Order Number 8915583

Morphological instability and pattern selection in solidifying succinonitrile

Chou, Henry, Ph.D.

City University of New York, 1988

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

MORPHOLOGICAL INSTABILITY AND PATTERN SELECTION IN SOLIDIFYING SUCCINONITRILE

by

HENRY CHOU

A dissertation submitted to the Graduate Faculty in Physics in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1988

This manuscript has been read and accepted for the Graduate Faculty in Physics in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

July 29, 1988
Date

Chair of Examining Committee

Aug 3 1988
Date

Executive Officer

Professor J. L. Birman

Professor F. W. Smith

Professor M. Tomkiewicz

Professor H. Levine
Supervisory Committee

The City University of New York

Abstract

MORPHOLOGICAL INSTABILITY AND PATTERN SELECTION IN SOLIDIFYING SUCCINONITRILE

by

Henry Chou

Adviser: Professor Herman Z. Cummins

Morphological instability in the early stage of solidification in succinonitrile has been studied with high-resolution video microscopy. Quantitative analysis of the Fourier spectrum of the evolving crystal-melt interface has demonstrated the crucial role of noise and indicates significant disagreement with predictions of the Mullins-Sekerka theory. In the dendritic stage of solidification, the denrite tip and radius were measured and the stability parameter of Langer, σ^*, was deduced to be 0.032 compared to a value of 0.024 predicted by the microscopic solvability theory of dendritic growth.

Dedicated to my parents, for their faith in an education.

That is the land of lost content,
I see it shining plain,
The happy highways where I went
and cannot come again.

A. E. Housman
A Shropshire Lad

ACKNOWLEDGMENT

I would like to thank Professor Cummins for his patient guidance and tutelage throughout my education here, and for making graduate study not only a most worthwhile but also a most enjoyable experience. I thank Professors Eberly and Bederson for providing invaluable advice which led me to the right decisions at crucial turning points, Professors Birman, Sarachik, Lax, Rubin, Callendar and Smith for the physics and non-physics they taught me in and out of classrooms, Professors Herbert Levine and Olivier Martin for discussions of details of pattern formation theory. I thank all those who, knowingly or unknowingly, have made important comments on our experiment through their generous discussions with me, they include Drs. Ken Jackson, John Weeks, Wim van Saarloos, Andy Dougherty, Boris Shraiman, Pat Cladis, Cliff Surko, Professors Albert Libchaber and Jerry Gollub. I thank my fellow graduate students and colleagues for exchange of useful as well as useless ideas, and for making the Physics Department here so lively. In particular, I thank George Sukenik for teaching me all about computers, Oscar Mesquita for getting me started on my project, Chu-Ling Wang and Boris Yudanin for advice on computer programs, Chester X. W. Qian and Martin Muschol for help and discussion on the experiments, Bob Quinlan for expert mechanical design and machining. Others include Mitra Dutta, Wen Li, Jiang-Ping Lu, Mark Turner, Tar-lok Aurora, Martin Ligare, Miguel Levy, Tracy Turner and Mrs. Francis Tritt. I thank all my friends for good comradeship and for making life in New York so much better, in particular Penny Jones for her steadfast friendship and lessons on arts and culture, Bob Mah the chemistry wizard and Badminton plus tennis partner *par excellence*, King-Soon Ong, Richard and Michelle Lim, Bland and Maria Addison, Frank and Lisa Brown, Peter and Michelle Arumemi, Kris Ishibashi, Stewart Lor, and Haini Wang for her love and concern. Only after I started making this list did I realize how much I am indebted to the people around me. There were undoubtedly others who passed by briefly but made an

indelible difference, I thank them and beg their forgiveness for failing to mention their names.

TABLE OF CONTENT

LIST OF FIGURES

1. INTRODUCTION

1.1 Pattern Formation in Nature

In a recent lecture[1], Victor Weisskopf pointed out that there was no answer to even the simplest of questions one can pose about snowflake growth (Fig. 1a), such as why the six arms are roughly identical in length and why the overall shape of each arm resembles the five others. Nature is full of examples of pattern formation, snowflakes are just a familiar and particularly beautiful example. Figs. 2-7 illustrate some other striking patterns from physical, chemical and biological systems. Fig. 2 shows patterns formed in the so-called Rayleigh-Benard[2,3] cells where a layer of fluid is heated from below. The roll[4] and rectangular[5] patterns arise from different experimental conditions. A good variety of patterns are possible in this system and they are the subject of much current research[6,7]. Another familiar example from the area of fluid flows is the Great Red Spot of Jupiter[8] (Fig. 3), which is known to have been stable for the past ~300 years. Its formation has recently been simulated in a laboratory experiment[9]. Fig. 4 shows spatial patterns formed in the so-called Belousov-Zhabotinskii reaction[8], which is an oscillating chemical reaction involving the oxidation of citric acid by potassium bromate catalyzed by the ceric-cerous ion couple. Fig. 5a shows a pattern formed while pushing air against liquid epoxy in a porous medium[10]. Fig. 5b shows a pattern formed during electrochemical deposition of zinc on an electrode[11]. These highly ramified patterns have been much studied recently for their fractal and "self-similar" properties[12]. Figs. 6 and 7 show pages from the notebooks of the 19th century explorer Ernst Haeckel, of skeletons of some plants (echinoderms in Fig. 6, and Radiolarian in Fig. 7)[13]. Most pattern-forming phenomena in the physical world, from clouds to living creatures, are exceedingly complicated from the physicist's point of view because they occur in thermodynamically open systems and many involve

non-linear interactions. The most studied systems are hydrodynamic flows (particularly Rayleigh-Benard[6,7], Taylor-Couette[14-16] and Saffman-Taylor[17,18]) and, more recently, solidification[19]. These systems involve relatively few parameters which can be experimentally controlled to vary their behavior.

1.2 Pattern Formation in Solidification

Broadly speaking, three types of morphologies are possible during solidification. Firstly, many crystals show macroscopic facets[20-23], quartz being a well-known naturally occuring example. Mostly, faceted growth occurs under well controlled conditions for crystals with sufficient anisotropy and high latent heat of transformation (see Section 2.1). Secondly, the unpredictable, seemingly disordered structure commonly observed in snowflakes is known as dendritic growth and is the usual morphology in most situations of solidification[22]. Fig. 1d shows a single free dendrite of succinonitrile growing into its melt[51]. Thirdly, an ordered structure of cells with well-defined wavelength is possible (Fig. 1c). Cellular growth occurs most frequently in growth of eutectics under the condition of directional solidification[22,24] (see Section 3.3).

Engineers have long recognized the importance of solidification morpholgy on their technologies. On the one hand, metallurgists need to control morphology during casting ("microstructure" is their terminology, Fig. 1b) because it principally determines the limit on mechanical strength of the finished product[22,25,26]. On the other hand, electrical engineers depend on the growth of nearly perfect single crystals of silicon for semiconductors. These two examples encompass perhaps more than half of present-day industry, yet the detailed physical mechanism which determines pattern formation in solidification is not well understood, as Weisskopf pointed out.

