Real Time Vision System for Collision Detection

Alok Mittal — Aditya Valilaya
Subhashis Banerjee — M. Balakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
New Delhi 110016
Email: {suban,mbala}@cse.iitd.ernet.in

Abstract

Collision detection is an essential feature for any Real-Time Active Vision system to safeguard it against imminent collisions. In this paper, we describe the design and implementation of a real-time (25Hz) collision detection and avoidance system. The implementation is based on a theory originally proposed in [5, 4]. A relative motion between an observer and an object induces a corresponding image velocity, which is divergent in nature when the object moves towards the camera. In this system, we detect imminent collision from the divergent image flow. The implementation is based on a combination of spatial parallelism and pipelining. We have developed a general purpose library for implementing a pipeline application. This library can be ported on to any multiprocessing machine supporting inter-process communications. To achieve real-time performance, we have developed the collision detection system using the pipe library on a network of T805 transputers. A mobile robot with a camera mount has been interfaced with the transputer network and applications illustrating collision detection and obstacle avoidance have been developed.

1 Introduction

Any mobile active vision system [1, 2] should have the capability to safeguard itself against accidental collision with surrounding objects, both due to its own motion and the motion of other objects [3, 4, 5, 6, 7, 8]. In this paper we describe the design and implementation of a real-time system for collision detection based on computer vision. The system consists of a video camera mounted on a mobile robot. The camera grabs images at the rate of 25 frames/sec (25 Hz) and feeds them to a vision engine. The vision algorithms detect an imminent collision due to a looming motion and reacts by generating a panic signal for the robot to take the necessary preventive action [9, 10]. We also present a detailed experimental analysis of the collision detection system on a number of real image sequences.

Our implementation of collision detection is based on a theory originally proposed in [5, 4]. The relative motion of an observer with respect to an object produces a time-varying image on the observer's retina, which can be represented by an image flow field. An image flow field is a two dimensional velocity field defined over the image plane. The image flow is also referred to as optic flow [11, 12]. In a small field of view, the first-order derivatives of the image flow are sufficient to determine the bounds on time-to-collision [4, 6]. The image flow field can be decomposed in terms of the three differential invariants: *divergence, curl and deformation*. The bounding values of the time-to-collision are related to these invariants by simple algebraic relations.

Our implementation of the vision algorithms is based on parallel processing on a network of transputers. A purposive and reactive vision system such as this must have fast vision algorithms; both to enable short latency reactions to real-world events and to provide sufficiently high frequency results with small enough delays so that the controllers remain stable. In particular, the frame rate (number of image frames/sec) that can be processed directly determines the maximum relative velocity between the camera and an object that can be handled, because the optic flow equation [11] is valid only for infinitesimal motions between successive frames. An obvious way to increase throughput, at any given processor speed, is to use parallel processing. We have chosen to implement our vision algorithms on a network of transputers, primarily because these devices provide a flexible, software reconfigurable parallel processing environment. All the vision processing discussed in this paper run on eight 32-bit T805 transputers with clock speed of 25 MHz. Each of these devices are equipped with four bi-directional links which are fully integrated into the model of concurrency, facilitating the construction of communication protocols between several vision processes. Further, these links support a data transfer rate of 20Mb/sec facilitating fast transfer of image data between processes. We use a combination of spatial parallelism and pipelining in our vision architecture to achieve real-time performance. Note that our implementation is in no way specific to the transputer network and can be ported on to any reconfigurable parallel processing machine which supports a high communication bandwidth, or even on to a fast single processor machine supporting multiprocessing and inter-process communications. We have developed general purpose pipeline protocols and have developed our parallel vision software using these. A detailed comparative feasibility analysis [9, 10] of the problem on the 486/487 and transputer platforms show that the transputer solution is much faster and it is not possible to obtain a throughput of 25 frames/sec using the 486/487.

The rest of the paper is organized as follows. Section 2 deals with the theoretical aspects of the problem and describes an algorithm for calculating the upper and lower bounds on time to collision. Section 3 gives the software implementation of pipeline protocols and the application. Here we present a concise description of the general purpose pipe library and the way it has been

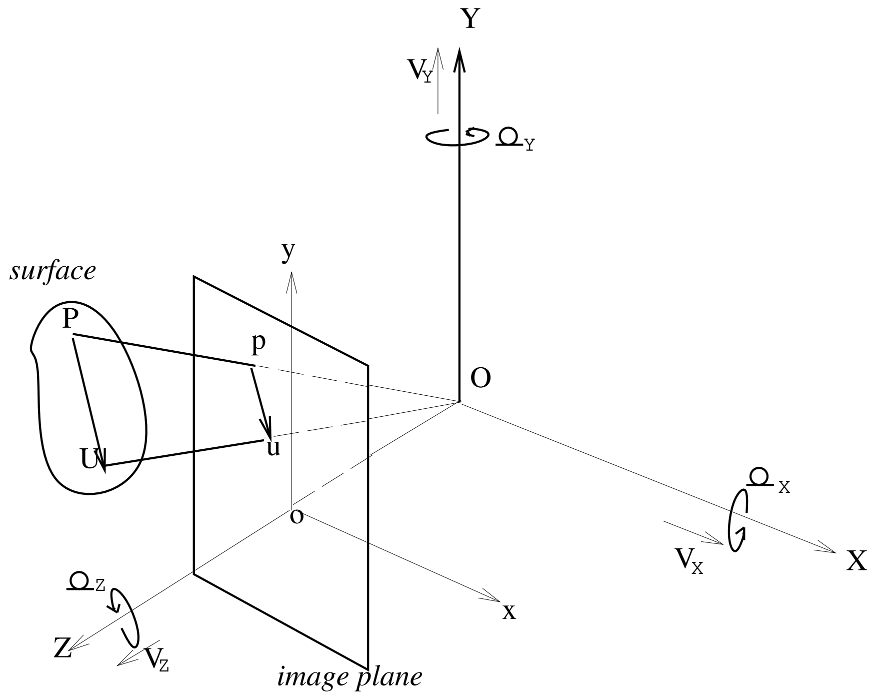

Figure 1: Camera Geometry

used for the specific task. In Section 4 we present some experimental results and discuss the system limitations.

2 Theory and Algorithm

In this section we describe the theory and the algorithm for the estimation of the time-to-contact from the image velocity field. The treatment follows [4]. Consider the camera model as illustrated in Fig. 1. The origin of a Cartesian coordinate system $OXYZ$ forms the *focus* and the Z-axis is aligned with the *optical axis*. Without loss of generality, the *image plane* is assumed to be at unit distance from the origin perpendicular to the optical axis. The image coordinate system oxy on the image plane has its origin at (0,0,1) and is aligned such that the x and y axes are, respectively, parallel to the X and Y axes.

Let the relative motion of the camera with respect to a rigid surface be described by a translational velocity (V_X, V_Y, V_Z) and a rotational velocity $(\Omega_X, \Omega_Y, \Omega_Z)$. Let the image velocity field (induced due to the relative motion) at a point (x, y) near the origin be represented as a 2D vector field $\overline{\mathrm{v}}(x, y)$ with x and y components (u, v). We assume that between two successive frames the motion is infinitesimal. This assumption, in turn, imposes limits on the velocities the system can handle. We discuss these limits later in Section 4.

