

AD-A107 858

TEXAS TECH UNIV LUBBOCK DEPT OF MATHEMATICS  
LIE GROUPS AND LIE ALGEBRAS IN VIDEO TRACKING.(U)  
OCT 81 T G NEWMAN

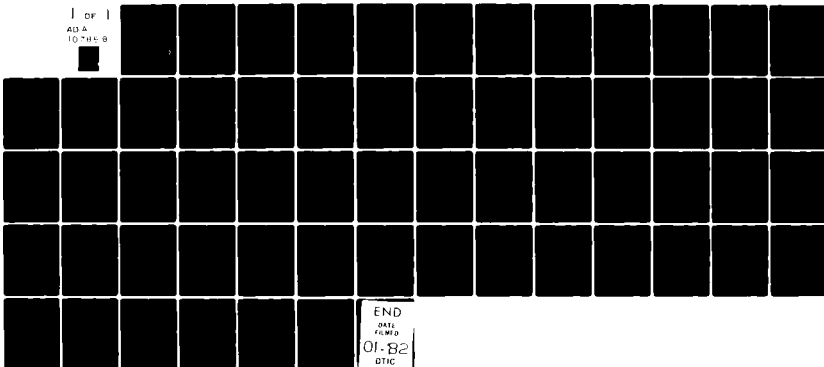
F/G 12/1

UNCLASSIFIED

ARO-17219.1-R-EL

DAAG29-80-C-0087  
NL

1 OF 1  
AD A  
10 \*HC 8



END  
DATE  
FILMED  
01-82  
DTIC

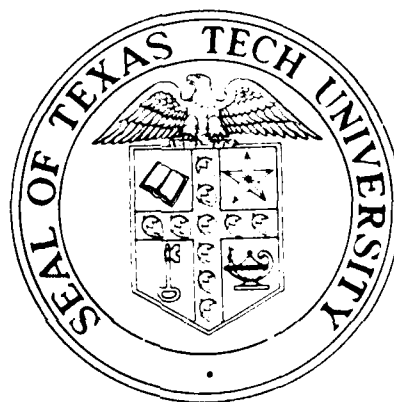
17

ARO 17219.1-R-EL

LEVEL

(12)

AD A1U7858



DTIC  
ELECTE  
NOV 27 1981

A

DTIC FILE COPY

This document has been approved  
for public release and sale; its  
distribution is unlimited.

81 11 24073

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                       | READ INSTRUCTIONS<br>BEFORE COMPLETING FORM                    |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|----------------------------------------------------------------|
| 1. REPORT NUMBER                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER                                  |
| 4. TITLE (and Subtitle)<br>Lie Groups and Lie Algebras in Video Tracking                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                       | 5. TYPE OF REPORT & PERIOD COVERED<br>Final 3/1/80 - 2/28/81   |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                       | 6. PERFORMING ORG. REPORT NUMBER                               |
| 7. AUTHOR(s)<br>Thomas G. Newman                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                       | 8. CONTRACT OR GRANT NUMBER(s)<br>DAAG 29-80-C-0087            |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Department of Mathematics<br>Texas Tech University<br>Lubbock, TX 79409                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                       | 10. PROGRAM ELEMENT, PROJECT, TASK<br>AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>U. S. Army Research Office<br>Post Office Box 12211<br>Research Triangle Park, NC 27709                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                       | 12. REPORT DATE<br>10-26-81                                    |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                       | 13. NUMBER OF PAGES<br>58                                      |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                       | 15. SECURITY CLASS. (of this report)<br>Unclassified           |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                       | 15a. DECLASSIFICATION/DOWNGRADING<br>SCHEDULE                  |
| 16. DISTRIBUTION STATEMENT (of this Report)<br><br>Approved for Public Release;<br>Distribution Unlimited.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                       |                                                                |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)<br><br>NA                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                       |                                                                |
| 18. SUPPLEMENTARY NOTES<br>The view, opinions, and/or findings contained in this report<br>are those of the author(s) and should not be construed as<br>an official department of the Army position, policy, or decision<br>unless so designated by other documentation.                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                       |                                                                |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number)<br><br>Video tracking, two dimensional image, Lie group, Lie algebra, differential<br>form, affine transformations                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                       |                                                                |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number)<br>When a three-dimensional dynamic scene is projected to a two-dimensional image<br>plane, a complex class of motions are observed, which may be approximated by<br>translation and linear transformation. In this report, we show how such motion<br>may be resolved into principal components and measured from 2-dimensional data.<br>Lie theoretic techniques are used to obtain a motion model, and to incorporate<br>generalized velocity measurements in a closed loop feedback tracker. Stability<br>of numerical calculations is enhanced by a method based on an application of<br>Stoke's Theorem to certain differential 2-forms. |                       |                                                                |

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

Lie Groups and Lie Algebras  
in Video Tracking

by

Thomas G. Newman

October 26, 1981

U.S. Army Research Office

Contract DAAG29-80-C-0087

Department of Mathematics

Texas Tech University

Lubbock, Texas 79409

Approved for Public Release;  
Distribution Unlimited.

|                             |   |
|-----------------------------|---|
| Approved for Public Release | ✓ |
| Contract DAAG29-80-C-0087   |   |
| Department of Mathematics   |   |
| Texas Tech University       |   |
| Lubbock, Texas 79409        |   |
| October 26, 1981            |   |
| Thomas G. Newman            |   |
| Author                      |   |
| Editor                      |   |
| Reviewer                    |   |
| Final Approval              |   |
| Signature                   |   |
| Initials                    |   |
| Stamp                       |   |
| Other                       |   |
| A                           |   |

## Table of Contents

|              |        |
|--------------|--------|
| Chapter I    | 1-8    |
| Chapter II   | 9-13   |
| Chapter III  | 14-18  |
| Chapter IV   | 19-20  |
| Bibliography | 21     |
| Appendix A   | A1-A11 |
| Appendix B   | B1-B17 |
| Appendix C   | C1-C6  |

# I. Affine Transformations and Tracking.

When a dynamic three-dimensional scene is observed via an optical projection, a quite complex class of motions are induced in the image plane [5,6,7,9]. In general, this class of motions is highly non-linear, being dependent on the geometry of the objects being observed as well as their trajectories in space[3]. Nevertheless, in many cases the motion is approximated closely by translation, magnification and rotation in the image plane. It is easy to see that this approximation is best for motion in space which consists of translation and rotation about a line parallel to the bare sight.

A better approximation results by consideration of the full affine group in the plane, which includes shearing in two directions as well as the motions mentioned above. By definition, an affine transformation in the plane  $R^2$  is of the form

$$T(y) = Ay + a, y \in R^2 \quad (1.1)$$

where  $A$  is a non-singular  $2 \times 2$  matrix and  $a \in R^2$  is considered as a column vector [4,10]. The set of all such transformations  $T$  is called the general affine group and is denoted  $GA(2)$ . It is easily seen that the subset consisting of translations, magnifications and rotations forms a subgroup, which we denote by  $SA(2)$ . In order that  $T(y) = Ay + a$  belong to  $SA(2)$  it is necessary and sufficient that  $A_{11} = A_{22}$  and  $A_{12} = -A_{21}$ . In this case, the magnification factor is  $(A_{11}^2 + A_{21}^2)^{1/2}$  and the rotation angle is  $\text{atn}(A_{21}/A_{11})$ .

In order to consider dynamic images, it is necessary to allow

$A$  and  $a$  in (1.1) to depend on time. This gives rise to a trajectory  $u(t,y)$  for each  $y \in \mathbb{R}^2$  given by

$$u(t,y) = A(t)y + a(t) \quad (1.2)$$

where  $(A(t), a(t)) \in GA(2)$ , and we require that  $A(0)=I$ ,  $a(0)=0$  in order that the trajectory pass through  $y$  at time  $t=0$ ; i.e.,  $u(0,y) = y$ .

As in [4], though only for linear transformations, we may realize the pair  $(A(t), a(t))$  as the solution of a linear system of differential equations. Let us define

$$\dot{\Lambda}(t) = \dot{A}(t) A^{-1}(t) \quad (1.3a)$$

$$\dot{\lambda}(t) = \dot{a}(t) - \Lambda(t) a(t), \quad (1.3b)$$

from which,

$$\dot{A}(t) = \Lambda(t) A(t), \quad A(0) = I \quad (1.4a)$$

$$\dot{a}(t) = \lambda(t) + \Lambda(t) a(t), \quad a(0) = 0 \quad (1.4b)$$

We may summarize the correspondences defined by (1.3) and (1.4) as follows:

Theorem 1.1: Equations (1.3) and (1.4) establish a one-to-one correspondence between differentiable curves  $(A(t), a(t))$  in  $GA(2)$  satisfying  $A(0) = I$ ,  $a(0) = 0$  and continuous curves  $(\Lambda(t), \lambda(t))$  where  $\Lambda(t)$  is an arbitrary  $2 \times 2$  matrix and  $\lambda(t) \in \mathbb{R}^2$ . Moreover, in order that  $(A,a)$  belong to  $SA(2)$  it is necessary and sufficient that  $\Lambda_{11} = \Lambda_{22}$  and  $\Lambda_{12} = -\Lambda_{21}$ .

The first part of the above theorem apparent from (1.3) and (1.4). A rigorous and detailed proof proceeds exactly as given in [4] for linear transformations. The last part can be deduced by a few calculations using the fact that elements of  $SA(2)$  satisfy  $A_{11} = A_{22}$  and  $A_{12} = -A_{21}$ .



Now, if we differentiate (1.2) with respect to  $t$ , use (1.4) and (1.2) again, we obtain

$$\frac{\partial u}{\partial t}(t, y) = \Lambda(t) u(t, y) + \lambda(t). \quad (1.5)$$

Of course,  $v(t, y) = -\frac{\partial u}{\partial t}(t, y)$  is the velocity field along the trajectory  $u(t, y)$ . Equ. (1.5) shows that the velocity at a point  $u$  depends on  $u$  as well as  $t$  and is therefore not spatially invariant.

Note that (1.5) in fact gives the differential equation for an arbitrary affine trajectory, and when  $\Lambda(t)$  is restricted as in Theorem 1.1, it gives the equation for a trajectory under the restricted group of motions  $SA(2)$ . By virtue of (1.2), we see that  $(\Lambda(t), \lambda(t))$  obtained from (1.4) may be considered as a fundamental system of solutions to the evolution equation (1.5). Now it is important to note that the fundamental system of solutions is completely determined by the pair  $(\Lambda(t), \lambda(t))$  which is spatially invariant, being a function of time only. To establish convenient terminology, let us give the following

Definition 1.1: The pair  $(\Lambda(t), \lambda(t))$  is the generalized velocity field of the family of affine trajectories  $u(t, y)$  defined by (1.5).

We may now state

Theorem 1.2: A family  $u(t, y)$  of affine trajectories satisfying  $u(0, y) = y$  is completely determined from its generalized velocity field, which is spatially invariant, via (1.4) and (1.2). Moreover, the absolute velocity  $v$  at a point  $u$  on a trajectory is given, as in (1.5), by  $v = \Lambda(t)u + \lambda(t)$ .

Let us write  $\lambda(t) = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$ ,  $\Lambda(t) = \begin{bmatrix} \lambda_3 & \lambda_5 \\ \lambda_4 & \lambda_6 \end{bmatrix}$  and expand the equa-

tion  $\dot{u} = \Lambda u + \lambda$  (where the  $t$ -dependence has been suppressed) in the form

$$\frac{\partial}{\partial t} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \lambda_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \lambda_3 \begin{bmatrix} u_1 \\ 0 \end{bmatrix} + \lambda_4 \begin{bmatrix} 0 \\ u_1 \end{bmatrix} + \lambda_5 \begin{bmatrix} u_2 \\ 0 \end{bmatrix} + \lambda_6 \begin{bmatrix} 0 \\ u_2 \end{bmatrix} \quad (1.6)$$

In this way we can identify individual vector fields  $v^1(u), \dots, v^6(u)$  and write (1.6) in the form

$$\frac{\partial u}{\partial t} = \sum_{i=1}^6 \lambda_i(t) v^i(u) \quad (1.7)$$

In a similar manner for  $SA(2)$ , we write

$$\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \text{ and } \Lambda = \begin{bmatrix} \lambda_3 & -\lambda_4 \\ \lambda_4 & \lambda_3 \end{bmatrix}$$

so that

$$\frac{\partial}{\partial t} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \lambda_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \lambda_3 \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \lambda_4 \begin{bmatrix} -u_2 \\ u_1 \end{bmatrix}, \quad (1.8)$$

allowing four vector fields  $v^1(u), \dots, v^4(u)$  to be identified.

Finally, we rewrite (1.8) as

$$\frac{\partial u}{\partial t} = \sum_{i=1}^4 \lambda_i(t) v^i(u) \quad (1.9)$$

It should be noted that the functions  $v^i$  defined by (1.7) or (1.9) are characteristic of the class of motions under consideration, and more general classes of motion can be treated by consideration of other generators  $v^1, v^2, \dots$ . In the cases of

interest, the sets of vector fields derived above define the Lie algebras [2] of GA(2) and SA(2). Any vector field  $v: R^2 \rightarrow R^2$  induces a differential operator  $Y_v$ , called an infinitesimal transformation, which is defined by

$$Y_v = v_1(y) \frac{\partial}{\partial y_1} + v_2(y) \frac{\partial}{\partial y_2} \quad (1.10)$$

where  $v_1(y)$  and  $v_2(y)$  are the components of  $v(y)$ . In Tables 1 and 2 we list the infinitesimal transformations for the groups GA(2) and SA(2), given in terms of a variable  $x=(x_1, x_2)$  for later application

Table 1. Infinitesimal transformations for GA(2).

$$\begin{array}{lll} X_1 = \frac{\partial}{\partial x_1} & X_3 = x_1 \frac{\partial}{\partial x_1} & X_5 = x_2 \frac{\partial}{\partial x_1} \\ X_2 = \frac{\partial}{\partial x_2} & X_4 = x_1 \frac{\partial}{\partial x_2} & X_6 = x_2 \frac{\partial}{\partial x_2} \end{array}$$

Table 2. Infinitesimal transformations for SA(2)

$$\begin{array}{ll} X_1 = \frac{\partial}{\partial x_1} & X_3 = x_1 \frac{\partial}{\partial x_1} + x_2 \frac{\partial}{\partial x_2} \\ X_2 = \frac{\partial}{\partial x_2} & X_4 = x_1 \frac{\partial}{\partial x_2} - x_2 \frac{\partial}{\partial x_1} \end{array}$$

Note that  $u(t, y)$  given by (1.2) may be regarded as the location at time  $t$  of the particle which was at  $y$  at time  $t=0$ . An observer at some point  $x$  will observe this particle provided that  $x = u(t, y) = A(t)y + a(t)$ . We may solve this equation for  $y$  to obtain  $y = A^{-1}(t)(x - a(t))$ . Thus we define the trace of the point  $x \in R^2$  to be

$$s(t,x) = A^{-1}(t)(x-a(t)), \quad x \in \mathbb{R}^2. \quad (1.11)$$

We may interpret  $s(t,x)$  to be the particle which will arrive at  $x$  at time  $t$ .