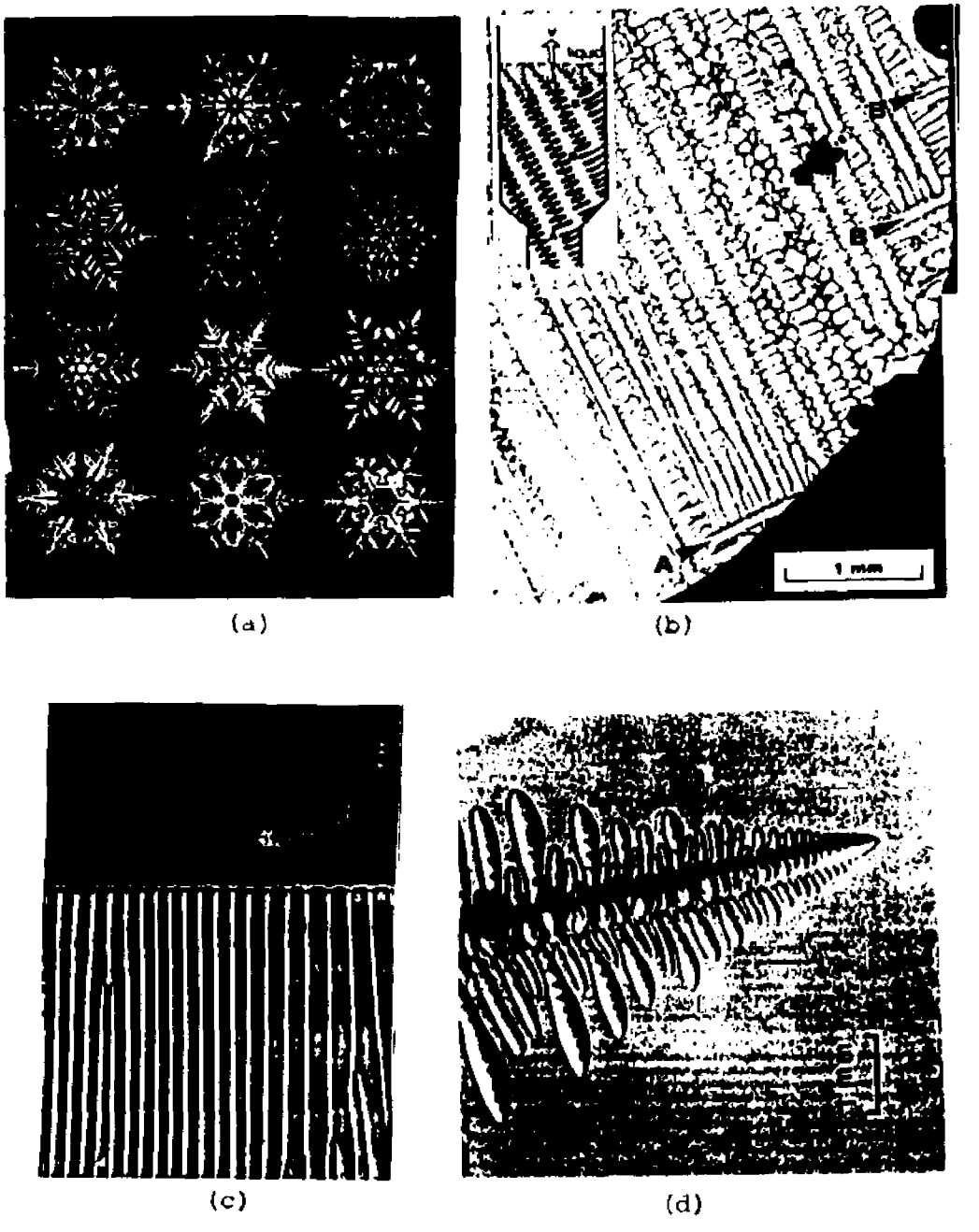


Fig. 1 (a) Snowflakes (U.S. Weather Bureau photographs by W. A. Bentley
(b) Metallographic section of a Ni-Cr-C single crystal (M. Rappaz and E. Blank, J. Cryst. Growth **74**, 67 (1986))
(c) Cellular interface of CBr_4 (Jackson and Hunt[24])
(d) A succinonitrile dendrite (Huang and Glicksman[51])

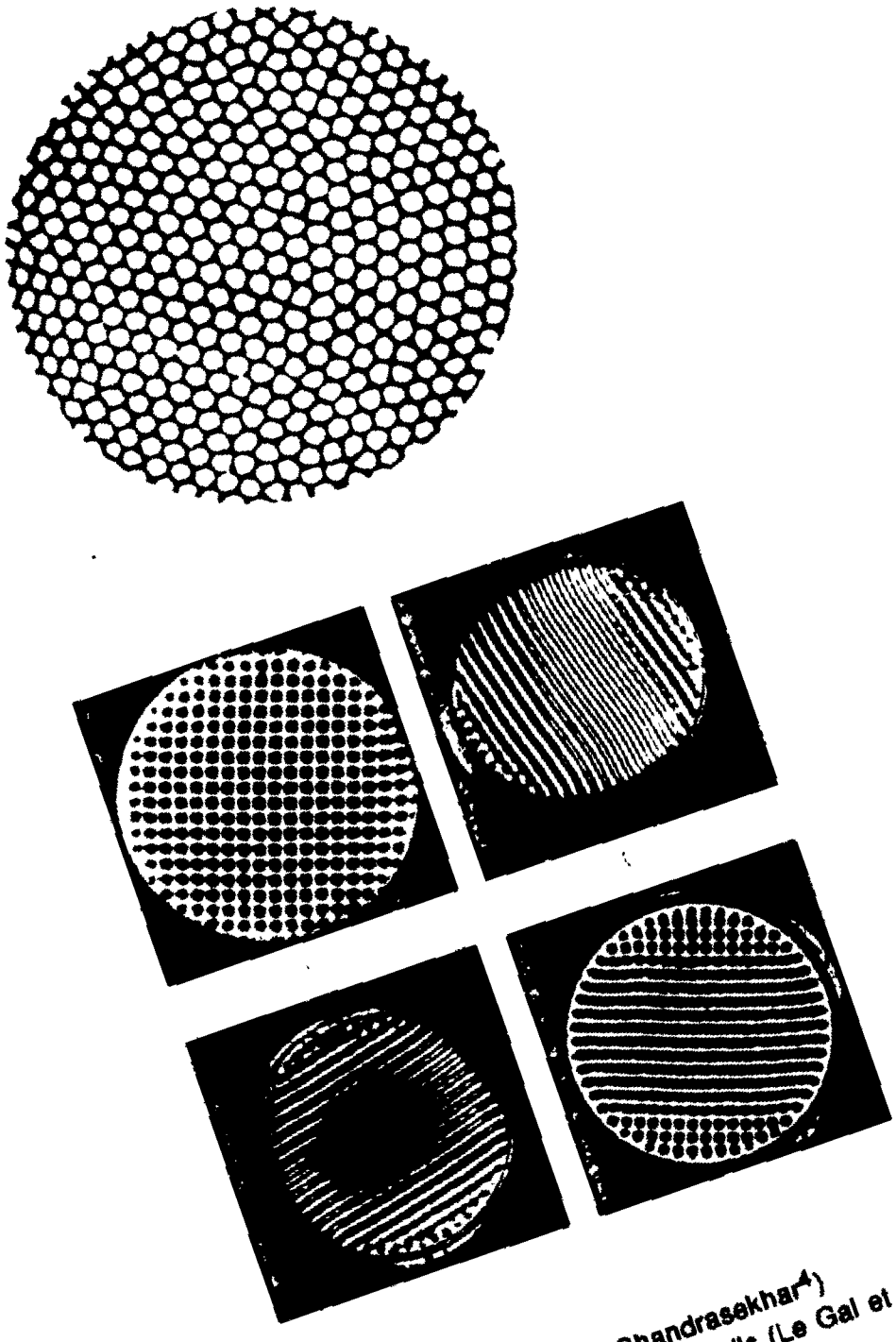


Fig. 2 (a) Rayleigh-Benard rolls (Chandrasekhar[4])
(b) Rayleigh-Benard rectangular cells (Le Gal et al.[5])

Fig. 3 Jupiter's Great Red Spot[8]