2.1 Differential Invariants of the Image velocity field and time-to-collision

The time-to-collision between an object and an observer in relative motion can be estimated from the differential invariants of the image velocity field [13, 5, 4]. For a sufficiently small field of view or for a small neighbourhood around the origin, the image velocity field can be described by a translation in the image (u_0, v_0) and by the first-order partial derivatives of the image velocity (u_x, u_y, v_x, v_y), where [13, 16, 4]:

$$\begin{array}{lllllll} u_0 & = & -U_1/\lambda - \Omega_2 & \quad & v_0 & = & -U_2/\lambda + \Omega_1 \\ u_x & = & U_3/\lambda + U_1\lambda_x/\lambda^2 & & u_y & = & \Omega_3 + U_1\lambda_y/\lambda^2 \\ v_x & = & -\Omega_3 + U_2\lambda_x/\lambda^2 & & v_y & = & U_3/\lambda + U_2\lambda_y/\lambda^2 \end{array} \tag{2.1}$$

here, the x and y subscripts represent differentiation with respect to these spatial parameters and λ is the distance (along Z-axis) between observer (focus) and the object. The above equations relate the image velocity field to the relative motion between the viewer and the object, the distance λ and the local surface orientation (λ_x, λ_y).

An image feature or shape will be transformed as a result of the image velocity field. Under the assumptions of (i) a small field of view and (ii) a small depth range of the object compared to the viewing distance (i.e., $\Delta\lambda/\lambda << 1$), the transformation in shape between times t and t + δt (for small δt) can be approximated by a linear (affine) transformation [13]. Here we assume that, due to the small field of view, the visible object surface is essentially planar and there is no occlusion. To first order, the image velocity field at a point (x, y) in the neighbourhood of the origin can be approximated by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} + \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + O(x^2, xy, y^2) \tag{2.2}$$

where $O(x^2, xy, y^2)$ represents non-linear terms which are neglected in the analysis. The first term is a vector $[u_0, v_0]$ representing pure translation while the second term is a 2×2 velocity gradient tensor which represents the distortion of the image shape.

It is well known, from the Cauchy-Stokes decomposition theorem [14], that the velocity gradient tensor can be decomposed into the first-order differential invariants of the image velocity field $\overline{\mathrm{v}}$ - the curl, divergence and pure deformation. These quantities are defined by [14]:

$$\begin{array}{lll} div(\overline{\mathrm{v}}) & = & (u_x + v_y) \\ curl(\overline{\mathrm{v}}) & = & -(u_y - v_x) \\ def(\overline{\mathrm{v}})\cos 2\mu & = & (u_x - v_y) \\ def(\overline{\mathrm{v}})\sin 2\mu & = & (u_y + v_x) \end{array} \tag{2.3}$$

where μ specifies the orientation of the axis of expansion. The curl, divergence and the magnitude of the deformation are scalar invariants and do not depend on the particular choice of coordinate system [13]. The axes of maximum and minimum extension and contraction rotate with the rotations of the image plane axes.

The relationships between the differential invariants and the 3D configuration and relative motion between the camera and the object are given by equations 2.1 and 2.3. In particular, the differential invariants are expressed in terms of the viewer translation $(U_1/\lambda, U_2/\lambda, U_3/\lambda)$ and the surface orientation $(\lambda_x/\lambda, \lambda_y/\lambda)$.

Defining $\mathbf{A} = (U_1/\lambda, U_2/\lambda)$, $\mathbf{F} = (\lambda_x/\lambda, \lambda_y/\lambda) = \mathbf{grad}\lambda/\lambda$ and $\mathbf{Q}$ to be a unit vector along the view direction, we have from equations 2.1 and 2.3 [13, 4]:

$$
\begin{aligned}
curl\ \overline{\mathrm{v}} &= -2\Omega.\mathbf{Q} + |\mathbf{F} \times \mathbf{A}| \\
div\ \overline{\mathrm{v}} &= 2\mathbf{U}.\mathbf{Q}/\lambda + \mathbf{F}.\mathbf{A} \\
def\ \overline{\mathrm{v}} &= |\mathbf{F}|\,|\mathbf{A}|
\end{aligned}
\tag{2.4}
$$

where μ bisects $\mathbf{A}$ and $\mathbf{F}$:

$$\mu = \frac{(\angle\mathbf{A} + \angle\mathbf{F})}{2}$$

Thus, time-to-collision, which is given by $t_c = \lambda/\mathbf{U}.\mathbf{Q}$ (the ratio of the distance between the camera and the object and the component of the translational velocity along the Z-axis) is obtained in terms of the differential invariants. From Eqn. 2.4, it follows that time-to-contact must lie in an interval given by

$$1/t_c = \frac{div\ \overline{\mathrm{v}} \pm def\ \overline{\mathrm{v}}}{2} \tag{2.5}$$

The upper and lower estimates of the time-to-contact are equal when there is no deformation, and in such a case the time-to-contact is inversely proportional to the divergence. Note, that the deformation is entirely due to lateral motion (U_1, U_2). Thus, when there is a component of the translational motion parallel to the image plane causing deformation in the image flow field, it is important to work with the lower bound for the time-to-contact.

2.2 Extraction of differential invariants

We extract the differential invariants using a global linear velocity fit to the image motion data in a foveal region around the origin. To this end we use the optic flow technique [11, 17, 12]. Although gradient based optic flow was found in earlier structure from motion studies [18] to be too error prone for detailed quantitative recovery of scene structure, recent work on qualitative motion understanding has revived interest in its use [12, 15, 19, 20], and this is the approach we adopt. The global linear velocity field to the optic flow data is valid for a small field of view and planar surfaces, and this is adequately supported by our experimental results presented later in Section 5.The residual error of the global least squares fit turns out to be negligible. The complete algorithm is given as follows:

- Subsample the images to a low resolution. We subsample a foveal image of size 128×128 (out of a total image of size 512×512) to a size of 64×64 (by averaging). Since the differential invariants are independent of the choice of the coordinate scale, they can be recovered from the subsampled images without any loss of accuracy. We have found in the course of experimentation that the subsampling aids the subsequent vision processing by introducing a smoothing of the images (thus reducing noise) and it does not adversely affect the quality of the results. All subsequent processing takes place at this coarse resolution.

- Calculate the spatial and temporal image derivatives $E_x = \partial E/\partial x$, $E_y = \partial E/\partial y$ and $E_t = \partial E/\partial t$ from the last two images using finite difference technique [11, 17]. Henceforth, we ignore those pixels for which $\sqrt{E_x^2 + E_y^2} < T_1$ (a threshold determined by experimentation).

- For infinitesimal motion at each pixel, the optic flow (motion constraint) equation is given by [11, 17]

$$uE_x + vE_y + E_t = 0$$

Though $\overline{v} = (u, v)$ cannot be explicitly determined from the equation, the normal image velocity, i.e. the component of $\overline{v}$ parallel to $\nabla.E = (E_x, E_y)$ is given by [17]

$$V_\perp = E_t / \sqrt{E_x^2 + E_y^2}$$

We label as MOVING those vectors for which $|V_\perp| > T_2$ (a threshold determined by experimentation).