Let us now consider a two-dimensional image, represented by a function  $f:\mathbb{R}^2 \rightarrow \mathbb{R}$ , and suppose that the image  $f$  is subjected to an affine transformation  $(A(t), a(t))$ . Here a value  $f(y)$  is regarded as a feature which propagates along the trajectories of the motion. This is an extremely powerful, and somewhat restrictive, assumption which is not always valid in real images. For example, it is violated by changes in radiance values which vary as a function of the angle of incident illumination. On the other hand, it is valid in most instances over short time intervals and deviations from this assumption may frequently be treated as higher order effects.

In any event, if the feature  $f(y)$  is propagated along trajectories, then a stationary observer, say at point  $x$ , will observe a value  $F(t,x) = f(s(t,x))$  at time  $t$ , since  $s(t,x)$  represents the particle arriving at  $x$  at time  $t$ . We may now state a most important result.

Theorem 1.3: Let a time-varying image  $F$  be given by

$$F(t,x) = f(s(t,x)) \quad (1.12)$$

where  $s(t,x)$  is an affine trace as in (1.11) with generalized velocities

$$\lambda(t) = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}, \Lambda(t) = \begin{bmatrix} \lambda_3 & \lambda_5 \\ \lambda_4 & \lambda_6 \end{bmatrix}. \quad \text{Then}$$

$$-\frac{\partial F}{\partial t} = \sum_{i=1}^6 \lambda_i(t) X_i F, \quad (1.13)$$

where  $X_1, \dots, X_6$  are given in Table 1.

A similar result holds if we restrict to  $SA(2)$ , using  $X_1, \dots, X_4$  from Table 2.

Proof: This result may be deduced from results given in [7], provided we compensate for the change from "left invariance" in that development to the "right invariance" of the current treatment. However, a direct proof is instructional and will be outlined herein. We first show

Lemma 1.3.1: For  $s(t, x)$  given by (1.11)

$$-\frac{\partial s}{\partial t} = \sum_{i=1}^6 \lambda_i(t) X_i s. \quad (1.14)$$

Proof: First note that by direct calculation we have  $\sum \lambda_i X_i s =$

$$\sum \lambda_i X_i A^{-1}(x-a) = A^{-1}(\sum \lambda_i X_i x) = A^{-1}(\sum \lambda_i v^i(x)) = A^{-1}(\Lambda x + \lambda). \quad \text{Also,}$$

$$\text{noting that } -\frac{d}{dt} A^{-1} = -A^{-1} \dot{A} A^{-1}, \text{ we have } \frac{\partial s}{\partial t} = \frac{\partial}{\partial t} A^{-1}(x-a) =$$

$$-A^{-1} \dot{A} A^{-1}(x-a) - A^{-1} \dot{a} = -A^{-1} \Lambda(x-a) - A^{-1}(\lambda + \Lambda a) = -A^{-1}(\Lambda x + \lambda).$$

Hence, the desired result follows.

Returning to the proof of Theorem 1.3, we have

$$\begin{aligned} \sum_i \lambda_i X_i F(tx) &= \sum_i \lambda_i(t) v_j^i(x) \frac{\partial}{\partial x_j} f(s(t, x)) \\ &= \sum_i \lambda_i(t) v_j^i(x) \frac{\partial s_k}{\partial x_j}(t, x) \frac{\partial f}{\partial s_R}(s) \end{aligned}$$

$$\begin{aligned}
&= \sum_i \lambda_i(t) X_i s_R(t,x) \frac{\partial f}{\partial s_k}(s) \\
&= - \frac{\partial s_k}{\partial t}(t,x) \frac{\partial f}{\partial s_k}(s) = - \frac{\partial f(s(t,x))}{\partial t} \\
&= - \frac{\partial F}{\partial t}(t,x) , \text{ as desired.}
\end{aligned}$$

Theorem 1.3 appears to be fundamental to the analysis of motion in dynamic images. As is evident from the proof, an analogue is valid in a much more general setting. In fact, scrutiny of the proof shows that it depends mainly on Lemma 1.3.1. Consequently, the theorem will hold for any class of motions for which a suitable form of the "trace" lemma can be obtained. The significance of Theorem 1.3 lies chiefly in the fact that the generalized velocities (which are usually unknown) appear as linear coefficients in (1.13), along with quantities which can be calculated from the data  $F(t,x)$ .

The main problem with the extraction of the generalized velocities from (1.13) is the general lack of numerical precision in the calculation of the derivatives from real data (e.g., digitized video). In subsequent sections we shall show how to incorporate (1.13) in a feedback loop which is very stable and how to obtain an equivalent formulation based on integration rather than differentiation.

## II. A Velocity Feedback Tracker

The theoretical results of this section result from research done under a separate contract\* and for which a publication is in preparation. In view of the fact that the techniques have been incorporated in the experimental portion of this report, these results will be presented in this report in the context of affine transformations.

Let absolute image coordinates be denoted by  $y = (y_1, y_2)$  and introduce additional coordinates as follow: Let coordinates  $z$  be established relative to a moving target, and let coordinates  $x$  be established in a movable "window". We assume that the motions of both the target and the window may be described by affine transformations relative to absolute image coordinates. It is assumed that the motion of the window may be chosen at will, while the motion of the target is prescribed (e.g. by nature) and is unknown.

By the affine assumption, we may describe the transformation from window coordinates to image coordinates by

$$y_w(t, x) = A(t)x + a(t) \quad , \quad x \in R_w^2 \quad , \quad (2.1)$$

where  $(A(t), a(t))$  is a suitable family of affine transformations. Similarly, the transformation from target coordinates to image coordinates is

$$y_T(t, z) = B(t)z + b(t) \quad , \quad z \in R_T^2 \quad (2.2)$$

for suitable  $(B(t), b(t)) \in GA(2)$ . Let us denote the respective generalized velocity fields by  $(\lambda_A, \lambda_A)$  and  $(\lambda_B, \lambda_B)$ .

By equating  $y_T(t, z) = y_w(t, x)$  we may solve for the point  $z$

---

\* ONR Contract N0014-76-C-1136

on the target which arrives at point  $x$  in the window at time  $t$ , to obtain:

$$z(t, x) = B^{-1}(Ax + a - b), \quad (2.3)$$

where dependence on  $t$  has been suppressed on the right. We note that  $z(t, x)$  in (2.3) may be regarded as a trace in the sense of the previous section. By an application of Lemma 2.3.1 we have:

Theorem 2.1: There exists a generalized velocity field  $(\Gamma(t), \gamma(t))$  such that

$$-\frac{\partial z(t, x)}{\partial t} = \sum_{i=1}^6 \gamma_i(t) X_i z(t, x), \quad (2.4)$$

where  $\gamma(t) = \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix}$ ,  $\Gamma = \begin{bmatrix} \gamma_3 & \gamma_5 \\ \gamma_4 & \gamma_6 \end{bmatrix}$  and the operators  $X_1, \dots, X_6$  are given in window coordinates as in Table 1.

Now, let  $f(z)$  be a feature of the target, measured at point  $z$ , and assume that this feature propagates along the target trajectories. An observer at point  $x$  in the window therefore observes data  $F(t, x) = f(z(t, x))$ , inasmuch as  $z(t, x)$  is the point which arrives at  $x$  at time  $t$ . From Theorem 1.3 we obtain

Theorem 2.2: In the above context,

$$-\frac{\partial F}{\partial t}(t, x) = \sum_{i=1}^6 \gamma_i(t) X_i F(t, x) \quad (2.5)$$

In principle, (2.5) allows the determination of the generalized velocities of the target relative to the window. Since we have free choice of window velocities, relative to the image coordinates, this is tantamount to measurement of absolute target



velocities. The conversion process will now be described.

Let us denote by  $\bar{X}$  the window space,  $\bar{Y}$  the image space, and let  $T(\bar{X})$  and  $T(\bar{Y})$  be the respective tangent spaces. Since the map from  $\bar{X}$  to  $\bar{Y}$  is  $y = A x + a$ , as in (2.1), it follows that the induced map on the tangent spaces is simply  $y^* = A x^*$  [1,2].

Now the velocity field  $(\Gamma, \gamma)$  defines a map  $\bar{X} \rightarrow T(\bar{X})$  given by  $x^* = \Gamma x + \gamma$  (see (1.5)). Accordingly, a velocity field  $(\nabla, \lambda)$  is induced on  $\bar{Y}$ , which maps  $\bar{Y} \rightarrow T(\bar{Y})$  in a similar fashion. The velocity field  $(\Lambda, \lambda)$  is defined by the commutative diagram

$$\begin{array}{ccc} \bar{X} & \xrightarrow{(\Gamma, \gamma)} & T(\bar{X}) \\ (A, a) \downarrow & & \downarrow A \\ \bar{Y} & \xrightarrow{(\Lambda, \lambda)} & T(\bar{Y}). \end{array}$$

Thus, we calculate  $y^* = \Lambda y + \lambda$ , by inverting  $(A, a)$  and taking the upper path, to be given by  $y^* = A(\Gamma A^{-1}(y-a) + \gamma) = A\Gamma A^{-1}y + A\gamma - A\Gamma A^{-1}a$ . By comparison, we obtain

$$\Lambda = A\Gamma A^{-1} \quad (2.6a)$$

$$\lambda = A\gamma - A\Gamma A^{-1}a \quad (2.6b)$$

Now, since the velocity field  $(\Gamma, \gamma)$  represents the difference between target and window velocities in the window coordinate system, we see that  $(\Lambda, \lambda)$  must represent this same difference relative to the absolute image coordinate system. That is,

$$\Lambda = \Lambda_B - \Lambda_A \quad (2.7a)$$

$$\lambda = \lambda_B - \lambda_A \quad (2.7b)$$

We may summarize these results in a useable form as follows:

Theorem 2.3: Let (2.1) and (2.2) define the motion of a window and a target, respectively, relative to a system of absolute image coordinates, and let  $(\Lambda_A, \lambda_A)$  and  $(\Lambda_B, \lambda_B)$  be the corresponding generalized velocities. Further, let  $(\Gamma, \gamma)$  be the generalized velocities of the target relative to the window, as determined by (2.5).

Then

$$\Lambda_B - \Lambda_A = A\Gamma A^{-1} \quad (2.8a)$$

$$\lambda_B - \lambda_A = A\gamma - A\Gamma A^{-1}a. \quad (2.8b)$$

The previous theorem immediately suggests an algorithm for determination of velocities in an image. More generally, the algorithm performs tracking since, as will be seen, the result is to force the window to follow the target by emulation of velocities.

The algorithm is as follows:

Step 1. Initialize the window by choice of  $A(0)$ ,  $a(0)$ . In the absence of a priori information, initialize  $\Lambda_A(0)=0$ ,  $\lambda_A(0)=0$ .

Sample window values  $F(t_0, x)$  at time  $t_0=0$ .

Step 2. Sample window values  $F(t_n, x)$  at time  $t_n=t_{n-1}+\delta$ . Approximate  $\frac{\partial F}{\partial t}$  and  $X_i F$  at various points in the window and form a system of linear equations using (2.5).

Step 3. Solve the resulting linear equations for  $\gamma_1, \gamma_2, \dots$ .

Step 4. Replace  $\Lambda_A + \Lambda_A + A\Gamma A^{-1}$  and  $\lambda_A + \lambda_A + A\gamma - A\Gamma A^{-1}a$ .

Note: If the calculation of  $\gamma_1, \gamma_2, \dots$  were exact, this would result in  $\Lambda_A + \Lambda_B(t_n)$  and  $\lambda_A + \lambda_B(t_n)$ .

Step 5. Take a  $\delta$  time step in the numerical solution  $\dot{A} = \Lambda_A A$ ,  $\dot{a} = \gamma_A + \lambda_A a$  to obtain  $(A(t_n), a(t_n))$ . This effectively moves

the window.

Step 6. Repeat from step 2.

Emperical results indicate that the above algorithm conveys rapidly over a fairly broad range of target velocities. Although the initial estimate of target velocities is usually fairly coarse, it is generally in the right direction and results in good estimates after 3 to 5 iterations. Subsequently, the target is tracked very well with only a nominal amount of slew. More importantly, the computational speed is such that it is feasible for real-time implementation, with calculations having been done at 25 to 100 iterations per second on various computers, including time spent in simulation support.

The most notable failure is a high degree of instability encountered in dealing with real data in the form of digitized images. The available image data, however, did not have a suitable dynamic range in comparison to the noise level. Considerable improvement resulted by expanding the contract and filtering to obtain a greater dynamic range.

The results of performing the above algorithm on simulated data is presented in appendix A.

### III. Alternate Formulation via 2 - forms.

The major source of error in the calculation of generalized velocities would appear to be that introduced in the numerical approximation of spatial derivatives. Although the situation is improved somewhat by filtering and the use of multi-point formulas, it is still desirable to seek alternate approaches. As can be seen from examination of the algorithm of the preceeding section, any method for calculation of generalized velocities may easily be inserted in the basic tracker.

In this section we appeal to a form of Stoke's Theorem [1] to obtain an integration based analogue of Theorem 1.3. The formula obtained is strictly valid only when the generalized velocities are constant, although it is a useful approximation when the rates of change of the velocities are small.