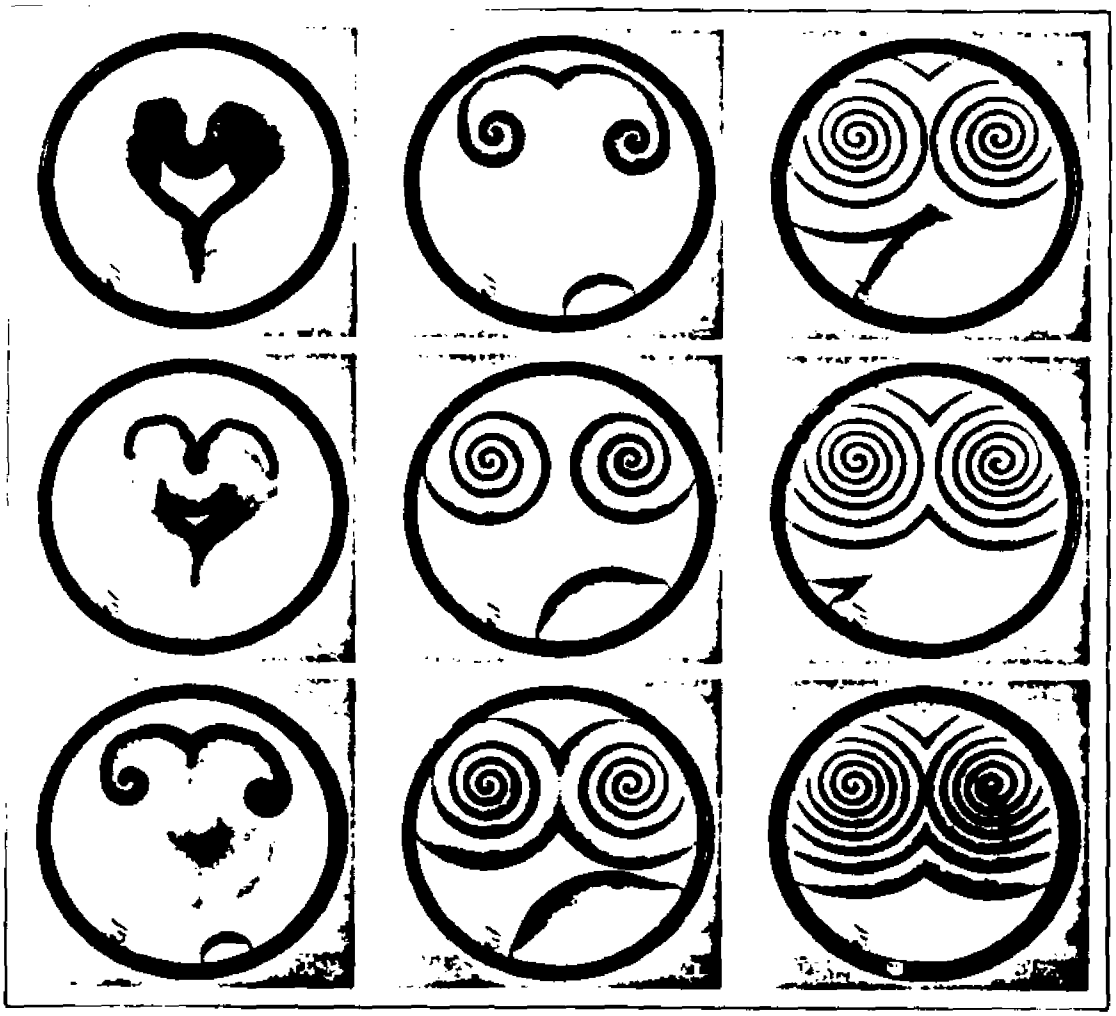
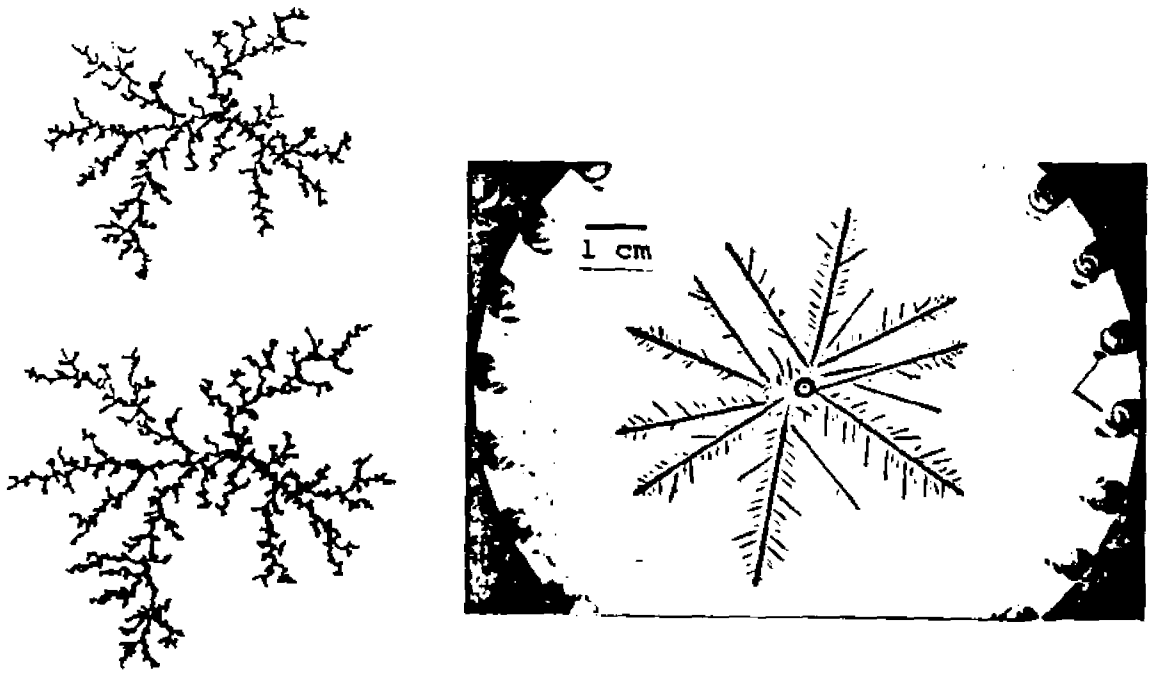


Fig. 4 Belousov-Zhabotinskii reaction[8]



(a) (b)

Fig. 5 (a) Pattern formed by pushing air against epoxy in a porous medium[10]
(b) Pattern formed during electrochemical deposition of zinc [11]

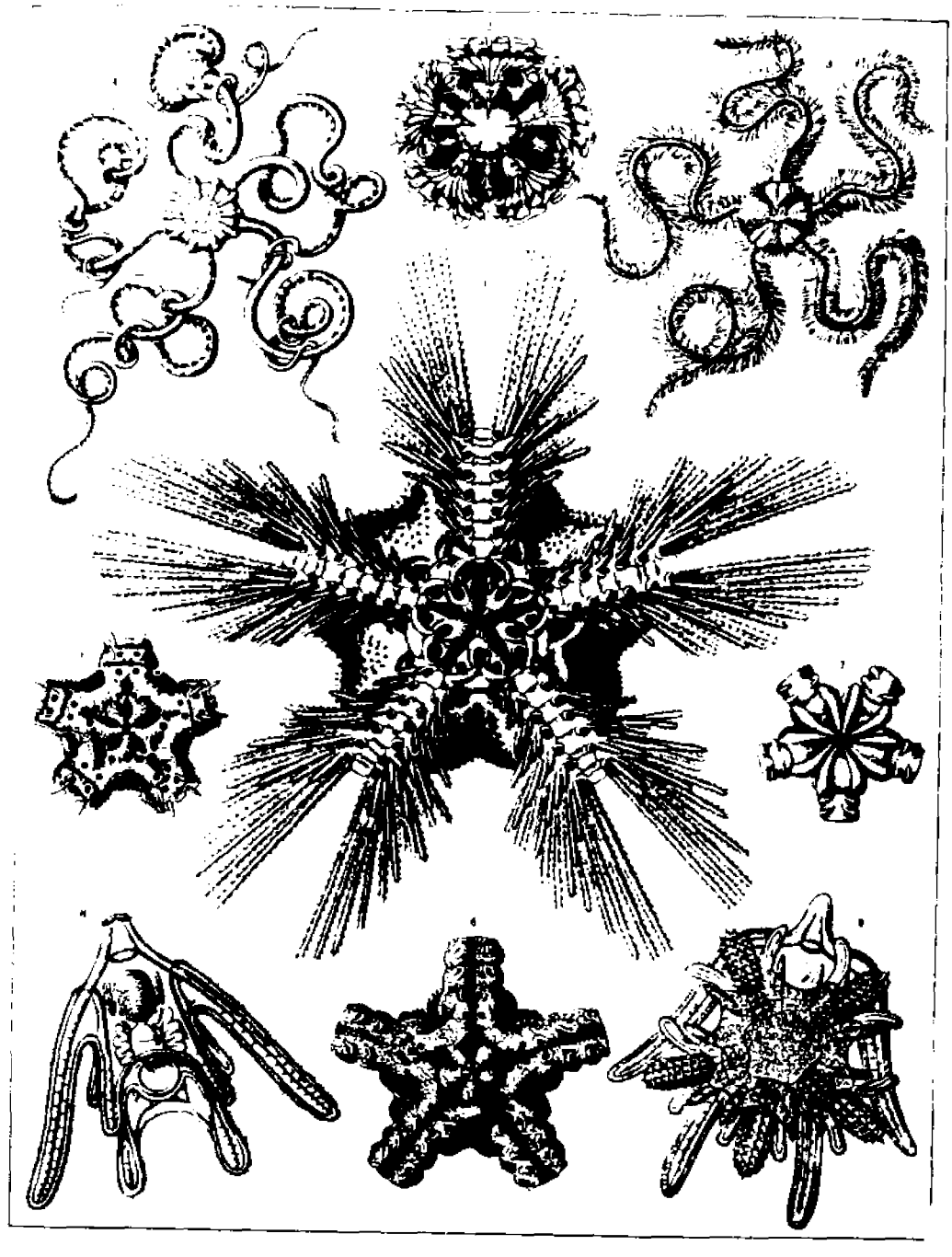


Fig. 6 Skeletons of plants (echinoderms)[13]