- The differential invariants are obtained by substituting equation 2.2 into the optic flow equation given above for the points labeled as MOVING, and carrying out a global least square fit for the parameters u_0 v_0 u_x u_y v_x v_y. The optic flow equation can be put into a standard measurement equation form for least square solution as:

$$Z_i = H_i \ \underline{X}$$

where $\underline{X} = (u_0 \ v_0 \ u_x \ u_y \ v_x \ v_y)^t$,
$H_i = (E_{x_i} \ E_{y_i} \ x_iE_{x_i} \ y_iE_{x_i} \ x_iE_{y_i} \ y_iE_{y_i})$, and
$Z_i = -E_{t_i}$
The least square solution for $\underline{X}$ minimizes
$J(\underline{X}) = \sum_i (Z_i - H_i\underline{X})^t \ (Z_i - H_i\underline{X})$
Differentiating this equation, the stationary values of $\hat{X}$ satisfies

$$H \ \hat{X} = Z \tag{2.6}$$

where, $H = \sum_i (H_i^t H_i)$ and $Z = \sum_i (H_i^t Z_i)$
We solve equation 2.6 using LU decomposition [21].

- Finally, from the global least-square-fit parameters, the differential invariants and the time-to-collision are computed according to equations 2.3 and 2.5.

3 Software Implementation

In this section, we first describe the general purpose pipe library which allows for a combination of spatial parallelism and pipelining, and a dynamic reinitialization facility. The general purpose pipe library, developed in the C programming language, facilitates the implementation of vision algorithms and architecture on any multiprocessing machine with facilities for inter-process communication. This gives flexibility to system design though at the cost of additional overheads. Next we move on to describe the implementation of the collision detection algorithm using the pipe library on the transputer network.

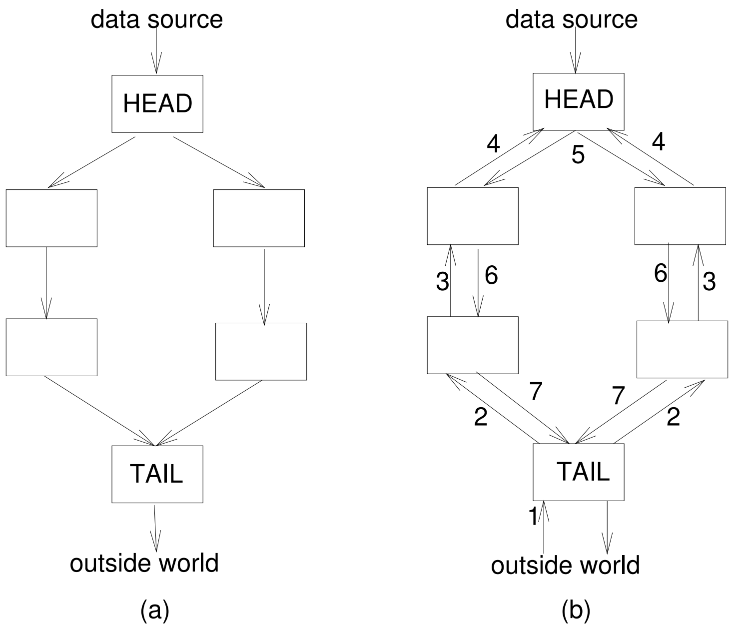

Figure 2: a) Example of pipe-group b) Steps for stopping data flow in a pipe-group

3.1 The General Purpose Pipe Library

The pipe-group software architecture has been designed so as to

- Maximize the throughput of each pipeline through synchronization of concurrent input, process execution and output processes.
- Allow dynamic re-initialization of all processors in a pipe-group with minimal interruption to normal data-flow.
- Incorporate the ability to modify the strict pipe-group structure without compromising the simplicity of the existing architecture.

The basic idea is that data enters the head processor, and is split between the pipelines. Each pipeline processor executes its part of the data processing on the input data stream and passes it on to the next processor. The tail processor merges the incoming result data, performs the final stage of the algorithm and outputs the result (refer to Fig. 2(a)).

We provide for *Dynamic Reinitialization* which obviates the necessity of rerunning or recompilation if parameters are to be changed. Stopping a pipe-group to prepare for reinitialization is accomplished by the following stages, illustrated in Fig. 2(b).

- The tail processor receives new parameters (step 1).
- Signals are sent up the pipelines while normal data flow continues down the pipelines (step 2-4).
- The head processor receives the signal and returns stop signals down the pipeline. It stops normal data flow (step 5).