We consider the three dimensional space  $R^3$  consisting of time  $t$  and two spatial variables  $x$  and  $y$ . Coordinates  $\xi=(t,x,y)$  are chosen to make a right-hand coordinate system, and we observe this orientation in defining differential forms. We state the form of Stoke's Theorem required:

Stoke's Theorem: Let  $\Omega$  be a rectangle in  $(t,x,y)$  space  $R^3$  and let  $w$  be a differentiable 2-form. Then

$$\int_{\partial\Omega} w = \int_{\Omega} dw \quad (3.1)$$

Here  $w$  is of the form  $w = \alpha_0 dx dy + \alpha_1 dy dt + \alpha_2 dt dx$ , with  $\alpha_0, \alpha_1, \alpha_2$ , differentiable functions of  $\xi \in R^3$ , and

$$dw = \frac{\partial \alpha_0}{\partial t} + \frac{\partial \alpha_1}{\partial x} + \frac{\partial \alpha_2}{\partial y} \quad dt dx dy.$$

Although the results to be presented may be generalized considerably, our derivation and experimental results will be given only for SA(2). Thus, the appropriate vector fields and corresponding infinitesimal transformations may be obtained from (1.8), (1.10) and Table 2, with  $x$  and  $y$  substituted in the obvious manner. By analogy with (1.9), for a constant velocity field

$\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ , let us define a vector valued map

$$\eta: R^4 \times R^2 \rightarrow R^2 \text{ by}$$

$$\eta(\lambda, \xi) = \sum_{i=1}^4 \lambda_i v^i(\xi) = \begin{bmatrix} \lambda_1 + \lambda_3 x - \lambda_4 y \\ \lambda_2 + \lambda_4 x + \lambda_3 y \end{bmatrix}. \quad (3.2)$$

As usual, let  $\eta_1, \eta_2$  denote the components of  $\eta$ .

Now, if  $s(t, \xi)$  is the trace corresponding to the generalized velocity field  $\lambda$  and  $F(t, \xi) = f(s(t, \xi))$  is observed data, we may express Equ. (1.13) of Theorem 1.3 as

$$-\frac{\partial F}{\partial t} = \eta_1 \frac{\partial F}{\partial x} + \eta_2 \frac{\partial F}{\partial y}. \quad (3.3)$$

We intend to apply Stoke's Theorem to the 2-form defined by

$$\omega = F dx dy + \eta_1 F dy dt + \eta_2 F dt dx \quad (3.4)$$

The principal result is stated as

Theorem 3.1: In the context above,

$$\begin{aligned} d\omega &= \left( \frac{\partial \eta_1}{\partial x} + \frac{\partial \eta_2}{\partial y} \right) F dt dx dy \\ &= 2\lambda_3 F dt dx dy \end{aligned} \quad (3.5)$$

To establish this, we calculate  $d\omega$ , using the fact that  $dx dy dt = dy dt dx = dt dy dx$  (whereas, for example, observing orientation,  $dt dy dt = -dt dx dy$ ). We have

$$d\omega = \left( \frac{\partial F}{\partial t} + \pi_1 \frac{\partial F}{\partial x} + F \frac{\partial \pi_1}{\partial x} + \pi_2 \frac{\partial F}{\partial y} + F \frac{\partial \pi_2}{\partial y} \right) dt dx dy. \quad \text{By}$$

application of (3.3) and then (3.2) this simplifies to  $d\omega =$

$$\left( F \frac{\partial \pi_1}{\partial x} + F \frac{\partial \pi_2}{\partial y} \right) dt dx dy = 2\lambda_3 F dt dx dy, \text{ as desired.}$$

By an application of Stoke's Theorem, and a somewhat tedious calculation, we immediately obtain:

Theorem (3.2). Let  $\Omega$  be a rectangle in  $R^3$  defined by opposing corners  $(t_1, x_1, y_2)$ . In the context described above, in particular with  $\lambda$  constant, we have

$$\sum_{i=1}^4 \lambda_i k_i = k_0 \quad (3.6)$$

where  $k_0, k_1, \dots, k_4$  are given in Table 3.

Table 3. Coefficients resulting from Stoke's Theorem.

$$\begin{aligned} k_0 &= - \int_{\partial \Omega} F dx dy \\ k_1 &= \int_{\partial \Omega} F dy dt \\ k_2 &= \int_{\partial \Omega} F dt dx \\ k_3 &= \int_{\partial \Omega} xF dy dt + \int_{\partial \Omega} yF dt dx - 2 \int_{\Omega} F dt dx dy \\ k_4 &= \int_{\partial \Omega} xF dt dx - \int_{\partial \Omega} yF dy dt \end{aligned}$$

It is important that orientation be considered in the evaluation of the coefficients in Table 3 (see [1]). The sign convention is such that for a principal 2-form (e.g.,  $dx dy$ ) a positive (negative) sign prevails on a face of the rectangle  $\Omega$  provided

that application of a right-hand rule points outward from (inward to ) the rectangle  $\Omega$ . Writing  $\int_x$  for  $\int_{x_1}^{x_2}$  (similarly for  $t$  and  $y$ ) and assuming that  $t_1 < t_2$ ,  $x_1 < x_2$ ,  $y_1 < y_2$ , by way of example we have,

$$\int_{\partial\Omega} F \, dx dy = \int_y \int_x F(t_2, x, y) \, dx dy - \int_y \int_x F(t_1, x, y) \, dx dy,$$

and

$$\int_{\partial\Omega} xF \, dy dt = x_2 \int_t \int_y F(t, x_2, y) \, dy dt - x_1 \int_t \int_y F(t, x_1, y) \, dy dt$$

and

$$\int_{\partial\Omega} xF dt dx = \int_x \int_t xF(t, x, y_2) \, dt dx - \int_x \int_t xF(t, x, y_1) \, dt dx.$$

The remaining integrals may be expanded in a similar fashion.

Observe that differences are not entirely eliminated from the final formulas. However, the formulas are so written to indicate that the differences are taken after integration, even though in certain cases the formula could be collapsed with a difference taken before evaluation of the iterated integral.

The advantage of (3.6) over (1.13) as a means of calculation of the generalized velocities is achieved mainly by the filtering effect of the surface and volume integrals. As a matter of practice, several rectangles  $\Omega_1, \dots, \Omega_m$  are selected. Each rectangle  $\Omega_e$  gives rise to an equation of the form (3.6),

$$\sum_{i=1}^4 k_i^{(e)} \lambda_i = k_0^{(e)} \quad (3.7)$$

The resulting system of  $m$  equations in 4 unknowns may then be solved by a least-squares method. Note that this approach may be applied to the feedback tracking algorithm presented in Section 2

to calculate the velocities  $v_1, \dots, v_4$  of the target relative to the window, replacing the corresponding calculations based on (2.5). This has been implemented in a computer program and tested on real image data. The results are very encouraging and are presented in part in Appendix B. This method involves more computational overhead, with the best rate achieved to this point being about 10 iterations (=frames) per second. With some streamlining we believe that real-time rates of 30 frames per second can be achieved.



#### IV. Summary and Conclusions.

This report presents a method based on the theory of Lie groups for velocity tracking in a dynamic image in which the motion of picture parts can be ascribed to affine transformations. A feedback tracking algorithm was developed and tested on simulated data.

Since the random disturbances in real images preclude the use of simple methods for obtaining equations involving the velocities of trajectory, a method based on integration of differential forms was developed. This method was incorporated in the feedback tracker and tested on real image data. The results are very encouraging, with computation speeds approaching ten frames per second on a VAX 11/780. We believe that this method is viable as a component of a real-time video tracking system.

Among the problems left outstanding, a satisfactory algorithm for target acquisition has not yet been developed. In the experimental work performed, the initial target location was supplied as an input parameter. To be useful, a method for automating this step is essential.

In addition, we continue to experience problems with numerical precision. This seems to be related to the absence of sufficient dynamic range in real image data, indicating that this could be improved by changes in the capability of sensors. We feel that a great improvement would result from greater contrast in the image data.

Finally, the class of motions considered herein (affine or restricted affine) is not general enough for many applications,

and the methods need to be extended to include projective distortion as well.

The equations which relate generalized velocities to time-varying images have other applications. They have been applied to a problem in pattern matching with considerable success, as fully described in [1]. The theoretical results and a summary of the experimental results of [11] have been submitted for publication as reference [8], which is attached as Appendix C.

### References

- [1] Buck, C.R. and E.F. Buck, Advanced Calculus, McGraw Hill Book Co., New York, 1965.
- [2] Cohn, P.M., Lie Groups, Cambridge University Press, London, 1957.
- [3] Fishback, W.T., Projective and Euclidean Geometry, John Wiley and Sons, Inc., 1969.
- [4] Greub, W.H., Linear Algebra, Springer-Verlag, Inc., New York, 1967.
- [5] Huang, T.S., "Noise filtering in moving images", Workshop on imaging trackers and autonomous acquisition applications for missile guidance, Redstone Arsenal, Huntsville, AL, 1979.
- [6] Martin, W., and J.K. Aggarwal, "Survey, dynamic scene analysis", Computer Graphics and Image Processing, Vol. 7, 1978.
- [7] Newman, T.G., and D.A. Demus, "Lie theoretic methods in video tracking", Workshop on imaging trackers and autonomous acquisition applications for missile guidance", Redstone Arsenal, Huntsville, AL, 1979
- [8] Newman, T.G. and L. Zlobec, "Adaptive pattern matching using control theory on Lie groups", Proceedings of the International Symposium on the Mathematical Theory of Networks and Systems, Santa Monica, CA, Aug. 1981.
- [9] Roach, J.W., and J.K. Aggarwal, "Computer tracking of objects moving in space", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 1, 1979.
- [10] Smirnov, V.I. (rev., ed. R.A. Silverman), Linear Algebra and Group Theory, McGraw-Hill Book Co., Inc., New York, 1961.
- [11] Zlobec, Leopold, Pattern Matching by Means of Adaptive Control, Master's Report, Texas Tech University, 1980.

## APPENDIX A

### Feedback Tracker Simulation

A typical imaging system might include a sensor with a diameter (or cross section) of 25 mm and an optical focal length of 200 mm. At a 500 x 500 pixel density we obtain a conversion factor of .05 mm/pix or 20 pix/mm. With a target range of 1 km, say, then we obtain a conversion rate from sensor to target of 5 m/mm @ 1 km.

In the results to be presented, translation velocities may be regarded as being given in mm/sec. Conversion to pix/sec or m/sec at the target may be done by multiplication by the appropriate factor. Thus, a translation velocity of 7 mm/sec at the sensor corresponds to 20 pix/sec or to 5 m/sec at a target having a range of 1 km. A magnification velocity of 1/sec translates by the same factor and would therefore represent a velocity of 5m/sec @ 1 km toward the sensor. On the other hand, rotational velocity may be considered as given in radius/sec.

In Table A-1 we present the output of the tracking simulator (the output routines were modified slightly for ease of presentation). Note that the target velocities (10, -10, 10, 10) correspond to a 3-D object with a translational velocity of 86.6 m/sec @ 1 km (about 194 miles/hour) which is rotating about 3 revolutions per second about bore sight. The time base was chosen as 100 frames/sec. Inspection of the last four columns of Table A-1 shows that the target velocities have been acquired satisfactorily after only 3 frames, at  $t=.03$ , and subsequently refined to exact values.

| +----- Actual Target Velocities -----+ |       |             |               |
|----------------------------------------|-------|-------------|---------------|
| Hor                                    | Trans | Vert Trans  | Magnification |
| 0.1000E+02                             |       | -0.1000E+02 | 0.1000E+02    |

| Time | Horizontal<br>Position | Vertical<br>Position | +----- Generalized Velocities -----+ |       |             |               |            |
|------|------------------------|----------------------|--------------------------------------|-------|-------------|---------------|------------|
|      |                        |                      | Hor                                  | Trans | Vert Trans  | Magnification | Rotation   |
| 0.00 | 0.0000E+00             | 0.0000E+00           | 0.0000E+00                           |       | 0.0000E+00  | 0.0000E+00    | 0.0000E+00 |
| 0.01 | 0.8356E-01             | -0.1050E+00          | 0.8356E+01                           |       | -0.1050E+02 | 0.7970E+01    | 0.5660E+01 |
| 0.02 | 0.1967E+00             | -0.2069E+00          | 0.8462E+01                           |       | -0.9949E+01 | 0.1001E+02    | 0.9678E+01 |
| 0.03 | 0.3367E+00             | -0.3083E+00          | 0.9969E+01                           |       | -0.1002E+02 | 0.1003E+02    | 0.9967E+01 |
| 0.04 | 0.5012E+00             | -0.4055E+00          | 0.9999E+01                           |       | -0.1001E+02 | 0.1001E+02    | 0.9999E+01 |
| 0.05 | 0.6919E+00             | -0.4960E+00          | 0.1000E+02                           |       | -0.1000E+02 | 0.1000E+02    | 0.1000E+02 |
| 0.06 | 0.9107E+00             | -0.5764E+00          | 0.1000E+02                           |       | -0.1000E+02 | 0.1000E+02    | 0.1000E+02 |
| 0.07 | 0.1159E+01             | -0.6429E+00          | 0.1000E+02                           |       | -0.1000E+02 | 0.1000E+02    | 0.1000E+02 |
| 0.08 | 0.1440E+01             | -0.6913E+00          | 0.1000E+02                           |       | -0.1000E+02 | 0.1000E+02    | 0.1000E+02 |
| 0.09 | 0.1753E+01             | -0.7165E+00          | 0.1000E+02                           |       | -0.1000E+02 | 0.1000E+02    | 0.1000E+02 |
| 0.10 | 0.2100E+01             | -0.7128E+00          | 0.1000E+02                           |       | -0.1000E+02 | 0.1000E+02    | 0.1000E+02 |