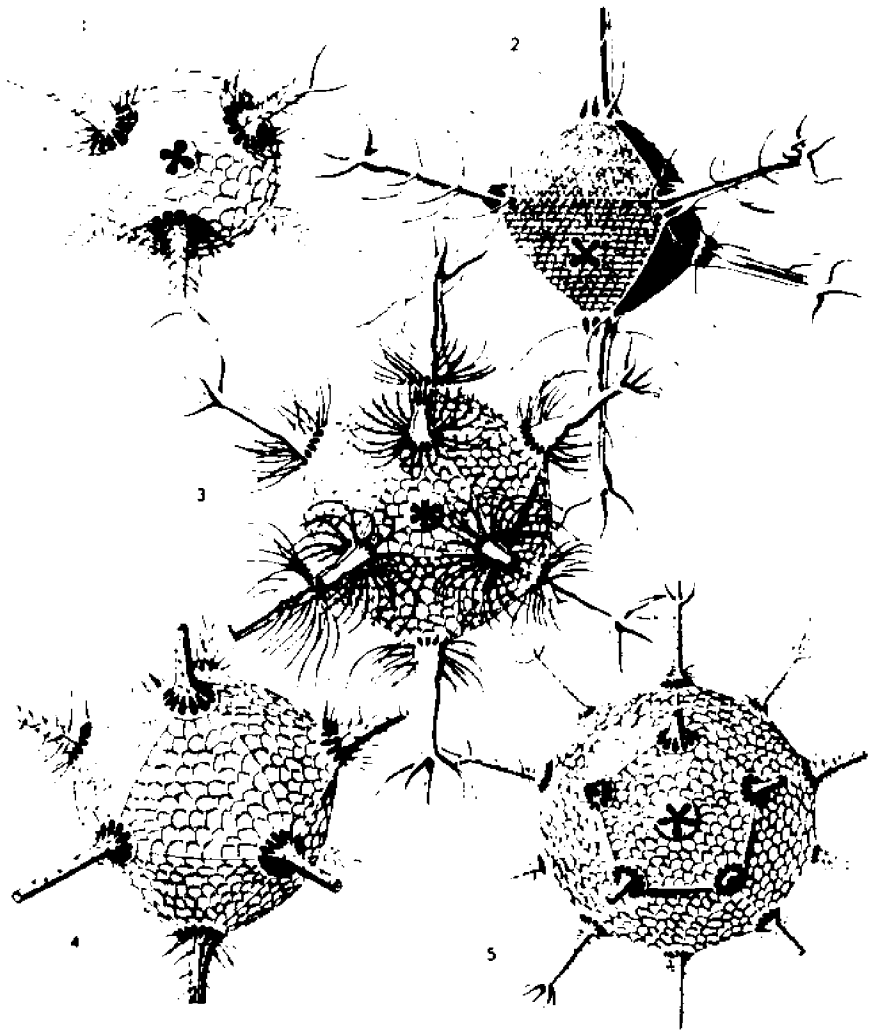


Fig. 7 Skeletons of plants (Radiolarians)[13]

2. THEORETICAL BACKGROUND

2.1 Crystal Growth Fundamentals

The macroscopic morphologies of crystals have long fascinated naturalists and scientists[27-29]. Theories of crystals growth had been proposed even before the atomistic theory of matter were well founded and are still in the process of revision[30,31]. It is now generally accepted that there are mainly three mechanisms for growth: continuous growth (Wilson-Frankel model), nucleated layer growth and growth mediated by defects such as screw dislocations.

The continuous growth model proposed independently by Wilson[32] and Frenkel[33] assumes that all interface sites are active growth sites and that growth is a reversible process with atoms leaving and joining the interface at rates governed by simple activated processes. The theory takes no account of the structure of the interface. Crystals whose growth occurs via this mechanism have no macroscopic facets. Growth is controlled by diffusion of heat and mass and the interface takes the shape of an isotherm of the system. For small undercoolings $\Delta TL/k_B T_m T_i << 1$, growth is predicted to be governed by the kinetic law

$$V_s = \mu_l \Delta T \qquad (2.1)$$

where ΔT = undercooling at the interface = $T_i - T_m$, T_i is the interface temperature, T_m the bulk melting temperature, L the latent heat, k_B the Boltzman constant. The kinetic coefficient μ_1 is predicted by Wilson and Frankel to be

$$\mu_l = \frac{D_M \Delta S}{a k_B T_i} \qquad (2.2)$$

where D_M is the mass diffusion coefficient, ΔS the entropy of melting, *a* the jump distance for diffusion. This kinetic law is valid for materials whose crystal interface is

microscopically "rough" in the sense that growth takes place on all interface sites with no regard to crystalline orientation.

For crystals with macroscopic facets, the relationship between growth rate and undercooling was observed to be different from linear. Volmer [34] and Becker and Doring[35] proposed nucleated layer growth in which the growth rate depends exponentially on undercooling

$$V_g \propto e^{-K/\Delta T} \tag{2.3}$$

where K is a constant. For small undercoolings, the growth rate turns out to be vanishingly small, which is not in accord with experience. This led Frank[36,37,31] to propose defect-mediated growth, such as screw-dislocation growth, which does not require nucleation of a critical nucleus on the interface. In the case of screw dislocation growth from the melt, Hillig and Turnbull[38] worked out the kinetic law to be

$$V_g \propto (\Delta T)^2 \tag{2.4}$$

A detailed treatment of these growth mechanisms can be found in Woodruff[30].

In 1958 Jackson[39-41] developed a remarkably successful theory for the structure of the interface during growth. His theory predicts that, close to thermodynamic equilibrium, the parameter $\alpha = \xi L/k_B T_E$ distinquishes rough ($\alpha < 2$) and faceted ($\alpha > 2$) interfaces, where ξ is the ratio of the number of nearest-neighbors in the plane of the interface to the number of total nearest-neighbors, $L/k_B T_E$ is the ratio of binding energy at the interface to thermal energy, L is the heat of transition (latent heat of evporation in the case of growth from vapor, latent heat of fusion in the case of growth from melt), T_E is the temperature at the transition. From Jackson's α factor, it is possible to predict which material will be faceted or non-faceted. In some materials, it is possible to go from continuous growth ($\alpha < 2$) to faceted growth ($\alpha > 2$) by tuning experimental parameters. This is known as the roughening transition[105]. Some materials, such as NH_4Br[74], seem to be faceted at equilibrium or at low growth rates but become smooth at sufficiently high growth rates. This is known as kinetic

roughening. Most metals growing from their melts have $\alpha < 2$ and grow by the continuous mechanism showing no macroscopic facets. Succinonitrile, with $\alpha \sim 1.4$, is in the category of continuous growth.

2.2 Morphological Instability

In 1963 and 1964, Mullins and Sekerka published two papers[42,43] which laid the grounds for understanding morphological instability of a growing crystal. For a pure substance growing into its undercooled melt, the advancing crystal-melt interface liberates latent heat of fusion. This heat must be conducted away, otherwise the local temperature rises and growth stops. Heat diffusion is more efficient if the interface has a larger temperature gradient ahead of it, namely a bulge is favorable to heat diffusion. A protrusion, however, costs energy because of surface tension. It is the competition between the destablizing (heat diffusion) and the stablizing (surface tension) effects that determines the morphology of the growing solid. For an impure substance (e.g. an alloy), impurities are rejected at the interface from the solid into the liquid, and impurity diffusion plays the decisive role because it is usually a much slower process than heat diffusion. In this case, because of impurity concentration build-up ahead of the interface, the actual temperature of the melt could be lower than the effective solidification temperature. This case is known as constitutional undercooling[22,30,44]. For simplicity, we shall consider only heat diffusion in the pure substance.

The diffusion of the latent heat of fusion from the interface is governed by the diffusion equation, plus two boundary conditions on the moving interface:

Heat diffusion:

Liquid:
$$D_T \nabla^2 T = \frac{\partial T}{\partial t} \tag{2.5}$$

Solid:
$$D_T' \nabla^2 T = \frac{\partial T}{\partial t} \tag{2.6}$$

Heat flux conservation:
$$LV_n = [\, D_T' \, c_p' (\nabla T)_s - D_T c_p (\nabla T)_l \,] \cdot \hat{n} \tag{2.7}$$

Gibbs-Thomson relation:
$$T_m(\kappa) = T_m^0 \left(1 - \frac{\gamma}{L}\kappa\right) \tag{2.8}$$

where D_T and c_p are thermal diffusivity and heat capacity, respectively (primed quantities refer to the solid), L is the latent heat of fusion, V is the velocity of the interface, T^o_m is the bulk melting temperature in the absence of interface curvature, γ is the interfacial tension, κ is the local curvature of the interface. The Gibbs-Thomson relation simply expresses the fact that it costs energy to maintain a curvature at the interface (see Appendix A for a derivation). If one makes the further assumption that the interface is at the equilibrium melting temperature of the curved interface (or, equivalently, the growth kinetics is infinite), then $T(\text{interface}) = T_m(\kappa)$. Eq. (2.8) becomes

$$T(\text{interface}) = T_m^0 \left(1 - \frac{\gamma}{L}\kappa\right)$$

Note that in Eqs. (2.5)-(2.8), the boundary conditions are applied to an interface that is moving, whose position and shape are in fact what one would like to solve. In addition, the curvature term in the Gibbs-Thomson relation makes this set of equations nonlinear. One thus has a set of nonlinear, nonlocal differential equations whose complete solution has defied mathematicians. A linear stability analysis, however, can be easily carried out for certain standard geometries. In the treatment of Mullins and Sekerka, they considered a growing sphere[42] and a plane[43]. In the following, the planar case will be presented in some detail as it is pertinent to our experiment.