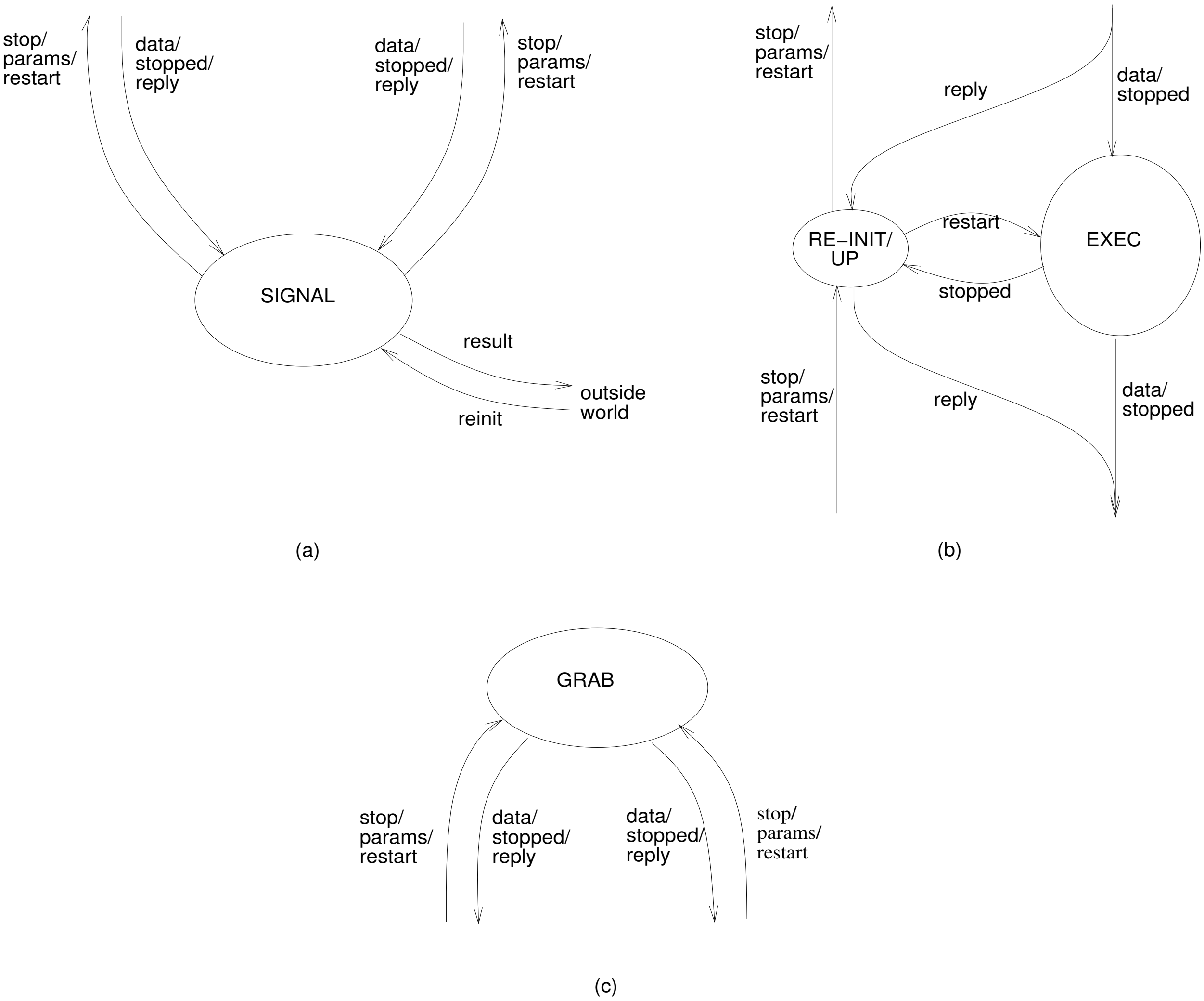

Figure 3: Arrangement of Processes on Processors: a) Tail b) Pipeline c) Head

- Each processor recognizes the stop signal, passes it onto the next processor and enters reinitialization mode (step 6-7).
- The tail processor receives the stop signal from all pipelines, indicating that data-flow has stopped.

Now reinitialization may proceed. Each processor is sent its parameters by the tail process. A processor picks up the packet it recognizes as its own and passes the rest up. Correct receipt of the parameters is signaled by acknowledgements (NO_PROBLEM signal). Finally the tail process sends PIPE_RESTART to all pipelines signaling them to resume normal functioning.

In what follows, we present the conceptual organization of the various processes:

The Tail Process

The Tail process (Fig. 3(a)) has the following functionalities:

1. inputs intermediate result data or stopped signals

2. merges intermediate result data into a single representation, executes the final stage of the algorithm, and outputs the result

3. reinitializes the pipe-groups, typically upon request by an external process

The Pipeline Processes

A pipeline process (Fig. 3(b)) is organized in terms of two independent processes - EXEC and RE-INIT/UP. The functionalities of the EXEC process are:

1. inputs data or stopped signals

2. executes relevant stages of the algorithm and outputs the new computed data to the next process down the pipeline

3. passes on the stopped signals to RE-INIT and the next process down the pipeline and then waits for restart signal from RE-INIT/UP

The functionalities of the RE-INIT/UP process are:

1. waits for the stop signal PIPE-STOP from down the pipeline, relays it up the pipeline and waits for the stopped signal from the EXEC.

2. reinitializes the process as explained earlier, signals EXEC to restart upon receiving the signal from down the pipeline and loops

If anything other than the PIPE-STOP is sent up the pipeline, it is transmitted up.

The Head Process

The functionalities of the Head process (Fig. 3(c)) are:

1. Gets the data (image) from the frame grabber, performs the initial stages of the algorithms, splits data spatially and outputs it to the pipelines.

2. stops execution on receiving the stop signal from the processes down the pipeline and passes the stopped signal down the pipeline

3. performs reinitialization and waits for restart signal. Upon receiving the restart signal, it starts sending new data down the pipeline.

In what follows we describe implementation of the collision detection using the pipe library described above.

3.2 Implementation of Collision Detection

The collision detection application has been developed using the pipe library described above. The experimental results demonstrating the correctness of the implementation are presented in the next section.

Various design options to place various processes on different transputers were considered. Fig. 4 illustrates the final structure. The GRAB process is the head process. It divides the image of size 128×128 pixels into two equal parts and sends one to each pipeline. The averaging (AVG) process subsamples this sub-image from 128×64 to 64×32 and passes it down to the GRADIENT process. The GRADIENT process finds gradients in x,y and time directions. It passes its results down to the FLOW processor which sets up the H and Z matrices based on the gradient and gray level

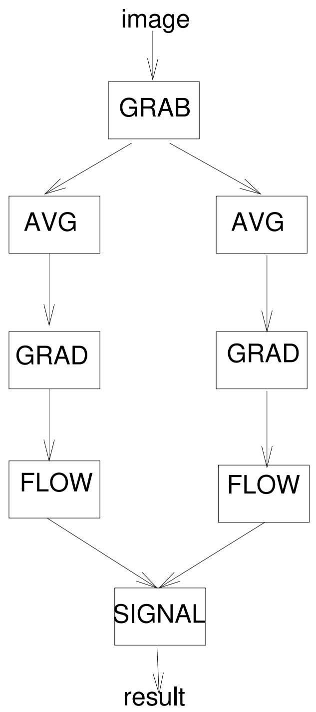

Figure 4: Final Arrangement of Processes on Processors for Collision Detection

values of the MOVING pixels. The results are merged by tail (SIGNAL) processor and the velocity of approach is calculated by solving the matrix equation 2.6 by LU decomposition. On basis of this, a collision is signaled if the minimum time-to-collision is less than a pre-determined threshold.

The software described above has been implemented on a network of T805 transputers running at 25MHz. A Quintek Frame Grabber Board (based on a single T805 transputer) is used to grab images at 25 Hz frame rate. These images are fed to the transputer network. The transputers (T805s) are organized on two boards having four transputers each. An ET-18 HERO mobile robot is interfaced to the transputer boards using the CENTRONICS interface. The transputer and the frame grabber boards are mounted on a PC 286 host system. A COHU-4710 camera is mounted on the robot to capture images [9, 10]. We have also implemented an off-line version on an HP9000/735 Workstation using the same software architecture.

In what follows we give a brief performance analysis of the system. The transputers run with a clock speed of 25 MHz and with a link speed of 20 Mbps. At these settings, the effective link speed was found to be 14.23 Mbps. The communication across transputers is:

from	to	K Bytes/frame
GRAB	AVG	8
AVG	GRADIENT	2
GRADIENT	FLOW	12
FLOW	SIGNAL	-

The computation time, communication time, total time and overheads measured on each transputer are as follows:

Transputer	Computed Time (ms)	Communication Time (ms)	Total Time (ms)	Overheads (ms)
AVG	4.91	5.62	26	15.47
GRADIENT	8.4	7.9	41	24.7
FLOW	9.05	6.75	36	20.2
SIGNAL	0.13	-	8	7.8

The GRAB process takes 40 ms. Thus the overall processing time for a single frame is 41 ms implying a frame rate of about 25 Hz.

Two applications have been built on the Robot. They are:

1. *Collision Avoidance:* In this application, the Robot normally remains idle. When it detects an imminent collision, it moves a fixed distance back and again stabilizes. This kind of application illustrates the use of the system in an environment where the Robot may act as a base for various equipments, sensors etc. and needs to protect itself against imminent collision due to movement of other objects.

2. *Obstacle Avoidance:* In this application, the Robot moves forward. If it detects an obstacle, it stops and moves back a fixed distance. Then it shifts laterally from its original path and tries another path. This application illustrates a possible use of the system in Dynamic Path Planning and Vision Guided Vehicles.

4 Results

In this section we present some typical experimental results of the performance of the collision detection system.

Example 1 In Fig. 5, we show a sequence of a book (telephone directory) being moved towards the camera from a distance of about 55cm to 25cm with an approximate speed of 25cm/s. We also show the normal image flow and the first order global velocity fit. In Fig. 6, we show the graphs of the differential invariants and the estimated time-to-collision plotted against the frame numbers of the sequence.