A3

The tracking simulation program, whose listing follows, is capable of a real-time rate of 33 frames/sec on a VAX 11/780, including the time spent simulating target motion.

```

100 C*****
200 C
300 C Program Title: TRACK4.FOR
400 C
500 C Function: Demonstrate feedback tracker using synthetic
600 C data. Provide simulated motion consisting of
700 C translation, magnification and rotation.
800 C
900 C Program Author: Thomas G. Newman
1000 C Department of Mathematics
1100 C Texas Tech University
1200 C Lubbock, Texas 79409
1300 C
1400 C Notice: Permission is herewith granted for use of these
1500 C programs, in whole or in part, for other than
1600 C personal or corporate gain.
1700 C
1800 C*****
1900 C
2000 COMMON /WINDOW/ IWSIZE,WIND(5,5),SWIND(5,5),COORD(2,5,5)
2100 COMMON /EQU/ NEQU,NUNK,A(9,9),XLAMDA(9),B(9)
2200 COMMON /PARMS/ TIME,XSTEP,TSTEP,FEED(6)
2300 COMMON /COCHGS/ XTRANI,YTRANI,ECOSI,ESINI,
2400 1 XTRANW,YTRANW,ECOSW,ESINW,
2500 2 XVELI,YVELI,VMAGI,VROTI,
2600 3 XVELW,YVELW,VMAGW,VROTW
2700 100 TYPE *, 'ENTER VELOCITIES AND PRESS RETURN'
2800 READ (5,*,END=200) XVELI,YVELI,VMAGI,VROTI
2900 CALL INIT
3000 DO 175 I=1,15
3100 CALL SAMPLE
3200 CALL DERIV
3300 CALL LINEQ
3400 CALL UPDATE
3500 CALL MOVER
3600 CALL COMPAR
3700 175 CONTINUE
3800 GO TO 100
3900 200 STOP
4000 END
4100 C
4200 C
4300 C
4400 C This subroutine performs various initialization functions.
4500 C
4600 C
4700 SUBROUTINE INIT
4800 COMMON /WINDOW/ IWSIZE,WIND(5,5),SWIND(5,5),COORD(2,5,5)
4900 COMMON /EQU/ NEQU,NUNK,A(9,9),XLAMDA(9),B(9)
5000 COMMON /PARMS/ TIME,XSTEP,TSTEP,FEED(6)
5100 COMMON /COCHGS/ XTRANI,YTRANI,ECOSI,ESINI,
5200 1 XTRANW,YTRANW,ECOSW,ESINW,
5300 2 XVELI,YVELI,VMAGI,VROTI,
5400 3 XVELW,YVELW,VMAGW,VROTW

```

```

5500 C
5600 C      Data is to be sampled at each point of a
5700 C      window of size IWSIZE by IWSIZE. A linear equation
5800 C      is to be formed at each 'interior' point, using the
5900 C      boundary points only in the differentiation process.
6000 C
6100 C      The number of equations, NEQU, is therefore the number
6200 C      of interior points, and involve NUNK unknowns, = 4
6300 C      in this program, but changeable to 6 for GA(2).
6400 C
6500 C      XSTEP and TSTEP are logical steps in space and time.
6600 C
6700 C      IWSIZE = 5
6800 C      NEQU = (IWSIZE - 2)**2
6900 C      NUNK = 4
7000 C      XSTEP = 0.05
7100 C      TSTEP = 0.01
7200 C
7300 C
7400 C      GENERATE THE COORDINATES OF THE SAMPLE GRID RELATIVE TO
7500 C      THE WINDOW.
7600 C
7700 C      MID = IWSIZE/2 + 1
7800 C      DO 10 I=1,IWSIZE
7900 C          DO 15 J=1,IWSIZE
8000 C
8100 C          WINDOW ORIGIN AT CENTER OF SAMPLE GRID
8200 C
8300 C          COORD(1,I,J)=XSTEP*FLOAT(I - MID)
8400 C          COORD(2,I,J)=XSTEP*FLOAT(J - MID)
8500 C
8600 C          WINDOW ORIGIN AT CORNER OF SAMPLE GRID
8700 C
8800 C          COORD(1,I,J)=XSTEP*FLOAT(I)
8900 C          COORD(2,I,J)=XSTEP*FLOAT(J)
9000 C
9100 C      15      CONTINUE
9200 C      10      CONTINUE
9300 C
9400 C      SET FEEDBACKS TO UNITY. LOWER VALUES PRODUCE
9500 C      SLOWER ACQUISITION BUT GREATER STABILITY. LARGER VALUES
9600 C      MAY SPEED ACQUISITION BUT CAUSE INSTABILITY.
9700 C
9800 C      FEED(1) = 1.0
9900 C      FEED(2) = 1.0
10000 C      FEED(3) = 1.0
10100 C      FEED(4) = 1.0
10200 C      FEED(5) = 1.0
10300 C      FEED(6) = 1.0
10400 C
10500 C      INITIALIZE THE TRAJECTORY OF THE IMAGE AT BORE SIGHT
10600 C
10700 C      XTRANI = 0.0
10800 C      YTRANI = 0.0

```



```

10900      ECOSI = 1.0
11000      ESINI = 0.0
11100      C
11200      C      INITIALIZE THE POSITION OF THE WINDOW AT BORE SIGHT
11300      C
11400      XTRANW = 0.0
11500      YTRANW = 0.0
11600      ECOSW = 1.0
11700      ESINW = 0.0
11800      C
11900      C      INITIALIZE THE WINDOW VELOCITY TO XERO
12000      C
12100      XVELW = 0.0
12200      YVELW = 0.0
12300      VMAGW = 0.0
12400      VROTW = 0.0
12500      C
12600      C      GET AN INITIAL SAMPLE FROM WINDOW
12700      C
12800      TIME = 0.0
12900      CALL SAMPLE
13000      C
13100      C      TAKE THE INITIAL STEP IN TIME
13200      C
13300      CALL MOVER
13400      C
13500      C      PRINT PAGE HEADINGS
13600      C
13700      WRITE (6,1000)
13800      WRITE (6,1010)
13900      1000  FORMAT('1','TIME',5X,'XTRANI',8X,'YTRANI',8X,'ECOSI',9X,
14000      1      'ESINI',9X,'XVELI',9X,'YVELI',9X,'VMAGI',9X,'VROTI')
14100      1010  FORMAT(15X,'XTRANW',8X,'YTRANW',8X,'ECOSW',9X,
14200      1      'ESINW',9X,'XVELW',9X,'YVELW',9X,'VMAGW',9X,'VROTW')
14300      C
14400      C      PRINT THE INITIAL COMPARISON BETWEEN TRAJECTORIES
14500      C
14600      CALL COMPAR
14700      RETURN
14800      END
14900      C
15000      C      >>>>>>>> SUBROUTINE SAMPLE <<<<<<<<<
15100      C
15200      C      This subroutine generates values in a rectangular
15300      C      grid in the tracking window, saving the old values.
15400      C
15500      SUBROUTINE SAMPLE
15600      COMMON /WINDOW/ IWSIZE,WIND(5,5),SWIND(5,5),COORD(2,5,5)
15700      DO 20 I=1,IWSIZE
15800      DO 25 J=1,IWSIZE
15900      X = COORD(1,I,J)
16000      Y = COORD(2,I,J)
16100      SWIND(I,J) = WIND(I,J)
16200      WIND(I,J) = FWIND(X,Y)

```

```

16300 25      CONTINUE
16400 20      CONTINUE
16500      RETURN
16600      END
16700 C
16800 C      >>>>>>>>> SUBROUTINE DERIV <<<<<<<<<<
16900 C
17000 C      This routine calculates the various derivatives needed
17100 C      for formation of the linear system for the generalized
17200 C      velocities.
17300 C
17400      SUBROUTINE DERIV
17500      COMMON /WINDOW/ IWSIZE,WIND(5,5),SWIND(5,5),COORD(2,5,5)
17600      COMMON /EQU/    NEQU,NUNK,A(9,9),XLAMDA(9),B(9)
17700      COMMON /PARMS/  TIME,XYPEST,TSTEP,FEED(6)
17800      SCALAR = 2.0 * XYPEST/TSTEP
17900      K = IWSIZE - 2
18000      DO 20 I=1,K
18100          II = I + 1
18200          DO 10 J=1,K
18300              JJ = J + 1
18400              L = (I-1)*K + J
18500              X = COORD(1,II,JJ)
18600              Y = COORD(2,II,JJ)
18700              DX=WIND(II+1,JJ) - WIND(II-1,JJ)
18800              DY=WIND(II,JJ+1) - WIND(II,JJ-1)
18900              A(L,1)= DX
19000              A(L,2)= DY
19100              A(L,3)= X*DX + Y*DY
19200              A(L,4)= X*DY - Y*DX
19300              B(L)= SCALAR * (WIND(II,JJ) - SWIND(II,JJ))
19400 10      CONTINUE
19500 20      CONTINUE
19600      RETURN
19700      END
19800 C
19900 C      >>>>>>>>> SUBROUTINE LINEQ <<<<<<<<<<
20000 C
20100 C      Modified from the argument form:
20200 C      LINEQ(M,N,A,X,B,CC)
20300 C
20400      SUBROUTINE LINEQ
20500      COMMON /EQU/    NEQU,NUNK,A(9,9),X(9),B(9)
20600      INTEGER CC
20700 C
20800 C      SOLVE AX=B.  T HOLDS AN UPPER TRIANGULAR MATRIX WHILE S
20900 C      IS WORKSPACE.  THE METHOD FACTORS A=U*T WHERE THE COLUMNS OF
21000 C      U ARE ORTHOGONAL AND T IS TRIANGULAR.  THE RESULTING SYSTEM
21100 C      T*X=B' IS EASILY SOLVED BY BACK SUBSTITUTION.  ASSUME M
21200 C      EQUATIONS AND N UNKNOWN.  ( N (= M) <= 9 )
21300 C      THE MATRIX OF COEFFICIENTS, A IS STORED IN THE FIRST N ROWS
21400 C      AND THE FIRST M COLUMNS OF THE 9X9 A ARRAY.  THE ROUTINE
21500 C      BRINGS IN THE WHOLE 9X9, BUT ONLY USES A(1,1) TO A(N,M)
21600 C      (RECALL THAT FORTRAN STORES THE ARRAY COLUMN-WISE, BUT

```

```

21700 C ADDRESSES THE ELEMENTS IN THE STANDARD ROW,COLUMN FORMAT)
21800 C NOTE: THE A ARRAY IS ALTERED DURING EXECUTION.
21900 C
22000 DIMENSION T(9,9)
22100 CC=1
22200 M = NEQU
22300 N = NUNK
22400 C M MUST BE <= 9, AND N<=M. CC IS A COMPLETION CODE; IF THE
22500 C SUBROUTINE EXECUTES PROPERLY CC WILL BE RESET TO 0 BEFORE RETURN
22600 DO 5 I=1,NUNK
22700 X(I) = 0.0
22800 5 CONTINUE
22900 DO 40 I=1,N
23000 IF (I.EQ.1) GO TO 25
23100 DO 20 J=1,M
23200 S=0
23300 I1=I-1
23400 DO 10 K=1,I1
23500 C IF (T(K,K) .LT. .0001) GO TO 5000
23600 S=S+A(J,K)*T(K,I)/T(K,K)
23700 10 CONTINUE
23800 A(J,I)=A(J,I)-S
23900 20 CONTINUE
24000 25 DO 40 K=I,N
24100 S=0
24200 DO 30 J=1,M
24300 S=S+A(J,I)*A(J,K)
24400 30 CONTINUE
24500 T(I,K)=S
24600 40 CONTINUE
24700 DO 60 I=1,N
24800 S=0
24900 DO 50 J=1,M
25000 S=S+A(J,I)*B(J)
25100 50 CONTINUE
25200 X(I)=S
25300 60 CONTINUE
25400 DO 80 I=1,N
25500 I1=N+1-I
25600 IF (I1.EQ.N) GO TO 75
25700 I2=I1+1
25800 DO 70 J=I2,N
25900 X(I1)=X(I1)-T(I1,J)*X(J)
26000 70 CONTINUE
26100 C IF (T(I1,I1).LT..0001) GO TO 5000
26200 75 X(I1)=X(I1)/T(I1,I1)
26300 80 CONTINUE
26400 CC=0
26500 RETURN
26600 5000 CC=-1
26700 C A COMPLETION CODE OF -1 INDICATES THAT THE SUBROUTINE
26800 C TRIED TO DIVIDE BY 0.
26900 RETURN
27000 END

```

```

27100 C
27200 C                >>>>>>>> SUBROUTINE UPDATE <<<<<<<<<
27300 C
27400 C    This routine updates the velocities of the window
27500 C    following the calculation of the target velocities
27600 C    relative to the window.  Sensitivity may be varied
27700 C    by the feedback factors in the array FEED.
27800 C
27900 C    SUBROUTINE UPDATE
28000 C        COMMON /WINDOW/ IWSIZE,WIND(5,5),SWIND(5,5),COORD(2,5,5)
28100 C        COMMON /EQU/    NEQU,NUNK,A(9,9),XLAMDA(9),B(9)
28200 C        COMMON /PARMS/  TIME,XSTEP,TSTEP,FEED(6)
28300 C        COMMON /COCHGS/ XTRANI,YTRANI,ECOSI,ESINI,
28400 C            1            XTRANW,YTRANW,ECOSW,ESINW,
28500 C            2            XVELI,YVELI,VMAGI,VROTI,
28600 C            3            XVELW,YVELW,VMAGW,VROTW
28700 C        DVELX = (ECOSW * XLAMDA(1) - ESINW * XLAMDA(2)) -
28800 C            1            (XLAMDA(3) * XTRANW - XLAMDA(4) * YTRANW)
28900 C        DVELY = (ESINW * XLAMDA(1) + ECOSW * XLAMDA(2)) -
29000 C            2            (XLAMDA(4) * XTRANW + XLAMDA(3) * YTRANW)
29100 C        DMAGV = XLAMDA(3)
29200 C        DVROT = XLAMDA(4)
29300 C
29400 C        XVELW = XVELW - FEED(1) * DVELX
29500 C        YVELW = YVELW - FEED(2) * DVELY
29600 C        VMAGW = VMAGW - FEED(3) * DMAGV
29700 C        VROTW = VROTW - FEED(4) * DVROT
29800 C
29900 C        RETURN
30000 C        END
30100 C
30200 C                >>>>>>>> SUBROUTINE MOVER <<<<<<<<<
30300 C
30400 C    This routine moves the window by taking a step in
30500 C    the differential equations for the affine transformation
30600 C    which controls the window location.  The method used is
30700 C    a simple Euler method.
30800 C
30900 C    The routine also simulates the motion of the target by
31000 C    solving the corresponding differential equation for the
31100 C    target.  This portion would be removed if real data were
31200 C    being used.
31300 C
31400 C    SUBROUTINE MOVER
31500 C        COMMON /PARMS/  TIME,XSTEP,TSTEP,FEED(6)
31600 C        COMMON /COCHGS/ XTRANI,YTRANI,ECOSI,ESINI,
31700 C            1            XTRANW,YTRANW,ECOSW,ESINW,
31800 C            2            XVELI,YVELI,VMAGI,VROTI,
31900 C            3            XVELW,YVELW,VMAGW,VROTW
32000 C        DECOS = VMAGW * ECOSW - VROTW * ESINW
32100 C        DESIN = VROTW * ECOSW + VMAGW * ESINW
32200 C        ECOSW = ECOSW + TSTEP * DECOS
32300 C        ESINW = ESINW + TSTEP * DESIN
32400 C