Consider a (1-dimensional) flat solid-melt interface advancing at a velocity V into an undercooled melt (Fig. 8a). Following Langer[19], we define the quantities $u = (T -$

$T_m)c_p/L$, l=thermal diffusion length = $2D/V$, d_0= capillary length = $\gamma T_m c_p/L^2$. In the reference frame moving with the interface, the steady state equations become

Liquid:
$$\nabla^2 u + \frac{2}{l}\frac{\partial u}{\partial z} = 0 \tag{2.9}$$

Solid:
$$\nabla^2 u' + \frac{2}{l}\frac{\partial u'}{\partial z} = 0 \tag{2.10}$$

Heat flux Conservation:
$$V_n = D\left[\frac{D' c_p}{D c_p}(\nabla u')_s - (\nabla u)_l\right]\cdot \hat{n} \tag{2.11}$$

Gibbs-Thomson condition:
$$u(\text{interface}) = -d_0 \kappa \tag{2.12}$$

The steady-state temperature field in the liquid is

$$T(z) = T_m - \frac{L}{c_p}\left[1 - e^{-\frac{V}{D_T} z}\right] \tag{2.13}$$

Note that the only allowed undercooling at infinity is L/c_p. This is a consequence of heat flux conservation and the steady-state condition. Now if we impose a sinusoidal perturbation on the flat interface of the form

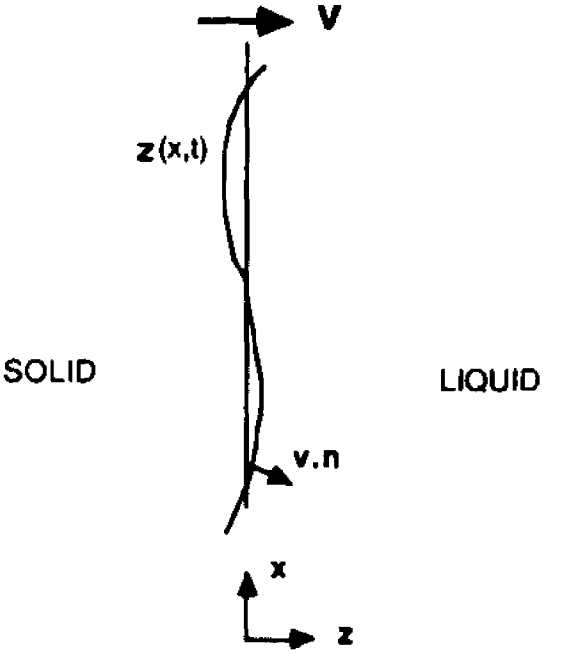
$$\zeta(x, t) = \hat{\zeta}_k e^{ikx + \omega t} \tag{2.14}$$

and the temperature field ahead of the interface is deformed correspondingly as

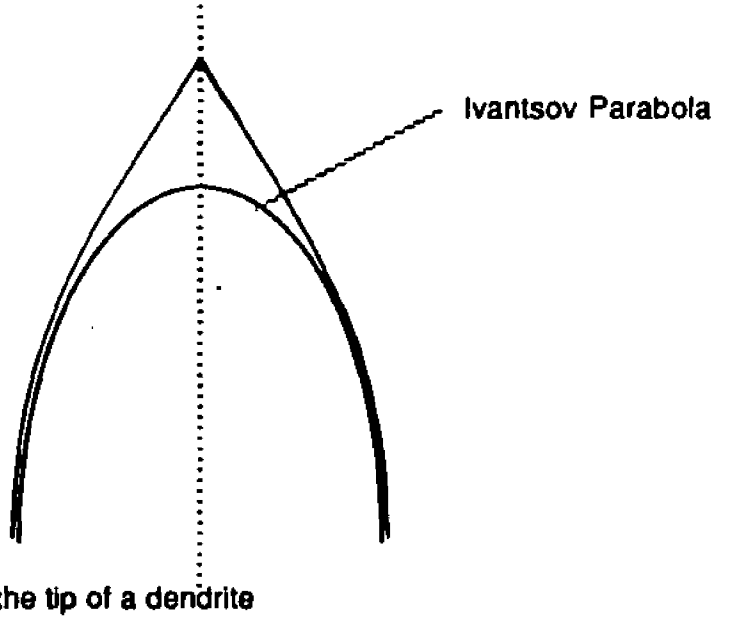
Liquid:
$$u(x, z, t) = u^0(z) + \hat{u}_k e^{-qz} e^{ikx + \omega t} \tag{2.15}$$

Solid:
$$u'(x, z, t) = \hat{u}'_k e^{q'z} e^{ikx + \omega t} \tag{2.16}$$

where $u^0(x,z,t)$ is the steady state temperature field as given in Eq. (2.13). Putting these expressions into Eqs. (2.9) and (2.10), we get



(a) Coordinates for the Mullins-Sekerka linear stability analysis of a planar interface



(b) Mismatch at the tip of a dendrite

Fig. 8

$$-k^2 - \frac{2}{l}q + q^2 = 0 \qquad (2.17a)$$

$$-k^2 + \frac{2}{l}q' + q'^2 = 0 \qquad (2.17b)$$

In Eqs. (2.15) and (2.16), $\hat{\zeta}_k$, $\hat{A}_k$ and $\hat{A}'_k$ are small amplitudes. In a linear stability analysis, we shall only keep terms to first order in these terms. Next we apply the Gibbs-Thomson condition $u(\text{interface}) = -d_0\kappa$.

Liquid: $u(\text{interface}) = u(x, z = \zeta, t)$

$$= [e^{-\frac{2}{l}(\hat{\zeta}_k e^{ikx+\omega t})} - 1] + \hat{A}_k e^{-q(\hat{\zeta}_k e^{ikx+\omega t})} e^{ikx+\omega t} \qquad (2.18)$$

Solid: $$u'(\text{interface}) = \hat{A}'_k e^{q'(\hat{\zeta}_k e^{ikx+\omega t})} e^{ikx+\omega t} \qquad (2.19)$$

Now expanding the exponents and keeping terms linear in $\hat{\zeta}_k$, $\hat{A}_k$ and $\hat{A}'_k$, we get

$$d_0\kappa = (\frac{2}{l}\hat{\zeta}_k - \hat{A}_k)e^{ikx+\omega t} = \hat{A}'_k e^{ikx+\omega t} \qquad (2.20)$$

The curvature expression in two-dimensional Cartesian coordinates is

$$\kappa = \frac{-\frac{\partial^2\zeta}{\partial x^2}}{\left[1 + \left(\frac{\partial\zeta}{\partial x}\right)^2\right]^{3/2}} \cong -\frac{\partial^2\zeta}{\partial x^2} \qquad (2.21)$$

where the approximation is good for small curvatures. When evaluated for the interface $\zeta(x,t)$ as written in Eq. (2.14) and the result linearized, we get

$$\kappa = \hat{\zeta}_k k^2 e^{ikx+\omega t}$$

Putting this into Eq. (2.20), we have

$$d_0 \hat{\zeta}_k k^2 = \frac{2}{l} \hat{\zeta}_k - \hat{u}_k = \hat{u}'_k \tag{2.22}$$

Finally we need to satisfy flux conservation. On the left-hand side of Eq. (2.11), the velocity is

$$V \equiv V_0 + \frac{d\zeta}{dt} = V_0 + \omega_k \hat{\zeta}_k e^{ikx + \omega t} \tag{2.23}$$

On the right-hand side, we have

$$D\left[\beta \left(\frac{\partial u}{\partial z} \right)_a - \left(\frac{\partial u}{\partial z} \right)_l \right]$$

$$= D\left[\beta \left(\hat{u}'_k \, q' e^{q' z} e^{ikx + \omega t} \right) - \left(-\frac{2}{l} e^{-\frac{2z}{l}} - q\hat{u}_k e^{-qz} e^{ikx + \omega t} \right) \right]$$

where $\beta = D'c_p'/D\,c_p$. Again evaluating at the interface $z = \zeta = \zeta_k e^{ikx+\omega t}$ and linearizing, we get

$$= \left[\beta \hat{u}'_k \, q' + \frac{2}{l} - \left(\frac{4}{l^2} \hat{\zeta}_k - q\hat{u}_k \right) \right] e^{ikx + \omega t} \tag{2.24}$$