In this case the motion is predominantly along the view direction. The rotational components of the velocity and the translational components parallel to the image plane are essentially uncorrelated in the different frames, and they are present, in some frames, only due to a shake of the hand holding the book in forward motion. It follows from Eqn. 2.4 that the curl component of the image flow field is affected only by the rotation of the object around the viewing direction (cyclo-rotation) and translation parallel to the image plane. The deformation component is entirely due to a translation parallel to the image plane. The divergence has a component due to the translation along the view direction which is inversely proportional to the time-to-contact. The other component of the divergence is entirely due to the deformation of the image velocity field.

Note that the results are qualitatively correct. The divergence increases with the frame numbers and the time-to-contact decreases. This is a direct consequence of moving the object towards the camera. The curl and the deformation do not show any correlation with the position of the object as they are introduced only due to a shake of the hand.

The validity of the global least square fit for the estimation of the differential invariants is demonstrated by a plot of the residual error of the least square fit against the frame numbers of the sequence Fig. 7. Note that the residual error is negligible.

Example 2 In Fig. 8 and Fig. 9, we show similar results for a sequence in which a hand moves towards the camera. In Fig. 10 we show a plot of the residual error. Note that in this example the texture in the image is considerably less than that in the previous example. We obtain similar qualitative results as in Example 1.

The qualitative results of the type indicated above are sufficient to trigger an alarm signal for imminent collision reliably. We have tested the system extensively under varying lighting conditions with a large number of different objects. In our real-time application of collision detection we generate a panic signal when the time-to-collision is below a pre-determined threshold in three successive frames.

However, the system has several limitations. The most important of these being limits on the maximum speeds that the system can handle. Both experimentally and theoretically, we find the maximum approach speed for which the system can detect collision is of the order of 30-40cm/s at a distance of 50cm from the camera at 25Hz. Beyond this speed range, the image velocities become too large resulting in gross residual errors in the least-square-fit. Note that the optic flow equation is not valid for very large image velocities at the given rate of discretization.

Other limitations of the system include the dependence on the thresholds (T_1 and T_2 in the algorithm of section 2.2) which have to be changed with different lighting conditions. However, for a given lighting condition, we have determined specific values for the thresholds ($T_1 = 4.0$ and $T_2 = 0.2$).

Finally, for the system to work, it is imperative that enough texture be present in the foveal image for the optic flow to be detected. The hand image sequence of Example 2 gives an indication of the amount of texture information required for the system to work reliably.

5 Conclusion

In this paper, we have presented the design and implementation of a vision based system which can detect imminent collision and take preventive action in real-time, i.e., 25Hz. It can handle relative velocities (between an observer and an object at a distance of 50cm moving towards each other) up to 40-50cm/s. We have also built a general purpose pipe library which can be used for any pipelining application. It provides for dynamic reinitialization of the pipeline without the need to recompile the code.

We have implemented a Centronics interface on the Robot. This can be used to control the mobile robot using a PC printer port directly from the transputer network.

This work has provided a starting point from where we can plan to develop more advanced applications in active machine vision. These include applications in tracking of moving objects, redirection of attention due to sudden motion, real-time gaze control, object manipulation and automatic path planning. Thus, this system can be considered as a first step towards the development of real-time, multi-purpose active vision systems.

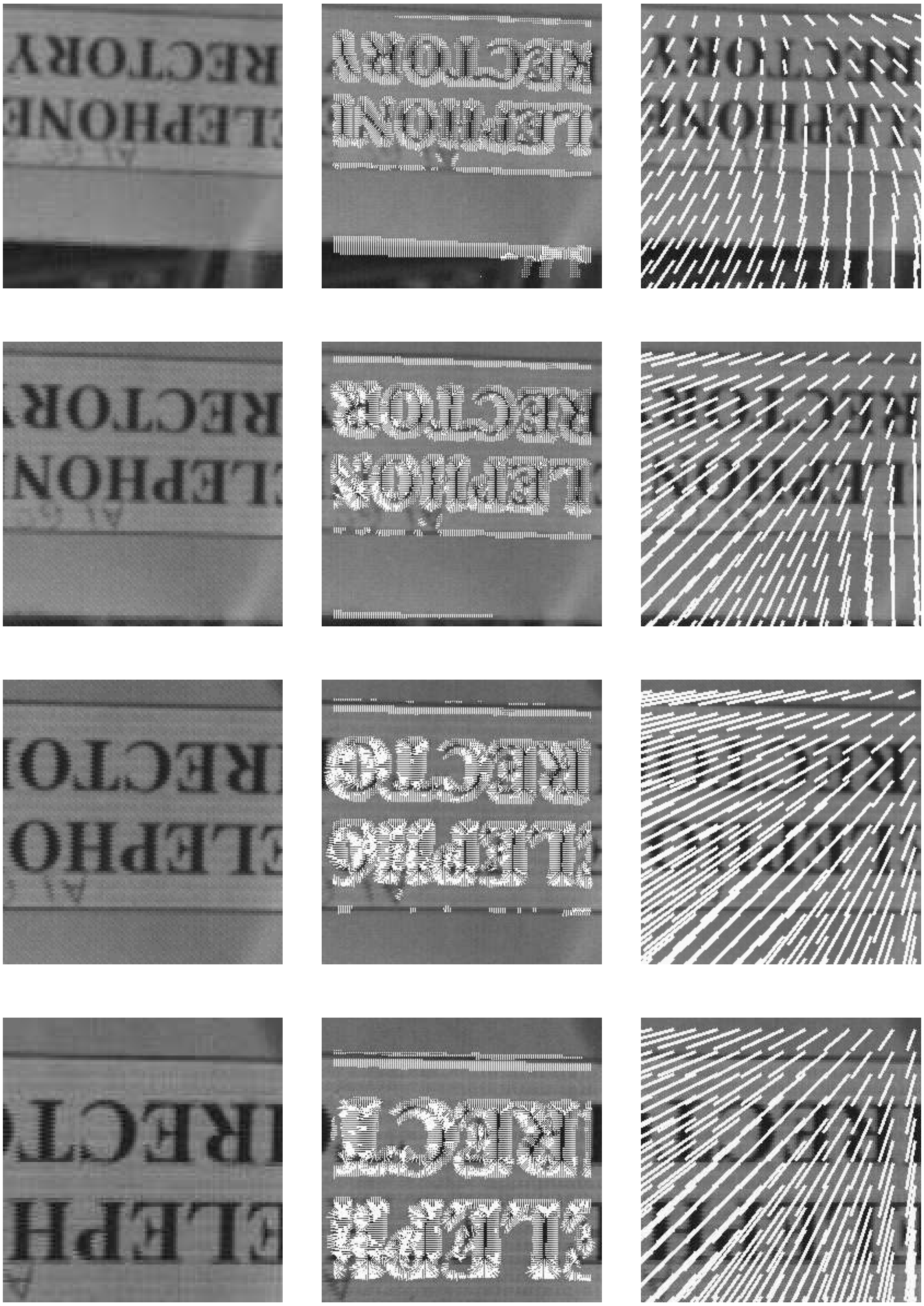

Figure 5: Frames 1,7,14,21, Image flows and 1st Order fits (Tel. directory)