```

```

32500      DXTRAN = XVELW + VMAGW*XTRANW - VROTW*YTRANW
32600      DYTRAN = YVELW + VROTW*XTRANW + VMAGW*YTRANW
32700      XTRANW = XTRANW + DXTRAN*TSTEP
32800      YTRANW = YTRANW + DYTRAN*TSTEP
32900      C
33000      C      The following portions simulate motion of the target.
33100      C
33200      DECOS = VMAGI * ECOSI - VROTI * ESINI
33300      DESIN = VROTI * ECOSI + VMAGI * ESINI
33400      ECOSI = ECOSI + TSTEP * DECOS
33500      ESINI = ESINI + TSTEP * DESIN
33600      C
33700      DXTRAN = XVELI + VMAGI*XTRANI - VROTI*YTRANI
33800      DYTRAN = YVELI + VROTI*XTRANI + VMAGI*YTRANI
33900      XTRANI = XTRANI + DXTRAN*TSTEP
34000      YTRANI = YTRANI + DYTRAN*TSTEP
34100      C
34200      C      Increment time.
34300      C
34400      TIME = TIME + TSTEP
34500      C
34600      RETURN
34700      END
34800      C
34900      C      >>>>>>>>> SUBROUTINE COMPAR <<<<<<<<<<
35000      C
35100      C      This routine produces printed output for evaluation
35200      C      purposes, and is therefore ancillary to the operation
35300      C      of the tracker.
35400      C
35500      SUBROUTINE COMPAR
35600      COMMON /PARMS/  TIME,XSTEP,TSTEP,FEED(6)
35700      COMMON /COCHGS/ XTRANI,YTRANI,ECOSI,ESINI,
35800      1              XTRANW,YTRANW,ECOSW,ESINW,
35900      2              XVELI,YVELI,VMAGI,VROTI,
36000      3              XVELW,YVELW,VMAGW,VROTW
36100      WRITE(6,2000)TIME,XTRANI,YTRANI,ECOSI,ESINI,XVELI,YVELI,VMAGI,
36200      1              VROTI
36300      2000  FORMAT(2X,F3.2,8(3X,E11.4))
36400      WRITE(6,2010)XTRANW,YTRANW,ECOSW,ESINW,XVELW,YVELW,VMAGW,
36500      1              VROTW
36600      2010  FORMAT(10X,8(3X,E11.4))
36700      RETURN
36800      END
36900      C
37000      C      >>>>>>>>> FUNCTION FWIND <<<<<<<<<<
37100      C
37200      C      This function returns a value at the point (x,y)
37300      C      in the window. It first maps to the corresponding
37400      C      point in absolute image coordinates and calls for
37500      C      image value at that point (see FIMAGE below).
37600      C
37700      FUNCTION FWIND(X,Y)
37800      COMMON /COCHGS/ XTRANI,YTRANI,ECOSI,ESINI,

```

```

37900      1      XTRANW,YTRANW,ECOSW,ESINW,
38000      2      XVELI,YVELI,VMAGI,VROTI,
38100      3      XVELW,YVELW,VMAGW,VROTW
38200      C
38300      C
38400      C
38500      XIMAGE = XTRANW + ECOSW*X - ESINW*Y
38600      YIMAGE = YTRANW + ESINW*X + ECOSW*Y
38700      FWIND = FIMAGE(XIMAGE,YIMAGE)
38800      RETURN
38900      END
39000      C
39100      C      >>>>>>>>> FUNCTION FIMAGE <<<<<<<<<
39200      C
39300      C      This function returns the value at point (x,y) in the
39400      C      absolute image coordinate system. It maps to target
39500      C      coordinates and calls for the value at the corresponding
39600      C      point on the target (see FOBJ below).
39700      C
39800      FUNCTION FIMAGE(X,Y)
39900      COMMON /COCHGS/ XTRANI,YTRANI,ECOSI,ESINI,
40000      1      XTRANW,YTRANW,ECOSW,ESINW,
40100      2      XVELI,YVELI,VMAGI,VROTI,
40200      3      XVELW,YVELW,VMAGW,VROTW
40300      C
40400      C
40500      DET = ECOSI**2 + ESINI**2
40600      DX = X - XTRANI
40700      DY = Y - YTRANI
40800      C
40900      C
41000      XOBJ= ( ECOSI*DX + ESINI*DY)/DET
41100      YOBJ= ( -ESINI*DX + ECOSI*DY)/DET
41200      FIMAGE = FOBJ(XOBJ,YOBJ)
41300      RETURN
41400      END
41500      C
41600      C      >>>>>>>>> FUNCTION FOBJ <<<<<<<<<
41700      C
41800      C      This function returns the gray value at a point (x,y)
41900      C      in target coordinates. For actual tracking, this
42000      C      routine would be replaced by one which retrieves from
42100      C      the image database. In the simulation, however, the
42200      C      routine merely returns a synthetic value generated
42300      C      from an expression.
42400      C
42500      FUNCTION FOBJ(X,Y)
42600      FOBJ = 1.0 + 10.0*X -5.0*Y + 20.0*X*Y
42700      RETURN
42800      END

```

## APPENDIX B

### Tracking With Differential Forms

A tracking program was developed which utilized the theory presented in Section III. In order to test this program, digitized video images were obtained from the Advanced Technology Office, Instrumentation Directorate, White Sands Missile Range. One such image is shown in Figure B-1. A sequence of test images was prepared from this data by shifting to inject additional motion. Sections of the first six frames are shown in the left hand column of Figure B-2.

Parameters in the tracking program were set to use a 3x3 window in 3 consecutive frames to form a 3x3x3 rectangle in space. Since the program uses 9 such windows, the actual track gate consisted of 9x9x3 points.

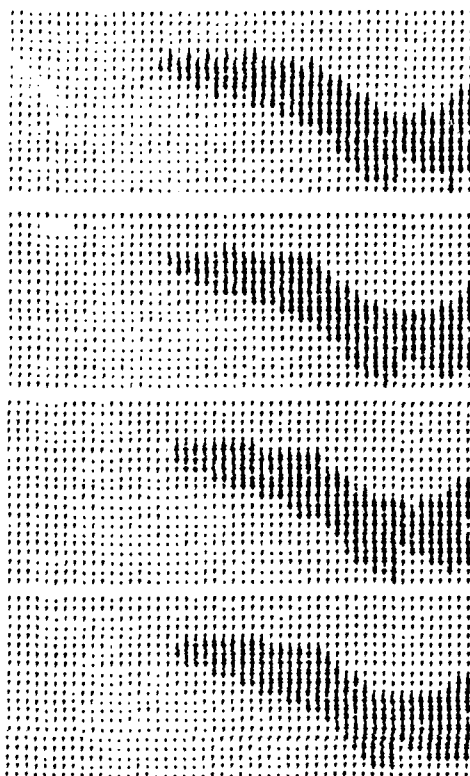
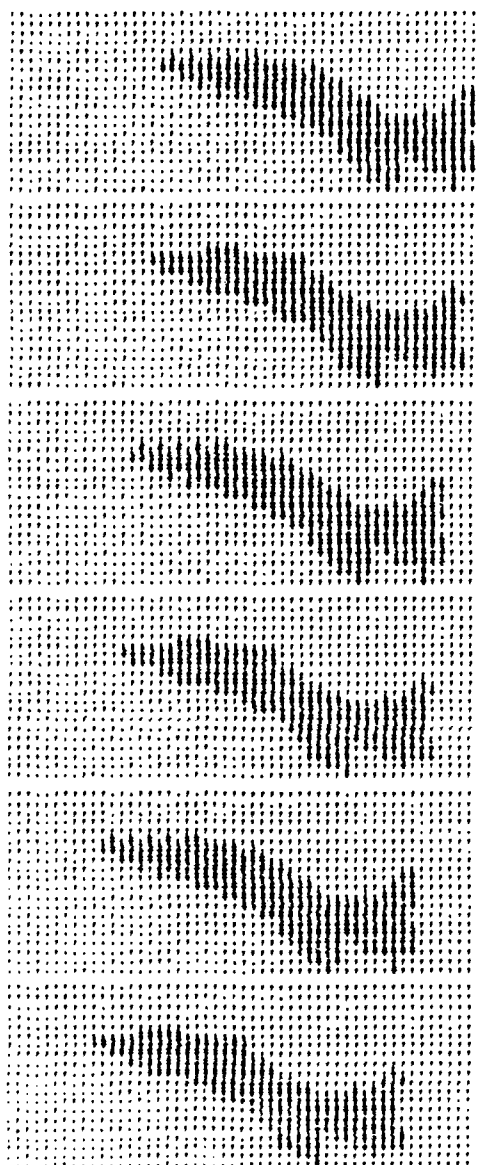
The right hand column of Figure B-2 shows the output frames obtained by the tracker. Observe that the output lags the input by the depth of the track gate. This is why there are fewer outputs frames than input frames. We see that the target was acquired immediately, and successfully tracked over the full sequence of frames.

The computation rate was about 10 frames per second on a VAX 11/780. However, the program is written to allow selection of window sizes and could be streamlined a great deal.

Although the results shown in Figure B-2 are impressive, we hasten to point out that the motion is mostly translation, and our attempts to test the algorithm on a wide range of motions have been frustrated by availability of data. Further work is ongoing, and refinements to the tracking program are expected to be forthcoming.







```

20* C      PROGRAMMER : DONNA K. TERRAL
40  C      DEPARTMENT OF MATHEMATICS
60  C      TEXAS TECH UNIVERSITY
80  C
100 C      PERMISSION IS HEREWITH GRANTED TO UTILIZE THIS PROGRAM FOR
120 C      OTHER THAN PERSONAL OR CORPORATIVE GAIN.
140 C
160 C
180 C *****
200 C ***** MAIN PROGRAM *****
220 C *****
240 C      PURPOSE: GIVEN A SET OF IMAGES, TRACK A TRAGET BY USING
260 C      INTEGRATION TO DETERMINE THE MOTIONS (TRANSLATION,ROTATION,
280 C      MAGNIFICATION) OF THE TARGET.
300 C *****
320 C
340 C      * COMMON DECLARATIONS *
360 C
380 C      PARAMETER (IW=9,JW=9,KW=3,IB=3,JB=3,KB=3,ISIZE=128,JSIZE=128)
400 C
420 C      COMMON /WINDOW/ WIND(IW,JW,KW),CORRD(ISIZE,JSIZE,2)
440 C      INTEGER WIND
460 C      COMMON /ORIGIN/ IO,J0,K0,X0,Y0
480 C      INTEGER IO,J0,K0,X0,Y0
500 C      COMMON /BUFFER/ BUF(IB,JB,KB)
520 C      INTEGER BUF
540 C      COMMON /CORNER/ X1,X2,Y1,Y2,T1,T2
560 C      INTEGER X1,X2,Y1,Y2,T1,T2
580 C      COMMON /WEIGHT/ W(IB,JB,KB),WX(JB,KB),WY(IB,KB),WT(IB,JB)
600 C      INTEGER W,WX,WY,WT
620 C      COMMON /COCHGS/ XVELI,YVELI,VROTI,VMAGI,TIME,TSTEP
640 C      REAL XVELI,YVELI,VROTI,VMAGI,TIME,TSTEP
660 C      COMMON /PARMS/ ROW,COL,PROW,PCOL,NUMROW,NUMCOL,WTEST
680 C      INTEGER ROW,COL,PROW,PCOL,NUMROW,NUMCOL,WTEST
700 C      COMMON /IMAGE/ IMAGE(ISIZE,JSIZE)
720 C      INTEGER*2 IMAGE
740 C      COMMON /EQU/ ALPHA(4),BETA,COEFF(9,4),VECTOR(9),SOLN(4),FEED(4)
760 C      INTEGER ALPHA,BETA
780 C
800 C      * MAIN VARIABLES *
820 C      INTEGER I1,J1,K1
840 C      INTEGER CC,COUNT
860 C      DATA COUNT,CC,K1 /1,1,1/
880 C
900 C      CALL INIT
920 C
940 C      *** MAIN LOOP ***
960 C      DO 100 LOOP = 1,4
980 C
1000 C      ALL OR A PORTION OF THE INTEGRATION RESULTS CAN BE USED
1020 C      SOLN(1) = FEED(1) * SOLN(1)
1040 C      SOLN(2) = FEED(2) * SOLN(2)
1060 C      SOLN(3) = FEED(3) * SOLN(3)
1080 C      SOLN(4) = FEED(4) * SOLN(4)

```

```

1100 C
1120 CALL WINDOW (LOOP)
1140 C
1160 CALL PRINT
1180 C
1200 C *** LOOP COMPUTES MOTION USED TO MOVE WINDOW ***
1220 DO 1 J1 = 1,JW,JB
1240 DO 2 I1 = 1,IW,IB
1260 CALL GETBUF (I1,J1,K1)
1280 IF (LOOP .NE. 1) GO TO 10
1300 CALL BLDW
1320 10 CALL GETEQU
1340 CALL MATRIX (COUNT)
1360 COUNT = COUNT + 1
1380 2 CONTINUE
1400 1 CONTINUE
1420 CALL LINEQ (COEFF,SOLN,VECTOR,9,4,CC)
1440 COUNT = 1
1460 C *** END MOTION LOOP ***
1480 C
1500 TIME = TIME + TSTEP
1520 100 CONTINUE
1540 C *** END MAIN LOOP ***
1560 C
1580 STOP
1600 END
1620 C
1640 C *****
1660 C *****
1680 C SUBROUTINE GETBUF FILLS A BUFFER ARRAY WHICH WILL CONTAIN THE
1700 C DATA POINTS FORMING THE CUBE TO BE USED IN SUBROUTINE GETEQU.
1720 C THE POSITION IN WHICH TO BEGIN IS PASSED THRU THE ARGUMENTS.
1740 C <BUF(1,1,1)=WIND(I1,J1,K1)>
1760 C *****
1780 C
1800 SUBROUTINE GETBUF (I1,J1,K1)
1820 C
1840 C * ARGUMENTS *
1860 C INTEGER I1,J1,K1
1880 C
1900 C * COMMON DECLARATIONS *
1920 C
1940 C PARAMETER (IW=9,JW=9,KW=3,IB=3,JB=3,KB=3,ISIZE=128,JSIZE=128)
1960 C
1980 C COMMON /WINDOW/ WIND(IW,JW,KW),CORRD(ISIZE,JSIZE,2)
2000 C INTEGER WIND
2020 C COMMON /ORIGIN/ IO,J0,K0,X0,Y0
2040 C INTEGER IO,J0,K0,X0,Y0
2060 C COMMON /BUFFER/ BUF(IB,JB,KB)
2080 C INTEGER BUF
2100 C COMMON /CORNER/ X1,X2,Y1,Y2,T1,T2
2120 C INTEGER X1,X2,Y1,Y2,T1,T2
2140 C
2160 C * LOCAL VARIABLES *

