad \sqrt{\frac{r - A}{2r}} \tag{A.18}$$

$$\sin\theta = -\cos(\phi/2) = \pm\sqrt{\frac{1}{2}(1 + \cos\phi)} = \pm\sqrt{\frac{r + A}{2r}}. \tag{A.19}$$

The $\pm$ sign in (A.19) must be chosen opposite to the sign of B, since $\cos(\phi/2) < 0$ when $\sin\phi = B/r < 0$. □

The special case $r = 0$ has an interesting interpretation in terms of complex numbers. Let $z_j = u_j + i\,v_j,\ j = 1 \ldots n$. Then

$$A + i\,B = \frac{1}{2}\sum_{j=1}^{n} z_j^2.$$

The condition $r = 0$ is then equivalent to the centroid of the z_j^2's being 0.