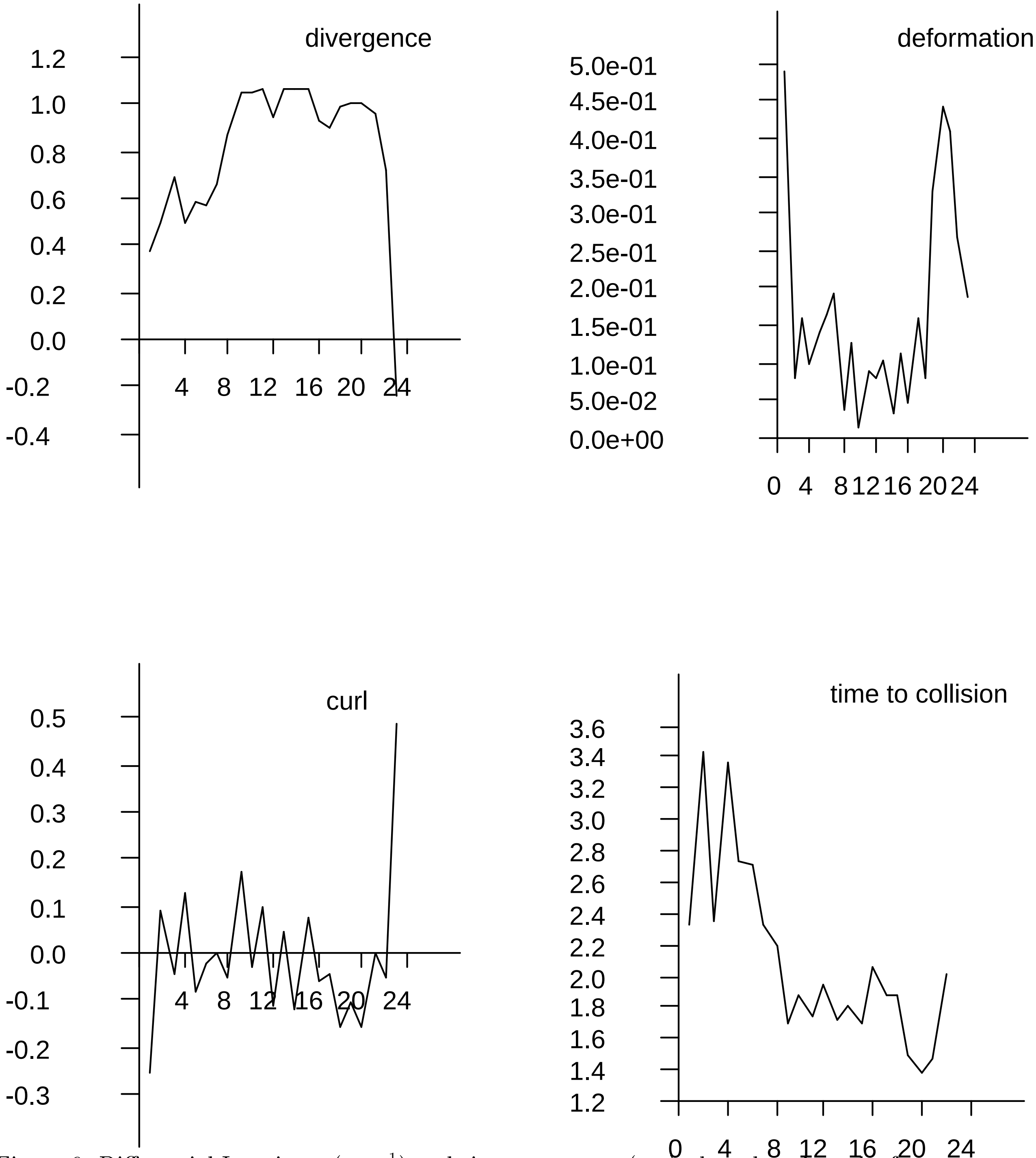

Figure 6: Differential Invariants (sec^{-1}) and time-to-contact (sec) plotted against the frame numbers (Tel. directory)

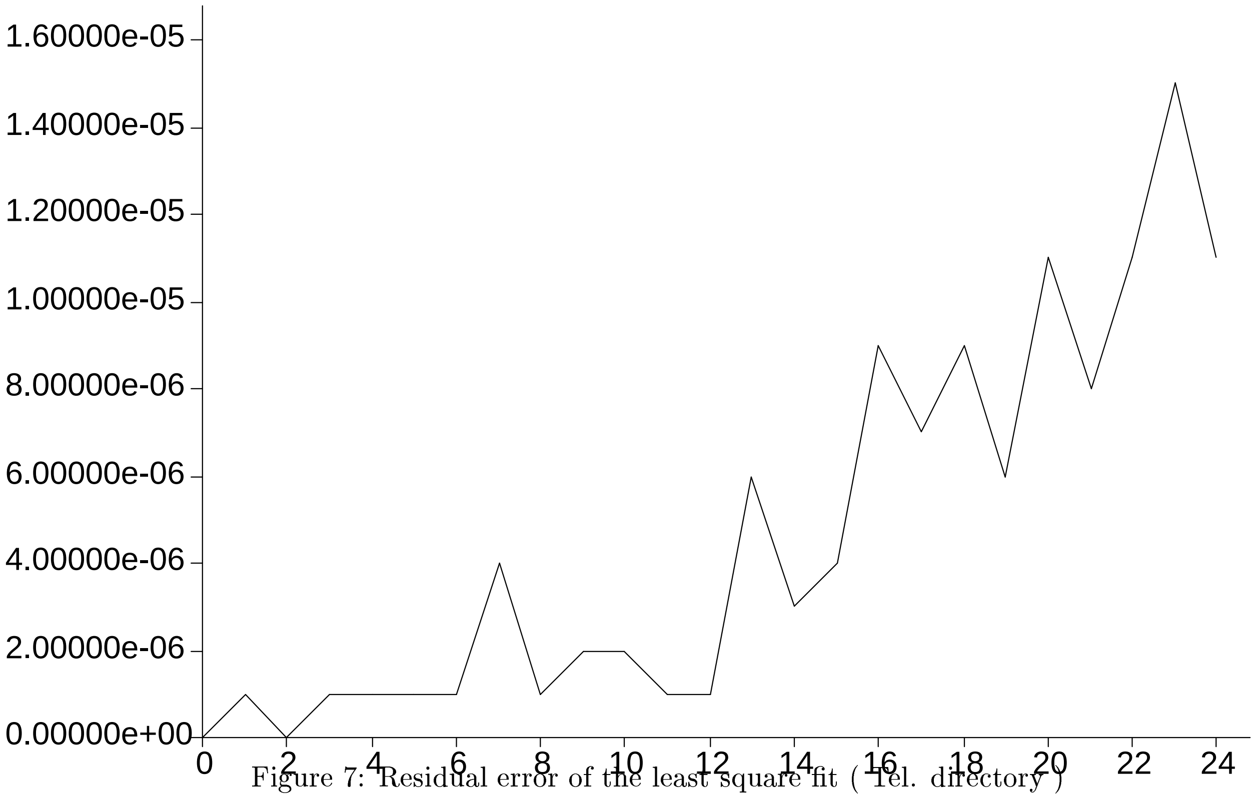

Figure 7: Residual error of the least square fit (Tel. directory)