```

```

2180      INTEGER X,Y,T
2200  C
2220      X1 = I1 - I0
2240      Y1 = J1 - J0
2260      T1 = K1 - K0
2280      X2 = X1 + IB - 1
2300      Y2 = Y1 + JB - 1
2320      T2 = T1 + KB - 1
2340  C
2360      DO 30 I = 1,IB
2380          DO 30 J = 1,JB
2400              DO 30 K = 1,KB
2420                  X = I1 + I - 1
2440                  Y = J1 + J - 1
2460                  T = K1 + K - 1
2480                  BUF(I,J,K) = WIND(X,Y,T)
2500      30  CONTINUE
2520      RETURN
2540      END
2560  C
2580  C *****
2600  C *****
2620  C      SUBROUTINE MATRIX TAKES THE ALPHAS AND BETA FROM THE SUBROUTINE
2640  C      GETEQU AND PUTS THEM IN THE FORM AX=B WHERE LOOPS OF GETEQU
2660  C      FORM THE 2-DIMENSIONAL ARRAY A AND THE VECTOR B.
2680  C      ONCE AX=B IS FORMED LINEQ IS USED TO SOLVE FOR X.
2700  C
2720  C      IN MATRIX, COEFF(9,4) IS ARRAY A AND VECTOR(9) IS ARRAY B
2740  C *****
2760  C
2780      SUBROUTINE MATRIX (COUNT)
2800  C
2820  C      * ARGUMENTS *
2840      INTEGER COUNT
2860  C
2880  C      * COMMON DECLARATIONS *
2900      COMMON /EQU/ ALPHA(4),BETA,COEFF(9,4),VECTOR(9),SOLN(4),FEED(4)
2920      INTEGER ALPHA,BETA
2940  C
2960      COEFF(COUNT,1) = FLOAT(ALPHA(1))
2980      COEFF(COUNT,2) = FLOAT(ALPHA(2))
3000      COEFF(COUNT,3) = FLOAT(ALPHA(3))
3020      COEFF(COUNT,4) = FLOAT(ALPHA(4))
3040      VECTOR(COUNT) = FLOAT(BETA)
3060      RETURN
3080      END
3100  C
3120  C *****
3140  C *****
3160  C      SUBROUTINE GETEQU
3180  C      COMPUTES CONSTANTS ALPHA(1) THRU ALPHA(4) AND BETA IN THE
3200  C      EQUATION ALPHA(1) * LAMBDA(1) + ALPHA(2) * LAMBDA(2) +
3220  C      ALPHA(3) * LAMBDA(3) + ALPHA(4) * LAMBDA(4) = BETA, WHERE
3240  C      ALPHA(1) THRU ALPHA(4) AND BETA ARE FORMED FROM SURFACE AND

```

[illegible]

```

4340 C
4360 DO 50 I = 1,IB
4380 DO 50 J = 1,JB
4400 FXY1 = FXY1 + BUF(I,J,1) * WT(I,J)
4420 FXY2 = FXY2 + BUF(I,J,KB) * WT(I,J)
4440 50 CONTINUE
4460 DO 60 J = 1,JB
4480 DO 60 K = 1,KB
4500 FYT1 = FYT1 + BUF(1,J,K) * WX(J,K)
4520 FYT2 = FYT2 + BUF(IB,J,K) * WX(J,K)
4540 YFYT1 = YFYT1 + (J + Y1 - 1) * BUF(1,J,K) * WX(J,K)
4560 YFYT2 = YFYT2 + (J + Y1 - 1) * BUF(IB,J,K) * WX(J,K)
4580 60 CONTINUE
4600 DO 70 I = 1,IB
4620 DO 70 K = 1,KB
4640 FTX1 = FTX1 + BUF(I,1,K) * WY(I,K)
4660 FTX2 = FTX2 + BUF(I,JB,K) * WY(I,K)
4680 XFTX1 = XFTX1 + (I + X1 - 1) * BUF(I,1,K) * WY(I,K)
4700 XFTX2 = XFTX2 + (I + X1 - 1) * BUF(I,JB,K) * WY(I,K)
4720 70 CONTINUE
4740 DO 80 I = 1,IB
4760 DO 80 J = 1,JB
4780 DO 80 K = 1,KB
4800 FXYT = FXYT + BUF(I,J,K) * W(I,J,K)
4820 80 CONTINUE
4840 C
4860 ALPHA(1) = FYT2 - FYT1
4880 ALPHA(2) = FTX2 - FTX1
4900 ALPHA(3) = X2 * FYT2 - X1 * FYT1 - FXYT + Y2 * FTX2 - Y1 * FTX1
4920 ALPHA(4) = YFYT1 - YFYT2 + XFTX2 - XFTX1
4940 BETA = FXY1 - FXY2
4960 C
4980 RETURN
5000 END
5020 C
5040 C *****
5060 C *****
5080 C SUBROUTINE BLDW BUILDS THE WEIGHING ARRAYS FOR THE
5100 C TRAPEZIOD RULE USED IN THE SUBROUTINE GETEQU. THESE ARRAYS
5120 C ARE PASSED TO GETEQU THRU A COMMON BLOCK.
5140 C *****
5160 C
5180 C SUBROUTINE BLDW
5200 C
5220 C * COMMON DECLARATIONS *
5240 C
5260 C PARAMETER (IW=9,JW=9,KW=3,IB=3,JB=3,KB=3,ISIZE=128,JSIZE=128)
5280 C
5300 C COMMON /WEIGHT/ W(IB,JB,KB),WX(JB,KB),WY(IB,KB),WT(IB,JB)
5320 C INTEGER W,WX,WY,WT
5340 C
5360 C LOCAL VARAIRLES
5380 C INTEGER X,Y,T
5400 C

```

```

5420      X = IB - 1
5440      Y = JB - 1
5460      T = KB - 1
5480      C
5500      DO 40 I = 2,X
5520          DO 40 J = 2,Y
5540              DO 40 K = 2,T
5560                  W(I,1,1) = 2
5580                  W(I,JB,1) = 2
5600                  W(1,J,1) = 2
5620                  W(IB,J,1) = 2
5640                  W(1,JB,K) = 2
5660                  W(IB,1,K) = 2
5680                  W(I,1,KB) = 2
5700                  W(1,J,KB) = 2
5720                  W(IB,J,KB) = 2
5740                  W(I,JB,KB) = 2
5760                  W(IB,JB,K) = 2
5780                  W(1,1,K) = 2
5800                  W(I,J,K) = 8
5820                  W(1,J,K) = 4
5840                  W(I,1,K) = 4
5860                  W(I,J,1) = 4
5880                  W(I,JB,K) = 4
5900                  W(IB,J,K) = 4
5920                  W(I,J,KB) = 4
5940                  WY(I,1) = 2
5960                  WY(1,K) = 2
5980                  WY(I,K) = 4
6000                  WY(I,KB) = 2
6020                  WY(IB,K) = 2
6040                  WT(I,1) = 2
6060                  WT(1,J) = 2
6080                  WT(I,J) = 4
6100                  WT(I,JB) = 2
6120                  WT(IB,J) = 2
6140                  WX(JB,K) = 2
6160                  WX(J,KB) = 2
6180                  WX(1,K) = 2
6200                  WX(J,1) = 2
6220                  WX(J,K) = 4
6240      40      CONTINUE
6260                  W(1,1,1) = 1
6280                  W(1,JB,1) = 1
6300                  W(1,JB,KB) = 1
6320                  W(1,1,KB) = 1
6340                  W(IB,1,1) = 1
6360                  W(IB,JB,1) = 1
6380                  W(IB,1,KB) = 1
6400                  W(IB,JB,KB) = 1
6420                  WX(1,1) = 1
6440                  WX(1,KB) = 1
6460                  WX(JB,1) = 1
6480                  WX(JB,KB) = 1

```

```

6500      WY(1,1) = 1
6520      WY(1,KB) = 1
6540      WY(IB,1) = 1
6560      WY(IB,KB) = 1
6580      WT(1,1) = 1
6600      WT(1,JB) = 1
6620      WT(IB,1) = 1
6640      WT(IB,JB) = 1
6660      RETURN
6680      END
6700      C
6720      C *****
6740      C *****
6760      C      SUBROUTINE PRINT WRITES TO UNIT = 66 THE MOTION PARAMETERS
6780      C      OF THE IMAGE AND THE WINDOW AT EACH TIME STEP. THE PIXEL
6800      C      VALUES IN THE WINDOW MAY ALSO BE PRINTED IF NEEDED.
6820      C *****
6840      C
6860      SUBROUTINE PRINT
6880      C
6900      C      * COMMON DECLARATIONS *
6920      C
6940      C      PARAMETER (IW=9,JW=9,KW=3,IB=3,JB=3,KB=3,ISIZE=128,JSIZE=128)
6960      C
6980      C      COMMON /WINDOW/ WIND(IW,JW,KW),CORRD(ISIZE,JSIZE,2)
7000      C      INTEGER WIND
7020      C      COMMON /ORIGIN/ IO,JO,KO,XO,YO
7040      C      INTEGER IO,JO,KO,XO,YO
7060      C      COMMON /COCHGS/ XVELI,YVELI,VROTI,VMAGI,TIME,TSTEP
7080      C      REAL XVELI,YVELI,VROTI,VMAGI,TIME,TSTEP
7100      C      COMMON /PARMS/ ROW,COL,PROW,PCOL,NUMROW,NUMCOL,WTEST
7120      C      INTEGER ROW,COL,PROW,PCOL,NUMROW,NUMCOL,WTEST
7140      C      COMMON /EQU/ ALPHA(4),BETA,COEFF(9,4),VECTOR(9),SOLN(4),FEED(4)
7160      C      INTEGER ALPHA,BETA
7180      C
7200      C      * LOCAL VARIABLES *
7220      C      REAL XTRANW,YTRANW,ROTW,MAGW,XVELW,YVELW,VROTW,VMAGW,XTRANI,
7240      C      &      MAGI,ROTI,YTRANI
7260      C
7280      C      DATA XTRANW,YTRANW,ROTW,MAGW /0,0,0,0/
7300      C
7320      C      XTRANI = XVELI*TIME
7340      C      YTRANI = YVELI*TIME
7360      C      MAGI = VMAGI*TIME
7380      C      ROTI = VROTI*TIME
7400      C
7420      C      XTRANW = SOLN(1) + XTRANW
7440      C      YTRANW = SOLN(2) + YTRANW
7460      C      MAGW = SOLN(3) + MAGW
7480      C      ROTW = SOLN(4) + ROTW
7500      C
7520      C      XVELW = SOLN(1)*(1/TSTEP)
7540      C      YVELW = SOLN(2)*(1/TSTEP)
7560      C      VMAGW = SOLN(3)*(1/TSTEP)

```



```

7580      VROTW = SOLN(4)*(1/TSTEP)
7600      C
7620      IF (WTEST .EQ. 1) THEN
7640          ITEMP = COL - PCOL + (IW+1)/2
7660          JTEMP = ROW - PROW + (JW+1)/2
7680          WRITE(66,30) CORRDI(ITEMP,JTEMP,1),CORRDI(ITEMP,JTEMP,2)
7700      30      FORMAT (1X,'CENTER OF WINDOW AT (' F9.5',' F9.5,')',/)
7720      C
7740          WRITE (66,36)
7760      36      FORMAT (//1X,'WINDOW VALUES',/)
7780          DO 4 K = 1,KW
7800              DO 5 J = 1,JW
7820                  WRITE (66,6) (WIND(I,J,K),I=1,IW)
7840      6          FORMAT (1X,<IW>(I4))
7860      5          CONTINUE
7880              WRITE (66,8)
7900      8          FORMAT (/)
7920      4          CONTINUE
7940      C
7960          WRITE (66,10) TIME,XTRANI,YTRANI,MAGI,ROTI,XVELI,YVELI,
7980      &          VMAGI,VROTI
8000      10      FORMAT (1X,F5.4,8(3X,E11.4),/)
8020          WRITE (66,11)
8040      11      FORMAT(10X,'XTRANW',8X,'YTRANW',8X,'MAGW',10X,'ROTW',
8060      &          10X,'XVELW',9X,'YVELW',9X,'VMAGW',9X,'VROTW')
8080          WRITE (66,12) XTRANW,YTRANW,MAGW,ROTW,XVELW,YVELW,VMAGW,VROTW
8100      12      FORMAT (6X,8(3X,E11.4),/)
8120          WRITE (66,13)
8140      13      FORMAT('1')
8160      C
8180      ELSE
8200          WRITE (66,14) TIME,XTRANI,YTRANI,MAGI,ROTI,XVELI,YVELI,
8220      &          VMAGI,VROTI
8240      14      FORMAT(1X,F5.4,8(3X,E11.4))
8260          WRITE(66,15) XTRANW,YTRANW,MAGW,ROTW,XVELW,YVELW,VMAGW,VROTW
8280      15      FORMAT(11X,8(3X,E11.4),/)
8300      ENDIF
8320      RETURN
8340      END
8360      C
8380      C *****
8400      C *****
8420      C      SUBROUTINE INIT IS USED TO INITIALIZE PARAMETERS.
8440      C *****
8460      C
8480      C      SUBROUTINE INIT
8500      C
8520      C      * COMMON DECLARATIONS *
8540      C      PARAMETER (IW=9,JW=9,KW=3,IB=3,JB=3,KB=3,ISIZE=128,JSIZE=128)
8560      C
8580      C      COMMON /WINDOW/ WIND(IW,JW,KW),CORRDI(ISIZE,JSIZE,2)
8600      C      INTEGER WIND
8620      C      COMMON /ORIGIN/ IO,JO,KO,XO,YO
8640      C      INTEGER IO,JO,KO,XO,YO