Equating this with Eq. (2.23), we have

$$\omega_k \hat{\zeta}_k = D\beta \hat{u}'_k \, q' - \frac{4D}{l^2} \hat{\zeta}_k + Dq\hat{u}_k \tag{2.25}$$

Now putting Eqs. (2.22) into Eq. (2.25)

$$\begin{aligned} \omega_k \hat{\zeta}_k &= D\beta q' \left(-d_0 k^2 \hat{\zeta}_k \right) - \frac{2\nu}{l} \hat{\zeta}_k + Dq \left(\frac{2}{l} - d_0 k^2 \right) \hat{\zeta}_k \\ &= -Dd_0 k^2 \left(\beta q' + q \right) + V \left(q - \frac{2}{l} \right) \end{aligned} \tag{2.26}$$

From Eqs. (2.17), we have $q \sim q' \sim k$ if $kl >> 1$, then we finally get

$$\omega_k \equiv Vk \left[1 - \tfrac{1}{2} \left(1 + \beta \right) d_0 l k^2 \right] \tag{2.27}$$

If we assume that the solid and liquid have identical thermal properties (symmetric model), then $\beta\ D'c_p'/D\ c_p = 1$, and we have simply

$$\omega_k \equiv Vk\ (1 - d_0 lk^2\ ) \tag{2.28}$$

which is the relation plotted in Fig. 15a for succinonitrile. Note that the cut-off k-value at which $\omega_k=0$ is $k_c=(d_0 l)^{-1/2}$, and the k-value at which growth rate is maximum is $k_{max}=(3d_0 l)^{-1/2}$. The wavelength associated with the maximum growth rate, $\lambda=2\pi/k_{max}=2\pi(3d_0 l)^{1/2}$, is essentially the geometric mean of the two length scales in the problem (thermal diffusion length and capillary length) and is somtimes referred to as the Mullins-Sekerka length.

We have presented here the simplest version of the Mullins-Sekerka linear stability analysis. A few words about the implicit assumptions are in order. Firstly, this theory is appropriate for a rough interface (Jackson's α factor less than 2) growing by the continuous growth mechanism. Mullins and Sekerka took no account of growth kinetics at the interface (or, equivalently, the kinetic coefficient is assumed infinite). The growth is assumed to be diffusion-controlled. Secondly, the Gibbs-Thomson relation is an equilibrium condition on the interface. Strictly, the interface cannot be at equilibrium because it is growing into an undercooled melt. Thirdly, there is no account for the possibility that there is mass transport at the interface. The densities of the solid and liquid are asumed to be equal. There have been various improvements[45-47] to account for growth kinetics, convection at the interface etc. but they will not be reviewed here.

2.3 Theory of Dendritic Growth

Much of the recent theoretical effort on pattern formation in solidification has focused on the problem of velocity and tip radius selection. Kessler et al.[48] and Langer[49] have reviewed our present understanding of the dendritic problem.

Experimentally it has been well established that growth rates and tip radii for a given material are reproduciably determined by the undercooling[50,51]. The long-standing theoretical problem was to work out a selection mechanism from the diffusion and boundary equations governing the growth.

In 1947 Ivantsov[52,19] solved the heat diffusion problem by completely neglecting surface tension and found parabolas as steady dendrite shapes. In the absence of surface tension, the tip radius ρ^* and velocity V^* are related to the undercooling by the relation

$$\Delta = 2\sqrt{p}\, e^{p} \int_{\sqrt{p}}^{\infty} e^{-y^2}\, dy = (\pi p)^{1/2} \tag{2.29}$$

where Δ is the undercooling $p = V^*\rho^*/2D$ is known as the Peclet number. The approximation is good for small undercoolings and $p << 1$. This relationship, however, indicates that the product of V^* and ρ^* is determined by the undercooling, rather than each being individually selected.

There have been several attempts at incorporating the surface tension into the dendrite problem. The simplest one is the spherical approximation[19] which treats the tip of the Ivantsov parabola as a sphere and assumes the growth rate of the spherical solid (which can be solved exactely). In the "modified Ivantsov" approximation[19], the curvature is taken as a constant $2/\rho$. The Temkin method[19] uses the full expression for the curvature but requires that the heat flux conservation be satisfied only at the tip. In 1978, Langer and Muller-Krumbhaar[53,54] reasoned that, since the length scale related to the surface tension (capillary length $d_o=\gamma c_p T_m/L^2$ - 0.01 μm) was orders of magnitude smaller than the thermal diffusion length $l=2D/V$ - 10,000 μm, it was perhaps valid to include surface tension in the problem as a small perturbation. They made a stability analysis on the Ivantsov parabola with the surface tension as a perturbation and came to the conclusion that $V^*\rho^{*2}$=constant, i.e. still no individual

selection of tip radius and velocity. They then proposed the marginal stability hypothesis[53,19] that "the selected operating point is the solution that is marginally stable". The predicted value of the "stability parameter", $\sigma^* = 2Dd_0/V^*\rho^{*2} \sim 0.025$ agreed rather well with Glicksman's experimental value of 0.0195 for succinonitrile[50,51].

Local models were then studied to describe the development of the interface. Ben-Jacob et al.[55-57] proposed the boundary layer model. Brower et al.[58-61] proposed the geometric model. Solutions to these models were sought numerically. In the course of these investigations, they came to the important realization that surface tension could not be treated as a perturbation as Langer and Muller-Krumbhaar[53,54] had originally done. The reason is that surface tension constitute a singular perturbation in the sense that

$$\left(\frac{dy}{dx}\right)_{tip} \sim e^{-1/\gamma} \tag{2.30}$$

where $(dy/dx)_{tip}$ is a discontinuity as one approaches from the left and the right of the tip (Fig. 8b), for a dendrite shape derived from the equations with surface tension γ included. A perturbative expansion for small γ thus invites singularities.

A new paradigm, borrowed from van den Broeck's approach to the closely related Saffman-Taylor problem[62,18], is now known as "microscopic solvabilty" . The starting point is to write the solution to the full set of equtions (including surface tension) as an integral representation. Far away from the tip, curvature is small and surface tension unimportant, therefore the Ivantsov solution is accurate. The idea is to start with the Ivantsov solution in the tail region, and relax the integral equation at the tip to allow for a possible cusp close to the tip. With this freedom, solutions exist for all values of V for any given undercooling Δ. Then one looks for solutions which have vanishing cusp magnitude (the discontinuity in dy/dx as one approaches from the left and from the right

of the tip) to determine the actual selected V^*. With this procedure, only a (countably infinite) discrete set of solutions survive, thus resolving the long-standing paradox of the continuous family of solutions. Kessler et al.[63-65] and Bensimon et al.[66] then made stability analyses on the discrete set and concluded that only one solution, the one with the highest velocity, was stable. This was predicted to be the selected velocity for the given undercooling. An unexpected feature of the theory is that finite-velocity solutions can be found only if a non-zero anisotropy was built into the system, such as surface tension $\gamma=\gamma_0[1 - \varepsilon\cos(n\theta)]$. Subsequently, Ben Amar et al.[106] and Barbieri et al.[68], following the work of Shraiman[67] on the Saffman-Taylor problem, analytically derived the scaling relationship (for the fully non-local two-dimensional dendrite problem in small undercoolings)

$$\sigma^* \sim \sigma_0 \varepsilon^{7/4} \tag{2.31}$$

valid for small values ε of surface tension anisotropy, where $\sigma^*=2Dd_0/V^*\rho^{*2}$ and σ_0 is a constant of order 1. Most of the results summarized above apply to the two-dimensional case. There have been extensions to the three-dimensional case and they have been reviewed by Kessler et al.[48]

3. EARLIER EXPERIMENTAL WORK

Twenty five years have passed since the publication of Mullins and Sekerka's papers but there has been but one experiment performed to check the predictions made therein. The paucity of direct experimental check may be due to the difficulty that the Mullins-Sekerka theory assumes an initially planar interface growing into a uniformly undercooled melt, which is an intrinsically unstable and therefore experimentally unattainable situation. There have been, however, many experiments on dendritic and cellular instabilities.

4.1 The Initial Instability

Hardy and Coriell[80] made a Mullins-Sekerka type of stability analysis on an ice cylinder. They then made an experimental observation of instability wavelength and growth rate on this geometry and concluded there was agreement between theory and experiment. The comparison of their experimental result with theory, however, was hampered by the lack of a reliable value of the interfacial surface tension γ and was flawed in logic. Trivedi et al.[69] took still photographs of the initial instability on a (three-dimensional) interface of succinonitrile containing known amounts of acetone as an impurity but made no quantitative analysis on the initial instabiltiy.