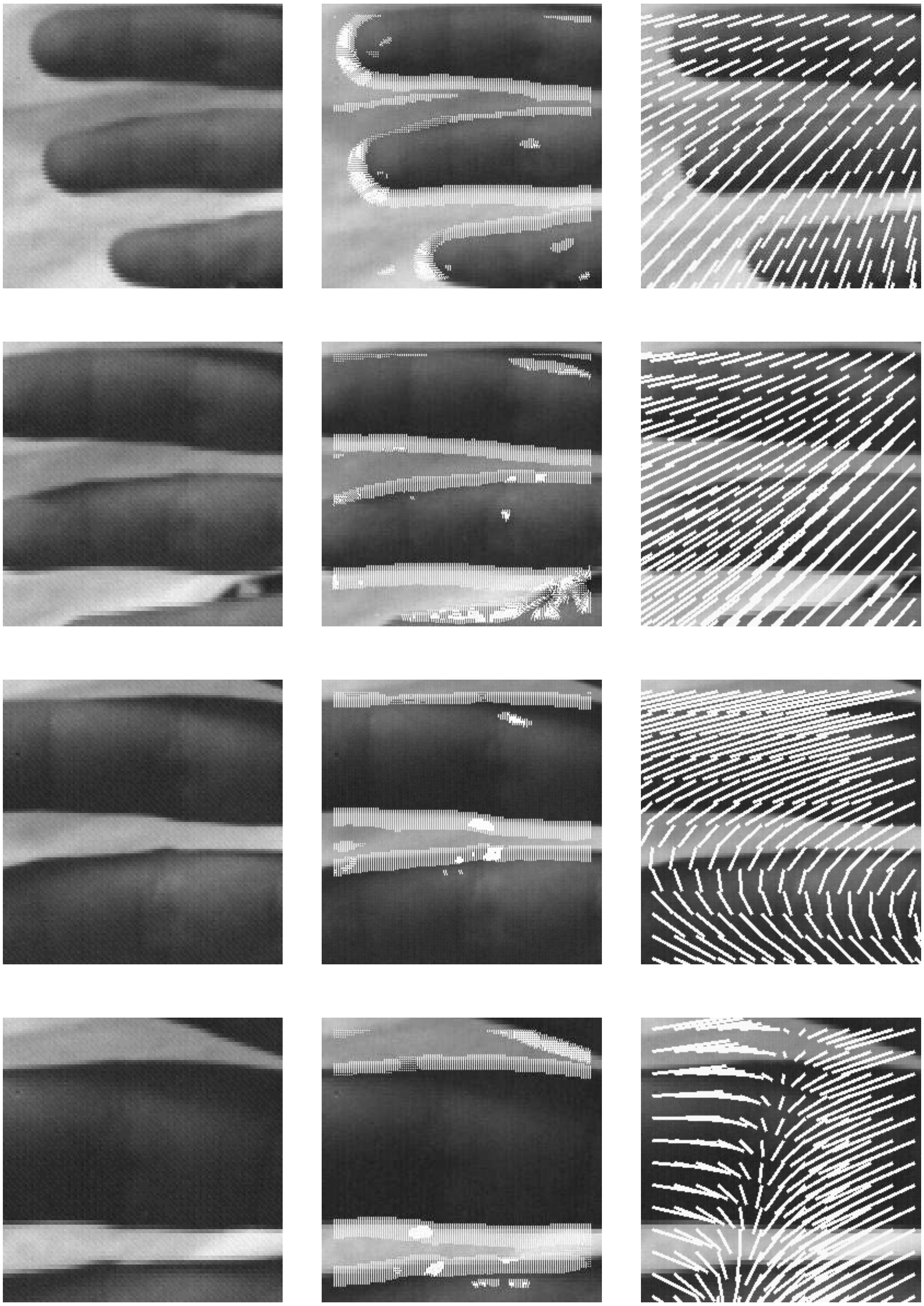

Figure 8: Frames 1,7,14,21, Image flows and 1st Order fits (Hand)

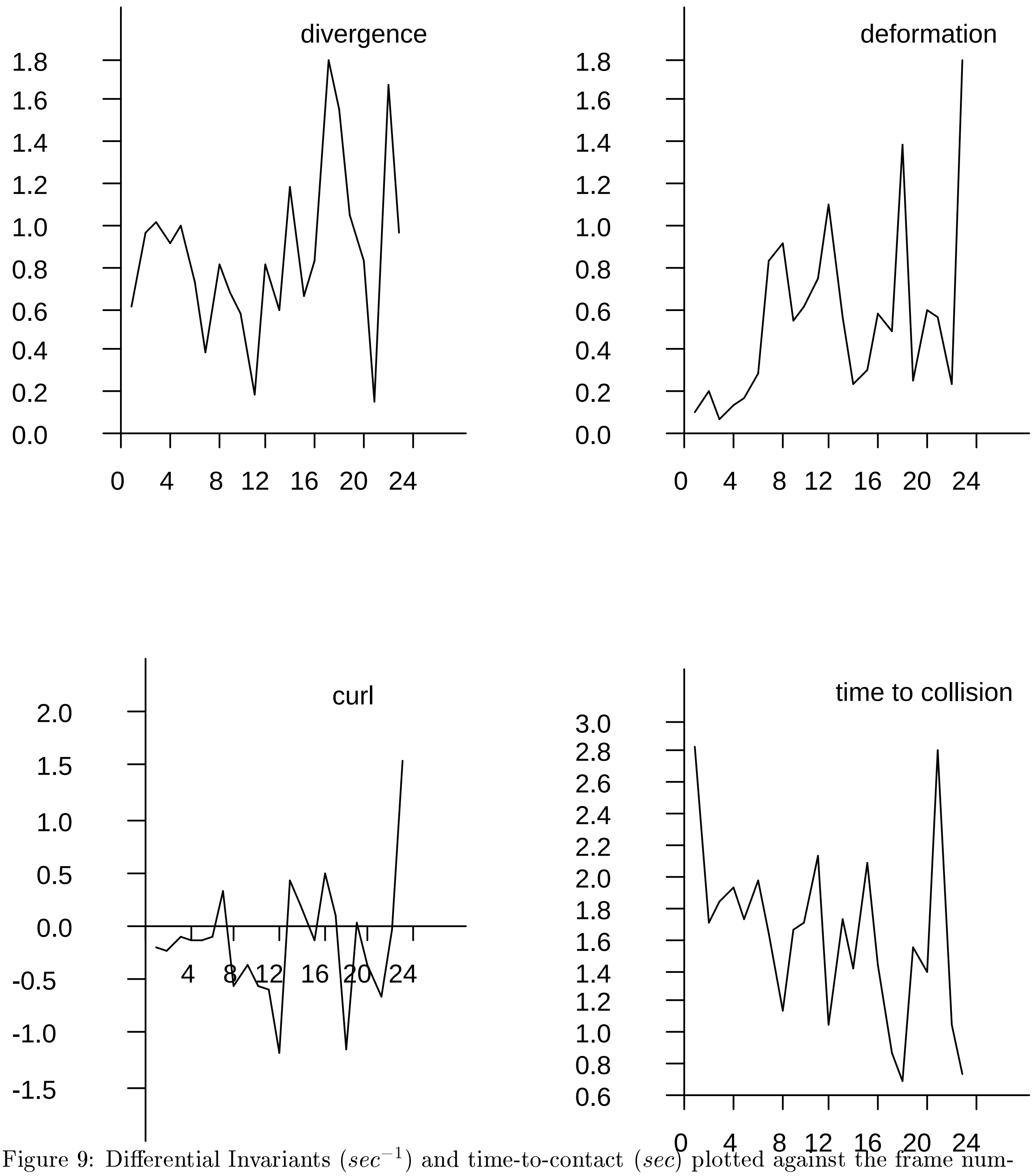

Figure 9: Differential Invariants (sec^{-1}) and time-to-contact (sec) plotted against the frame numbers (Hand)