```

```

8660      COMMON /PARMS/ ROW, COL, PROW, PCOL, NUMROW, NUMCOL, WTEST
8680      INTEGER ROW, COL, PROW, PCOL, NUMROW, NUMCOL, WTEST
8700      COMMON /COCHGS/ XVELI, YVELI, VROTI, VMAGI, TIME, TSTEP
8720      COMMON /EQU/ ALPHA(4), BETA, COEFF(9,4), VECTOR(9), SOLN(4), FEED(4)
8740      INTEGER ALPHA, BETA
8760      C
8780      C      THE PIXEL VALUE AT (COL,ROW) IS PUT INTO WIND(1,1,1)
8800      WRITE (6,20)
8820      20      FORMAT (1X, 'BEGIN WINDOW')
8840      WRITE (6,50)
8860      50      FORMAT (1X, 'ROW      COL')
8880      READ (5,*) ROW, COL
8900      C
8920      C      THE MOTION TO BE TRACKED
8940      WRITE (6,30)
8960      30      FORMAT (1X, 'INPUT: XVELI, YVELI, VROTI/PI, VMAGI')
8980      READ (5,*) XVELI, YVELI, VROTI, VMAGI
9000      C
9020      PROW = 1
9040      PCOL = 1
9060      NUMROW = 128
9080      NUMCOL = 128
9100      C
9120      X0 = COL + (IW-1)/2
9140      Y0 = ROW + (JW-1)/2
9160      C
9180      DO 60 I=1, NUMCOL
9200          DO 70 J=1, NUMROW
9220              CORRDI(I,J,1) = I
9240              CORRDI(I,J,2) = J
9260      70      CONTINUE
9280      60      CONTINUE
9300      C
9320      TSTEP = 0.033
9340      TIME = 2*TSTEP
9360      PI = 3.14159
9380      VROTI = VROTI*PI
9400      C
9420      IO = (IW+1)/2
9440      JO = (JW+1)/2
9460      KO = (KW+1)/2
9480      C
9500      DO 10 I=1, 4
9520          SOLN(I) = 0.0
9540      10      CONTINUE
9560      C
9580      FEED(1) = 1.0
9600      FEED(2) = 1.0
9620      FEED(3) = 1.0
9640      FEED(4) = 1.0
9660      C
9680      WRITE (6,40)
9700      40      FORMAT (1X, 'INPUT 1 cr. TO WRITE WINDOW OR 0 cr. TO SKIP')
9720      READ (5,*) WTEST

```

```

9740      IF (WTEST .EQ. 0) THEN
9760          WRITE (66,5) IB
9780      5      FORMAT (1X,'WINDOW SIZE = ',I2,/)
9800          WRITE (66,3)
9820      3      FORMAT(1X,'TIME',5X,'XTRANI',8X,'YTRANI',8X,'MAGI',10X,'ROTI'
9840          &      ,10X,'XVELI',9X,'YVELI',9X,'VMAGI',9X,'VROTI')
9860          WRITE (66,4)
9880      4      FORMAT(15X,'XTRANW',8X,'YTRANW',8X,'MAGW',10X,'ROTW',
9900          &      10X,'XVELW',9X,'YVELW',9X,'VMAGW',9X,'VROTW',/)
9920      ENDIF
9940      C
9960      RETURN
9980      END

10000     C
10020     C *****
10040     C *****
10060     C      SUBROUTINE WINDOW (1) USES THE INTEGRATION RESULTS TO MOVE THE
10080     C      WINDOW IN ORDER TO TRACK THE TARGET AND (2) PERFORMS THE MOTION
10100     C      ON THE 1ST IMAGE USED TO GET THE WINDOW AND WRITES THE RESULT
10120     C      TO UNIT = M + 15.
10140     C *****
10160     C
10180     C      SUBROUTINE WINDOW (M)
10200     C
10220     C      * COMMON DECLARATIONS *
10240     C
10260     C      PARAMETER (IW=9,JW=9,KW=3,IB=3,JB=3,KB=3,ISIZE=128,JSIZE=128)
10280     C
10300     C      COMMON /WINDOW/ WIND(IW,JW,KW),CORRD(ISIZE,JSIZE,2)
10320     C      INTEGER WIND
10340     C      COMMON /EQU/ ALPHA(4),BETA,COEFF(9,4),VECTOR(9),SOLN(4),FEED(4)
10360     C      INTEGER ALPHA,BETA
10380     C      COMMON /IMAGE/ IMAGE(ISIZE,JSIZE)
10400     C      INTEGER*2 IMAGE
10420     C      COMMON /BUFFER/ BUF(IB,JB,KB)
10440     C      INTEGER BUF
10460     C      COMMON /PARMS/ ROW,COL,PROW,PCOL,NUMROW,NUMCOL,WTEST
10480     C      INTEGER ROW,COL,PROW,PCOL,NUMROW,NUMCOL,WTEST
10500     C
10520     C      * LOCAL VARIABLES *
10540     C      LOGICAL*1 BIMAGE(ISIZE),BLINE(2*ISIZE,JSIZE)
10560     C      CHARACTER*(ISIZE) IMAGELINE
10580     C      INTEGER*2 NEWIM(ISIZE,JSIZE)
10600     C      REAL ECOSW,ESINW,XW,YW
10620     C      EQUIVALENCE (BIMAGE,IMAGELINE),(BLINE,NEWIM)
10640     C
10660     C      AFFINE TRANSFORMATION TO MOVE WINDOW
10680     C
10700     C      ITEMP = COL - PCOL + (IB+1)/2
10720     C      JTEMP = ROW - PROW + (JB+1)/2
10740     C      XW = CORRD(ITEMP,JTEMP,1)
10760     C      YW = CORRD(ITEMP,JTEMP,2)
10780     C      ECOSW = EXP(SOLN(3)) * COS(SOLN(4))
10800     C      ESINW = EXP(SOLN(3)) * SIN(SOLN(4))

```

```

10820 C
10840 DO 10 J=1,NUMROW
10860 DO 20 I=1,NUMCOL
10880 CORRDI(I,J,1)=ECOSW*(CORRDI(I,J,1)-XW)-ESINW*(CORRDI(I,J,2)
10900 & -YW)+ECOSW*SOLN(1)-ESINW*SOLN(2)
10920 CORRDI(I,J,2)=ESINW*(CORRDI(I,J,1)-XW)+ECOSW*(CORRDI(I,J,2)
10940 & -YW)+ESINW*SOLN(1)+ECOSW*SOLN(2)
10960 CORRDI(I,J,1) = CORRDI(I,J,1) + XW
10980 CORRDI(I,J,2) = YW + CORRDI(I,J,2)
11000 20 CONTINUE
11020 10 CONTINUE
11040 C
11060 C OPEN A IMAGE FILE AND FILL PIXEL VALUES INTO THE ARRAY IMAGE
11080 C
11100 DO 30 K=1,KW
11120 KCOUNT = M + K + 9
11140 OPEN(UNIT=KCOUNT,STATUS='OLD',ACCESS='DIRECT',
11160 & RECORDTYPE='FIXED',READONLY)
11180 C
11200 DO 80 I=1,ISIZE
11220 DO 90 J=1,JSIZE
11240 IMAGE(I,J) = 0
11260 90 CONTINUE
11280 80 CONTINUE
11300 C
11320 DO 40 J=1,NUMROW
11340 READ (KCOUNT,J) IMAGELINE
11360 DO 50 I=1,ISIZE
11380 IMAGE(I,J) = BIMAGE(I) .AND. 255
11400 50 CONTINUE
11420 40 CONTINUE
11440 C
11460 C USE THE TRANSFORMATION TO TRACK; STORE THE RESULT IN NEWIM
11480 C
11500 DO 60 J=1,NUMROW
11520 DO 70 I=1,NUMCOL
11540 IF (CORRDI(I,J,1) .LT. 1 .OR. CORRDI(I,J,1) .GT. ISIZE
11560 & .OR. CORRDI(I,J,2) .LT. 1 .OR. CORRDI(I,J,2) .GT. JSIZE)
11580 & THEN
11600 NEWIM(I,J) = 0
11620 ELSE
11640 NEWIM(I,J) = IBILIN(CORRDI(I,J,1),CORRDI(I,J,2))
11660 ENDIF
11680 70 CONTINUE
11700 60 CONTINUE
11720 C
11740 CLOSE (UNIT=KCOUNT)
11760 C
11780 C FILL WINDOW WITH NEW PIXEL VALUES
11800 C
11820 DO 110 I=1,IW
11840 DO 120 J=1,JW
11860 ITEMP = COL - PCOL + I
11880 JTEMP = ROW - PROW + J

```

```

11900          WIND(I,J,K) = NEWIM(ITEMP,JTEMP)
11920      120          CONTINUE
11940      110          CONTINUE
11960      C
11980      C          WRITE TRACKED IMAGE INTO A FILE
12000      C
12020          IF (K .EQ. 1) THEN
12040              OPEN(UNIT=KCOUNT+15,STATUS='NEW',ACCESS='DIRECT',
12060                  &          RECTYPE='FIXED',RECL=NUMCOL/4,BLOCKSIZE=NUMCOL)
12080              DO 130 J=1,NUMROW
12100                  DO 140 I=1,NUMCOL
12120                      BIMAGE(I) = BLINE(I*2-1,J)
12140      140              CONTINUE
12160                  WRITE (KCOUNT+15,J)IMAGELINE
12180      130              CONTINUE
12200              CLOSE (UNIT=KCOUNT+15)
12220          ENDIF
12240      C
12260      30          CONTINUE
12280          RETURN
12300          END
12320      C
12340      C *****
12360      C *****
12380      C
12400          SUBROUTINE LINEQ(A,X,B,M,N,CC)
12420          INTEGER CC
12440      C
12460      C      SOLVE AX=B.  T HOLDS  AN UPPER TRIANGULAR MATRIX  WHILE S
12480      C      IS WORKSPACE.  THE METHOD FACTORS A=U*T WHERE THE COLUMNS OF
12500      C      U ARE ORTHOGANAL AND T IS TRIANGULAR.  THE RESULTING SYSTEM
12520      C      T*X=B' IS EASILY SOLVED BY BACK SUBSTITUTION.  ASSUME M
12540      C      EQUATIONS AND N UNKNOWN.  ( N <= M <= 9 )
12560      C      THE MATRIX OF COEFFICIENTS, A IS STORED IN THE FIRST N ROWS
12580      C      AND THE FIRST M COLUMNS OF THE 9X9 A ARRAY.  THE ROUTINE
12600      C      BRINGS IN THE WHOLE 9X9, BUT ONLY USES A(1,1) TO A(N,M)
12620      C      (RECALL THAT FORTRAN STORES THE ARRAY COLUMN-WISE, BUT
12640      C      ADDRESSES THE ELEMENTS IN THE STANDARD ROW,COLUMN FORMAT)
12660      C      NOTE: THE A ARRAY IS ALTERED DURING EXECUTION.
12680      C
12700          DIMENSION A(9,9),T(9,9),X(N),B(M)
12720          CC=1
12740      C      M MUST BE <= 9, AND N<=M.  CC IS A COMPLETION CODE; IF THE
12760      C      SUBROUTINE EXECUTES PROPERLY CC WILL BE RESET TO 0 BEFORE RETURN
12780          DO 40 I=1,N
12800              IF (I.EQ.1) GO TO 25
12820              DO 20 J=1,M
12840                  S=0
12860                  I1=I-1
12880                  DO 10 K=1,I1
12900                      IF (T(K,K) .LT. .0001) GO TO 5000
12920                      S=S+A(J,K)*T(K,I1)/T(K,K)
12940      10              CONTINUE
12960                  A(J,I)=A(J,I)-S

```

```

12980      20      CONTINUE
13000      25      DO 40 K=I,N
13020          S=0
13040          DO 30 J=1,M
13060              S=S+A(J,I)*A(J,K)
13080      30      CONTINUE
13100          T(I,K)=S
13120      40      CONTINUE
13140          DO 60 I=1,N
13160              S=0
13180              DO 50 J=1,M
13200                  S=S+A(J,I)*B(J)
13220      50      CONTINUE
13240          X(I)=S
13260      60      CONTINUE
13280          DO 80 I=1,N
13300              I1=N+1-I
13320              IF (I1.EQ.N) GO TO 75
13340              I2=I1+1
13360              DO 70 J=I2,N
13380                  X(I1)=X(I1)-T(I1,J)*X(J)
13400      70      CONTINUE
13420              IF (T(I1,I1).LT..0001) GO TO 5000
13440      75      X(I1)=X(I1)/T(I1,I1)
13460      80      CONTINUE
13480          CC=0
13500          RETURN
13520      5000  CC=-1
13540      C      A COMPLETION CODE OF -1 INDICATES THAT THE SUBROUTINE
13560      C      TRIED TO DIVIDE BY 0.
13580          RETURN
13600          END
13620      C
13640      C *****
13660      C *****
13680      C
13700          INTEGER FUNCTION IBILIN*2(XX,YY)
13720      C
13740      C      THIS FUNCTION RECEIVES 2 REAL COORDINATES (PRODUCED BY THE TRANS-
13760      C      FORMATION IN THE CALLING ROUTINE) WHICH ARE COORDINATES RELATIVE
13780      C      TO THE OLD IMAGE. SINCE THE COORDINATES ARE REAL VALUED,
13800      C      THE POSITION WILL NOT BE ON A PARTICULAR PIXEL, BUT RATHER AMONG
13820      C      4. THIS FUNCTION RETURNS A BILINEAR INTERPOLATION FOR THE 4
13840      C      SURROUNDING POINTS.
13860      C
13880          PARAMETER (ISIZE=128,JSIZE=128)
13900          COMMON /IMAGE/ IMAGE(ISIZE,JSIZE)
13920          INTEGER*2 IMAGE
13940          REAL H,V,HTEMP1,HTEMP2,VTEMP
13960      C
13980          MX1=MAX(0,INT(XX))
14000          MX2=MIN(ISIZE,INT(XX)+1)
14020          MY1=MAX(0,INT(YY))
14040          MY2=MIN(JSIZE,INT(YY)+1)

```

```
14060      H=XX-MX1
14080      V=YY-MY1
14100      C
14120      HTEMP1=H*IMAGE(MX2,MY1)+(1.0-H)*IMAGE(MX1,MY1)
14140      HTEMP2=H*IMAGE(MX2,MY2)+(1.0-H)*IMAGE(MX1,MY2)
14160      VTEMP =V*HTEMP2+(1.0-V)*HTEMP1
14180      IBILIN=ININT(VTEMP)
14200      RETURN
14220      END
14240      C *****
14260      C *****
```

Appendix C

Adaptive Pattern Matching using Control

Theory on Lie Groups

by

Thomas G. Newman and Leopold Zlobee

Proceedings of the International Symposium  
on the Mathematical Theory of  
Networks and Systems

August, 1981



## ADAPTIVE PATTERN MATCHING USING CONTROL THEORY ON LIE GROUPS\*

Thomas G. Newman and Leopold Zlobec  
Texas Tech University  
Lubbock, Texas

Abstract

A method is given for matching a subpattern of a two-dimensional image against a stored prototype, where the latter is defined on a window whose position and shape is determined by the action of a Lie group of transformations. The method involves the construction of a path in the control group along which the matching error decreases to a local minimum.