4.2 Dendritic Growth

Experiments on dendritic growth have been made on succinonitrile[50,51,69-72], NH_4Br[73-75], NH_4Cl[76], cyclohexanol[77], Krypton[78] and methyl sulfoxide[79]. In the following, the most significant findings in these experiments will be summarized.

(1) Succinonitrile: The Jackson α-factor of succinonitrile is 1.4 and it grows by the continuous growth mechanism like many metals. In addition, it has a convenient melting temperature of 58 °C and is optically transparent in the visible. Succinonitrile is by far the most studied material in dendritic growth with its physical parameters measured to good precision. The seminal work quoted by all is the 1976 paper of Glicksman, Schaefer and Ayers[50] in which they studied (3-demensional) free dendrites growing into an undercooled bath of melt. Samples of purities ranging from 99% to 99.9995% were used. They measured growth speed and tip radius as functions of undercoolings ranging from 0.5 °C to 9 °C. From their data they deduced that the dendrite velocity was well described by the scaling relationship

$$V \equiv 0.02 (\Delta\theta)^{2.6}$$

where v is measured in cm/s and $\Delta\theta$ is the normalized undercooling measured in units of L/C_p=23.1 K. The tip radius of the dendrites scaled with the undercooling as

$$\rho \equiv 8 \times 10^{-3} (\Delta\theta)^{-1.1}$$

where ρ is measured in cm. Although they pointed out that there were problems of achieving steady state tip shapes, their data certainly suggests that well-defined tip radius and velocity were selected for a given undercooling.

Subsequently, Huang and Glicksman[51] made another study of the same system at smaller undercoolings (0.05 °C to 1 °C) specifically to check the marginal stability hypothesis of Langer and Muller-Krumbhaar[53,54]. The scaling behaviors of velocity and tip radius in this range of undercoolings deviate slightly from those written above, but they concluded that $\rho^2 V \sim$ constant and the stability parameter $\sigma^* = 2Dd_0/V\rho^2$ was 0.0195, in good agreement with the value of 0.025 predicted by theory. In addition, they measured side-branch spacing λ_2 as a function of undercooling and found $\lambda_2/\rho \sim 3.0$, in fair agreement with predictions of 2.1 by the marginal stability theory.

More recently, Trivedi et al.[69,70] studied directional solidification in succinonitrile/acetone system. They measured tip radius, primary dendrite spacing and side-

branch spacing as functions of composition, temperature gradient and growth velocity. Their results were supportive of the marginal stability hypothesis for they found that (i) $\rho^2 V$ ~ constant ~ 441 ± 30 $\mu m^3/s$ compared to the predicted value of 411 $\mu m^3/s$, and (ii) λ_2/ρ ~ 2.2 ± 0.03 compared to the predicted value of 2.1. Esaka and Kurz[71] performed essentially the same experiments as Trivedi and again found λ_2/ρ ~ 2.09 ± 0.22.

In an entirely different vein, Honjo, Ohta and Matsushita[72] recently studied dendritic morphologies of succinonitrile in cells whose surfaces have been randomly roughened to various degrees to emulate a variable "anisotropy" in the system. They did this to qualitatively check some length scale arguments made in the microscopic solvability theory, in which anisotropy plays a crucial role. With the undercooling as an additional control parameter, they observed various morphologies which they called stable parabolic, tip-oscillating, symmetric tip-splitting and asymmetric tip-splitting.

(2) NH_4Br: The first experimental study was by Honjo and Sawada[75] on aqueous solution of NH_4Br growing in a capillary at undercoolings ranging from 1 to 10 °C. They found

$$V = (1.99 \pm 0.93)\Delta T^{2.10 \pm 0.29} \ \mu m/s$$
$$\rho = (5.15 \pm 0.58)\Delta T^{-0.57 \pm 0.25} \ \mu m$$
$$\lambda_2 = (23.39 \pm 5.01)\Delta T^{-0.50 \pm 0.11} \ \mu m$$

where $\Delta T = T_s - T_o$, with T_s = saturation temperature = 67.5 °C and T_o is the temperature far away from the interface. From the first two equations, they obtained $V\rho^2$ ~ 25 $\mu m/s$ as compared to a value of 80 $\mu m/s$ predicted for this system by the marginal stability theory. From the last two equations, they deduced λ_2/ρ ~ 4.68 as compared to a value of 2 predicted by theory.

Dougherty, Kaplan and Gollub[73] made a careful study of side-branching activity of NH_4Br dendrites grown from aqueous solution in a capillary. Their findings are (i) side-branching is only weakly periodic, (ii) side-branching activities on the right and

left of the tip are poorly correlated with each other, and (iii) within their instrumental resolution, there is no clear onset distance (measured from the tip) at which side-branching begins. In fact, the side-branch amplitudes measured as a function of distance from the tip seems to suggest that the amplitude can be extrapolated back to some finite value (or zero) all the way to the tip. Based on these observations, they concluded that the origin of side-branching was probably noise. This possibility had in fact been suggested by Pieters and Langer[107,108] based on some numerical work within the boundary-layer model[55-57].

Dougherty and Gollub[74] then measured the surface tension anisotropy ε for NH_4Br, and tip-radius as a function of dendrite velocity, from which they deduced that the stability parameter $\sigma^* = 0.081 \pm 0.02$ as compared to a value of 0.065 ± 0.02 predicted by microscopic solvability.

(3) NH_4Cl: Honjo, Ohta and Sawada[76] studied dendrites growing from aqueous solutions at different undercoolings in a thin (5 μm) cell. They observed three different types of morphologies: tip-stable, tip-oscillating, and tip-splitting. For the tip-stable type, they found that the tip velocity varies linearly with tip curvature ρ, i.e. $V \sim 1/\rho$, in disagreement with marginal stability theory which predicts $V \sim 1/\rho^2$.

(4) Cyclohexanol: Okamoto and Kishitake[77] studied dendrite spacing and side-branch spacing as a function of cooling rate, temperature gradient and growth velocity. Their data, while containing a wealth of empirical information, is not in a form that can be compared with known theories of dendritic growth.

(5) Krypton: Bilgram, Firman and Kanzig[78] measured growth velocity and tip-radius as a function of undercooling in the range $0.005\ K < \Delta T < 0.25\ K$. They found that

$$V = 3.1 \times 10^{-2} (\Delta T)^{1.73}\ cm/s$$
$$\rho = 6.7 \times 10^{-4} (\Delta T)^{-0.68}\ cm$$

which is consistent with $V\rho - \Delta T$, rather than $V\rho^2$=constant asserted by the marginal stability hypothesis. Their measurements, however, has come under the criticism that the thermal diffusuion length in their system is comparable to the size of the apparatus and that convective effects are probably important in their experiment.

(6) Dimethyl sulfoxide: Fattinger, Honegger and Lukosz[79] studied dendritic growth of this material in thin samples (thicknesses ranging from 0.6 to 10 μm). They found that for samples thinner than ~ 2 μm, the usual dendritic morphology breaks down due to wetting effects with the glass cell. With fluorescent techniques, they established the wetting layer to be ~ 50-80 nm in such cases.

3.3 Other Experiments on Solidification Morphologies

Cellular morphology is an area of much current research. It has been known for some time that cellular structure was a steady state solution of the diffusion equation with moving boundary conditions, Eqs. (2.9)-(2.12)[24,100,101]. The next question is whether there is a sharp wavelength selection analogous to the dendrite tip radius. Karma[102], based on his numerical study in the framework of microscopic solvability, claimed that there was selection. Some doubts, however, have been raised by Ben-Amar et al.[103] , based on their new numerical results. It is now believed by theorists that steady-state solutions exist for a band of wavelengths but there is no selection of a single wavelength for a given growth speed. Suzuki et al.[81] studied onset and evolution of instabilities at the solid-melt interface of tin undergoing directional solidification, as a function of growth velocity, temperature gradients and impurity content. By employing ultrasound techniques, they observed onset of instabilities and its passage to "pseudo-steady state" of cellular growth. Their technique, however, does not seem to allow them to answer the question of wavelength selection in such situations.

de Cheveigne et al.[82,83] studied cellular interfaces of impure CBr_4 undergoing directional solidification in thin sample cells of 5-150 μm thicknesses. They found that above a critical pulling speed, the initially flat interface becomes cellular and the bifurcation is subcritical. Above the critical speed, the cellular wavelength varies approximately as $V^{-1/2}$. In addition, they found that the critical speed was strongly dependent on the thickness of the sample and thus concluded that the problem could not be considered two-dimensional. The selected wavelength for a given temperature gradient and pulling velocity (above the critical) was found to be independent of cell thickness.