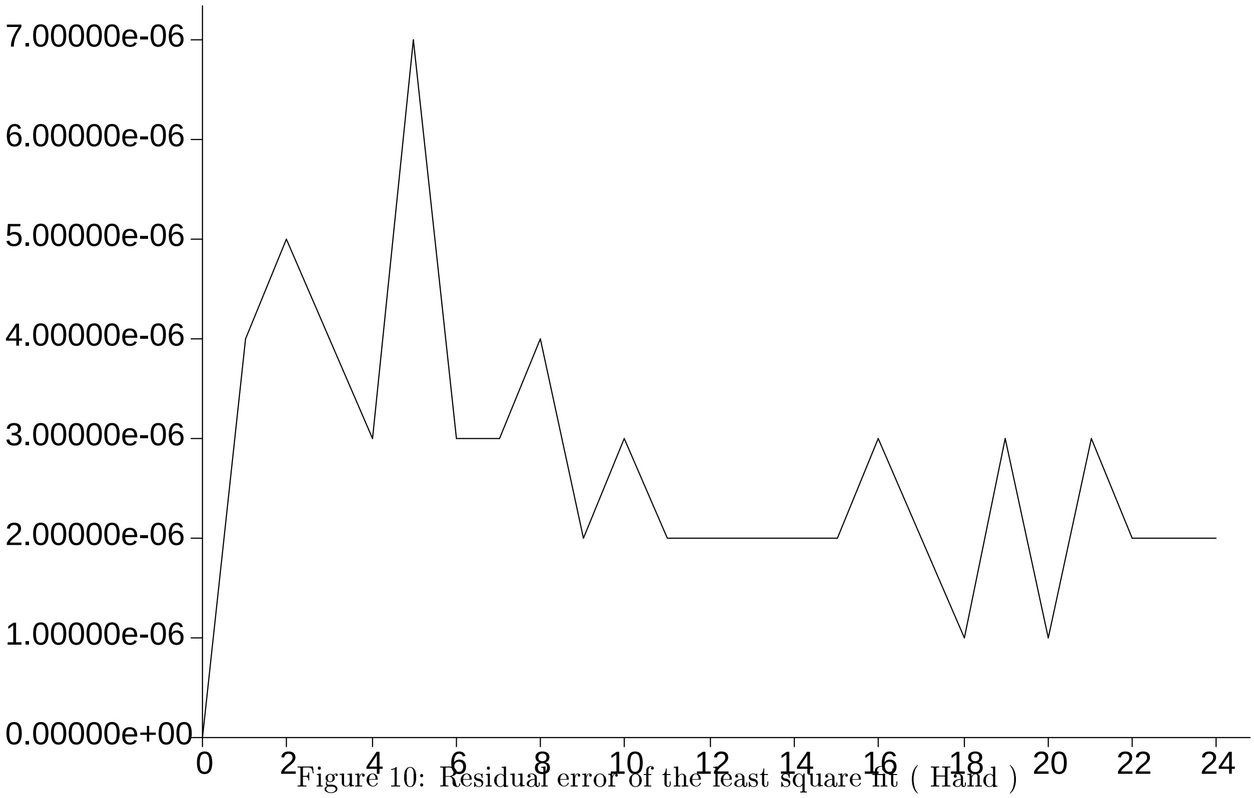

Figure 10: Residual error of the least square fit (Hand)

References

[1] J. Aloimonos and A. Bandopadhyay. *Active Vision.* Proc. 1st ICCV, London, 35-54, 1987.

[2] D.H. Ballard. *Animate Vision.* Artificial Intelligence, 48:57-86, 1991.

[3] D.N.Lee. *The optic flow field: the foundations of vision.* Phil. Trans. R. Soc. London, 290, 1980.

[4] Muralidhara Subba Rao. *Bounds on time-to-collision and rotational component from first-order derivatives of Image Flow.* Computer Vision,Graphics and Image Processing, 50:329-341,1990.

[5] R.C.Nelson and J.Aloimonos. *Using flow field divergence for obstacle avoidance: towards qualitative vision.* Proc. 2nd ICCV, 188-196, 1988.

[6] D.W.Murray, P.McLauchlan, I.D.Reid and P.M.Shakey. *Reactions to Peripheral Image motion using Head/Eye platform.* Proc. 4th ICCV, Berlin, May 1993.

[7] P.McLauchlan, Ian Reid. *2D Vision system for Real-time Gaze Control.* Tech. Report, Dept. of Engineering Science, University of Oxford, 1993.

[8] Subhashis Banerjee, Andrew Zisserman, Philip McLauchlan, Ian Reid and David Murray. *Collision detection using divergent Image Flow.* Department of Engineering Science, University of Oxford, 1993.

[9] A.Mittal and A.Vailaya. *Real Time Collision Detection using Divergent Image Flow.* B.Tech Major Project Thesis, Dept. Computer Science and Engineering, I.I.T. Delhi, 1994.

[10] A.Mittal, A.Vailaya, S.Banerjee and M.Balakrishnan. *Real Time Collision Detection using Divergent Image Flow.* Tech. Report 94/4, Dept. Computer Science and Engineering, I.I.T. Delhi, 1994.

[11] B.K.P.Horn and B.G.Schunck. *Determining Optic Flow.* Artificial Intelligence, 17:185-203, 1981.

[12] J.J.Koenderink. *Optic Flow.* Vision Research, 26(1):161-179, 1986.

[13] J.J.Koenderink and A.J.Van Doorn. *Invariant properties of motion parallax field due to movement of rigid bodies relative to an observer.* Optica Acta, 22(9):773-791, 1975.

[14] R. Aris. *Vectors, Tensors, and the basic equations of fluid mechanics.* Prentice-Hall, Englewood, N.J., 1962.

[15] M. Campani and A.Verri. *Motion analysis from first-order properties of optical flow.* CVGIP: Image Understanding, 56(1):90-107, 1992.

[16] H.C.Longuet-Higgins and K.Prazdny. *The interpretation of a moving retinal image.* Proc. Roy. Soc. London B, 208:385-397, 1980.

[17] B.K.P.Horn. *Robot Vision.* McGraw-Hill, NY, 1986.

[18] A.Verri and T.Poggio. *Against quantitative optic flow.* Proc. 1st ICCV, London, pg171-180, 1987.

[19] A.Verri and T.Poggio. *Motion Field and Optic Flow: qualitative properties.* IEEE Trans. Pattern Anal. Machine Intelligence, PAMI-11:490-498, 1989.

[20] R.C.Nelson. *Qualitative detection of Motion by a moving observer.* Intl. Journal of Computer Vision, 7(1):33-46, 1991.

[21] W. H. Press, S. A. Teukolsky, W. T. Vellering, B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press. 1993.