## 1. INTRODUCTION

A problem of classical interest in pattern recognition is that of determining the presence or absence of a particular subpattern or subpattern class. In the analysis of two-dimensional imagery this can take the form of detection of corners and edges or the location of a specific silhouette. More particularly, we may be interested in obtaining an exact match of a specific portion of the image to a subimage, often a prototype, which may appear in an arbitrary manner, varying in size, location and orientation. This is the problem which is herein addressed.

A related question was considered by Dirilten and Newman [3] where it was shown

how two planar images could be matched under arbitrary affine transformation of the plane, if a match were at all possible. In addition to affine transformations, an allowance was also made for dilation of intensity scale such as that which results from under or over exposure of film within latitude limits. The results cited, however, are of little use in matching subpatterns, since the algorithms are highly sensitive to the background context. Nevertheless, the utility of a group theoretic approach to pattern matching was clearly demonstrated.

In the following we present a method for performing a local search for an imbedded subpattern of a two-dimensional image. The

\*This research was supported by the Army Research Office, Contract DAAG29-80-C-0087 and by the Office of Naval Research, Contract N0014-76-C-1136.

method is one involving adaptive control of a retina which seeks the desired sub-pattern by evolving along a curve in the space of parameters in a direction which assures improvement in the goodness of fit.

## 2. BACKGROUND

Let  $G$  be a Lie group of transformation on an analytic manifold  $M$ . Suppose  $G$  has dimension  $n$  while  $M$  has dimension  $m$ . Let  $x$  and  $y$  denote the coordinates of elements  $f$  and  $g$  in  $G$ , respectively, in a patch containing the identity element  $e$  of  $G$ . Also, let  $p$  denote coordinates of an element  $u$  of  $M$  in some patch in  $M$ . We may then express the coordinates  $z$  of the product  $h = fg$  and the coordinates  $q$  of the element  $v = gu$ , relative to suitable patches, by means of analytic functions

$$z = J(x, y) \quad (2.1)$$

$$q = K(y, p) \quad (2.2)$$

$K$  and  $J$  are vector-valued, having values in  $n$ -dimensional space  $R^n$  or  $C^n$  and  $m$ -dimensional space  $R^m$  or  $C^m$ . Hereafter we shall assume that these underlying spaces are real. We denote the  $i$ th component of  $J$  by  $J_i$  and the  $j$ th component of  $K$  by  $K_j$ .

In order to define the Lie algebra of  $G$  we first introduce real-valued maps on  $G$  by

$$P_{ij}(x) = \frac{\partial J_i}{\partial y_j}(x, y) \Big|_{y=e}, \quad (2.3)$$

where  $i$  and  $j$  each range from 1 to  $n$ . The cross-section  $P_{*j}$ , which consists of the  $P_{ij}$  as  $i$  ranges from 1 to  $n$ , and  $j$  is fixed, may be thought of as a vector field in  $R^n$ . Such a vector field attaches to a point  $x$  the vector  $P_{*j}(x)$ . As such,  $P_{*1}, P_{*2}, \dots, P_{*n}$  form a basis for the tangent space at the point  $x$  [1,2]. The infinitesimal transformations of  $G$  may now be defined by

$$X_j = \sum_{i=1}^n P_{ij}(x) \frac{\partial}{\partial x_i}, \quad (2.4)$$

for  $j = 1, 2, \dots, n$ .

The differential operators so defined are to be considered as linear operators on the space of analytic functions on  $G$ , or, more generally, on the space of differentiable functions on  $G$ . The Lie algebra of  $G$  is simply the  $n$ -dimensional vector space consisting of all linear combinations of these operators, and will be denoted by  $L(G)$  [2]. The Lie algebra of  $G$  may also be defined in terms of its actions on the manifold  $M$ . Analogous to (2.3) we define

$$Q_{ij}(p) = \frac{\partial K_j}{\partial y_i}(y, p) \Big|_{y=e} \quad (2.5)$$

for  $i = 1, 2, \dots, m$  and  $j = 1, 2, \dots, n$ . Finally, as in (2.4) above we set

$$X'_j = \sum_{i=1}^m Q_{ij} \frac{\partial}{\partial p_i}. \quad (2.6)$$

The operators  $X'_1, X'_2, \dots, X'_n$  apply to functions defined on  $M$  and span a Lie algebra isomorphic to  $L(G)$ .

The following result from [4] will be used later, and is stated for reference:

**Theorem 2.1.** Let  $f: M \rightarrow R$  be differentiable and define  $F: G \times M \rightarrow R$ , in terms of coordinates, by

$$F(x, p) = f(K(x, p)). \quad (2.7)$$

Then for each  $j = 1, 2, \dots, n$  we have

$$X_j F = X'_j F. \quad (2.8)$$

Let us consider a curve  $t \rightarrow g(t)$  in  $G$  satisfying  $g(0) = e$ . In terms of a coordinate patch at  $e$ ,  $g(t)$  may be described by a curve  $x(t)$  in  $R^n$  satisfying  $x(0) = 0$ . We shall consider the case in which  $x(t)$  is given as the solution of an evolution equation of the form

$$\dot{x}(t) = \sum_{i=1}^n u_i(t) P_{*i}(x(t)), \quad x(0) = 0, \quad (2.9)$$

where  $P_{*1}, \dots, P_{*n}$  are cross-sections of the array of functions given by (2.3), and  $u_1(t), \dots, u_n(t)$  are suitable control functions.

Now let  $p$  denote the coordinates of a point  $u$  in some coordinate patch. For a differentiable map  $f: M \rightarrow R$  we may define  $H: R \times M \rightarrow R$  by setting

$$H(t, p) = f(g(t)u). \quad (2.10)$$

We recognize that  $H(t, p) = F(x(t), p)$  where  $F$  is the extension of  $f$  to  $G \times M$  as in Theorem 2.1 above. From the point of view of application, if we regard  $f: M \rightarrow R$  as an image, then  $H(t, p)$  represents the moving image obtained by translation due to the curve  $g(t)$ . Also from [4], we have

**Theorem 2.2.** In the context above,

$$\frac{\partial H}{\partial t} = \sum_{i=1}^n \lambda_i(t) X_i^* H. \quad (2.11)$$

### 3. THE CONTROL MODEL

By an image we mean a map  $f: M \rightarrow R$ , where the value  $f(p)$  at a point  $p \in M$  represents the gray value at the picture element at  $p$ . In practice, values are observed on a subset  $W \subset M$ , which we regard as a window which may be translated by the action of  $G$  on  $M$ . Thus, upon translation by an element  $x \in G$ , the value observed at  $p \in W$  is given by  $F(x, p) = f(X(x, p))$ , as in (2.7) above.

We consider a given prototype sub-image  $V$  defined on the window  $W$ ,  $V: W \rightarrow R$ . The problem then is to determine  $x \in G$  such that  $F(x, p) = V(p)$  for all  $p \in W$ , or determine that no such  $x$  exists. As a matter of practice, we seek  $x \in G$  which minimizes the objective function

$$\Psi(x) = \frac{1}{2} \int_W (F(x, p) - V(p))^2 dp, \quad (3.1)$$

where  $dp$  represents a volume element and the integral is over the window  $W$ , which is assumed to be of bounded volume.

In general, for any two functions  $f_1, f_2: W \rightarrow M$  we define

$$\langle f_1, f_2 \rangle = \int_W f_1 f_2 dp \quad \text{and} \\ \|f_1\| = \langle f_1, f_1 \rangle^{1/2}.$$

Thus,  $\Psi(x) = \|F - V\|^2/2$ , where  $x$  is regarded as a parameter.

The following is a well-known property of the Lie group  $G$  [2]:

**Lemma 1.** In order that the differential  $d(x) = 0$  at a point  $x \in G$ , it is necessary and sufficient that each  $X_i^* \Psi(x) = 0$  where  $X_1, X_2, \dots, X_n$  are the generators of  $L(G)$  given by (2.4).

By direct calculation, we obtain  $X_i^* \Psi(x) = \int_W (F(x, p) - V(p)) X_i^* F(x, p) dp$ . In practice, this expression is difficult to compute numerically, due to the presence of the term  $X_i^* F$ , which cannot be computed directly from observed data. However, by Theorem (2.1) we have  $X_i^* F = X_i^* H$ , and the latter can be calculated from a single value of  $x$ .

Suppose now that a curve in  $G$  is given by coordinates  $x(t)$  obtained as a solution of Equation (2.9). We seek to find  $\lambda(t) = (\lambda_1(t), \dots, \lambda_n(t))$  so that  $\psi(t) = \Psi(x(t))$  decreases to a minimum value. Defining  $H(t, p) = F(x(t), p)$  we obtain,

$$\dot{\psi}(t) = \int_W (H(t, p) - V(p)) \frac{\partial H}{\partial t}(t, p) dp \quad (3.2)$$

which, by application of Theorem (2.2), becomes

$$\begin{aligned} \dot{\psi}(t) &= \sum_{i=1}^n \lambda_i(t) \int_W (H(z, p) - V(p)) X_i^* H(t, p) dp \\ &= \sum_{i=1}^n \lambda_i(t) \langle H - V, X_i^* H \rangle \end{aligned} \quad (3.3)$$

Upon observing that  $\langle H - V, X_i^* H \rangle = \langle F - V, X_i^* F \rangle = X_i^* \Psi$  at  $x = x(t)$ , we deduce:

**Theorem 3.1.** If  $\lambda_i(t)$  is chosen so that  $\text{sgn} \lambda_i(t) = -\text{sgn} \langle H - V, X_i^* H \rangle$ , we have

$\dot{\psi}(t) \leq 0$  for all  $t$ , with equality at  $t = t_0$  if and only if  $d\Psi = 0$  at  $x = x(t_0)$ .

Among the class of bounded controls,

$|\lambda_i(t)| \leq 1$ , we see that the rate of decrease of  $\psi(t)$  is maximized by the choice

$$\lambda_i(t) = -\text{sgn} \langle H - V, X_i^* H \rangle, \quad (3.4)$$

for  $i = 1, 2, \dots, n$ . Of course, other strategies can be formulated, including steepest descent, and some methods using unbounded controls. By proceeding along trajectories defined by the solution of (2.9) with  $x(t)$  given by (3.4), we approach a critical point of  $V$  (i.e.  $dV/dt = 0$ ). Since maxima and saddle points are unstable under perturbation, in practice this extreme point will always be a minimum.

#### 4. SIMULATION RESULTS

The results discussed in the previous section have been implemented by a discrete algorithm and tested on simulated data [5]. A digitized two-dimensional image was first generated in the form of a large two-dimensional array, and the prototype was generated in a  $20 \times 20$  window array.

The image space was assumed to be subject to translation, magnification and rotation, giving rise to a four parameter Lie group of transformations in the plane,  $R^2$ .

A number of cases were considered, including some involving multiple (false) targets and others in which the prototype was absent from the image being searched. In some cases the image was contaminated by 5% random noise. In all cases the search was started with overlap between the prototype target and the image target.

The differential equation (2.9) was solved by means of a Runge-Kutta fourth order method, with a dynamic step size, which was increased as necessary to accelerate convergence and decreased as necessary to maintain stability. Integration was replaced by summation, although we conjecture that convergence could have been accelerated by the use of a trapezoid rule.

Generally, search times ranged from 30 to 50 steps, with the longer search times prevailing for the more difficult cases.

In all cases, the final results were quite reasonable, even in those cases where the prototype was absent. In the latter cases, the search terminated with a "best" match, with a commensurately large final error.

As an example, Figure 1 shows that starting position for a noisy image containing two objects. The prototype is indicated by the central silhouette, while the true target is shifted upward, slightly to the right and is reduced in size. A false target overlaps the lower right corner of the prototype.

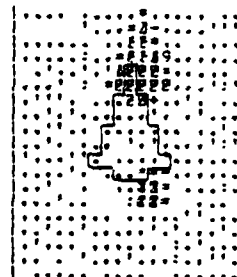


Fig. 1. Initial Window Position.

The termination conditions are shown in Figure 2, where the true target was located after 49 steps. All parameters were correct with the exception of magnification, which was about 5% too large. Smaller values of magnification, however, increase the error due to the presence of the false object, which is barely touching the bottom edge of the window in Figure 2.

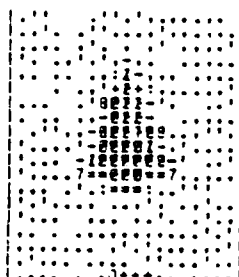


Fig. 2. Terminal Window Position.

#### BIBLIOGRAPHY

- [1] Auslander, L., Differential Geometry, Harper and Row, New York, 1967.
- [2] Cohn, P. M., Lie Groups, Cambridge University Press, London, 1957.
- [3] Dirilten, H. and T. G. Newman, Pattern Matching under Affine Transformations, IEEE Trans. Comp., Vol. C-24, No. 12, 1975, pp. 1191-1201.
- [4] Newman, T. G. and D. A. Demus, Lie Theoretic Methods in Video Tracking, Proceedings of the MICOM Workshop on Imaging Trackers and Autonomous Acquisition Applications, Redstone Arsenal, Nov. 1979.
- [5] Zlobec, L., Pattern Matching by Means of Adaptive Control, Masters Report, Texas Tech University, 1980.

DATE  
FILMED

-8