Heslot and Libchaber[84] studied qualitativly the role of crystalline orientation (hence anisotropy) in directional solidification of succinonitrile. The instabilities at the tips of the cells are different for cells growing along differing crystalline orientations. If cells are aligned to a crystalline axis, then the tips are stable, otherwise the tips undergo either oscillating or splitting instabilities. They therefore concluded that anisotropy was important in understanding evolution in the nonlinear regime. Bechhoefer and Libchaber[85] studied directional solidification in pivalic acid, which reportedly has large surface tension anisotropy (5% as compared to 0.7% of succinonitrile, although no publication has shown this measurement in detail). They found that there was a well-defined wavelength for a given pulling velocity. This, they claimed, was in agreement with microscopic solvability. Ben-Amar et al.[103] recently pointed out that an analysis within the frame-work of microscopic solvability indicated that there was no selection in the directional solidification case. There are some questions as to whether Bechhoefer and Libchaber indeed observed steady-state or some slow transient state of the system. Oswald, Bechhoefer and Libchaber[86], and Bechhoefer et al.[87] have also studied directional solidification in liquid crystals 8CB and 4O8.

Finally, there are quite a few theoretical as well as experimental studies of the morphology of eutectic growth because of its technological importance. As we are not

really concerned with this area, they will not be reviewed here. For more details, see Chapter 5 of Kurz and Fisher[22] and the references therein.

4. EXPERIMENTAL APPARATUS

The overall set-up of the apparatus used in the experiments is schematically shown in Fig. 9. The crystal-melt interface is observed under a microscope (section 4.1) through a video camera whose output is routed to either a VCR for recording or to a set of image digitization/acquisition boards (section 4.3). The image aquisition hardware is interfaced to a PDP-11/73 computer (section 4.4) for control and storage of data.

4.1 Microscope

The microscope is a Nikon Diaphot inverted microscope with Nomarski differential interference (DIC) optics. The principle of Nomarski differential interference is explained by Lang[88]. The Nomarski optics highlights regions of refractive index gradient in the specimen. Since the solid and melt have slightly different refractive index, the interface shows up prominently under Nomarski optics. The inverted construction of the microscope has the advantage of mechanical stability which is useful for making peripheral attachments such as a camera.

DIC objectives of 10x and 20x power were utilized. Working distances are 9.2 mm for 10x and 0.45-2.16 mm (adjustable) for 20x. Because of the short working distance for the 20x objective, the designs of the temperature stage and sample holder are severely constrained (see section 4.2). The eyepieces are 10x magnification, thus producing a total magnification of 100x or 200x. When viewed through the eyepiece, the field of view is about 200 μm for the 10x objective.

The microscope is equipped with side ports for both video and still cameras. The camera in use is a Dage-MTI video camera with a Nuvicon tube. A magnifying projection lens can be inserted before the camera. Projection lenses of 4.5x, 2.5x and 1x

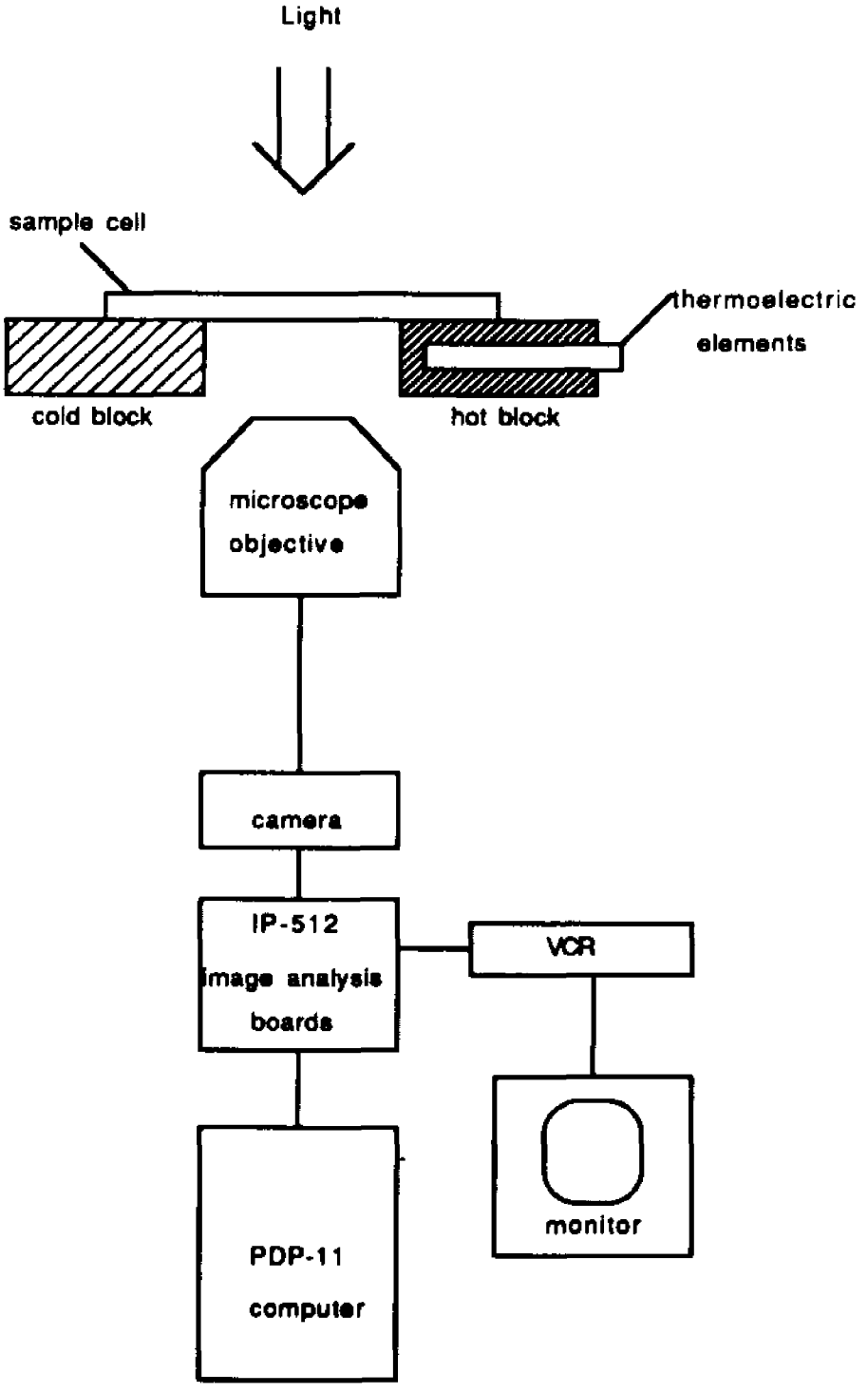
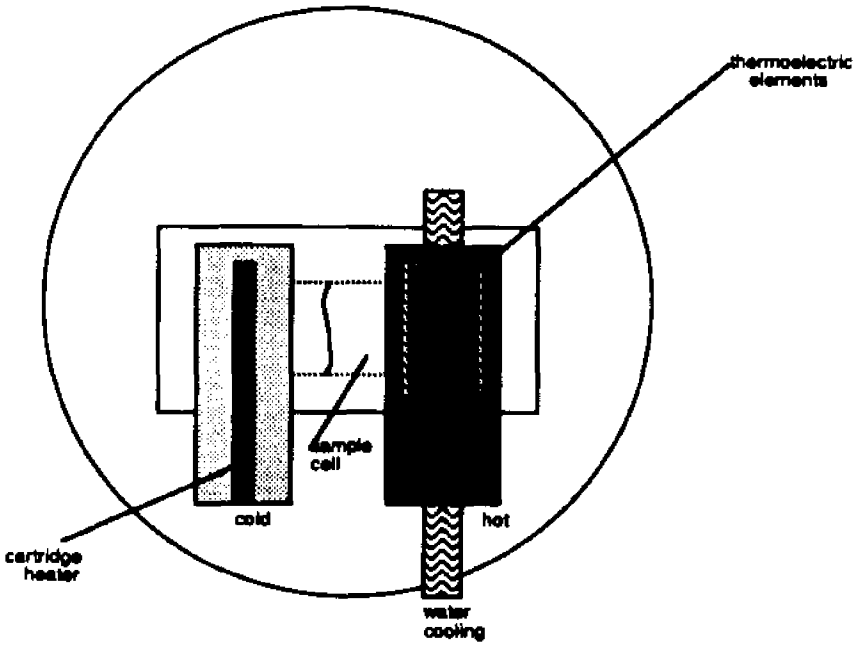


Fig. 9 Experimental set-up

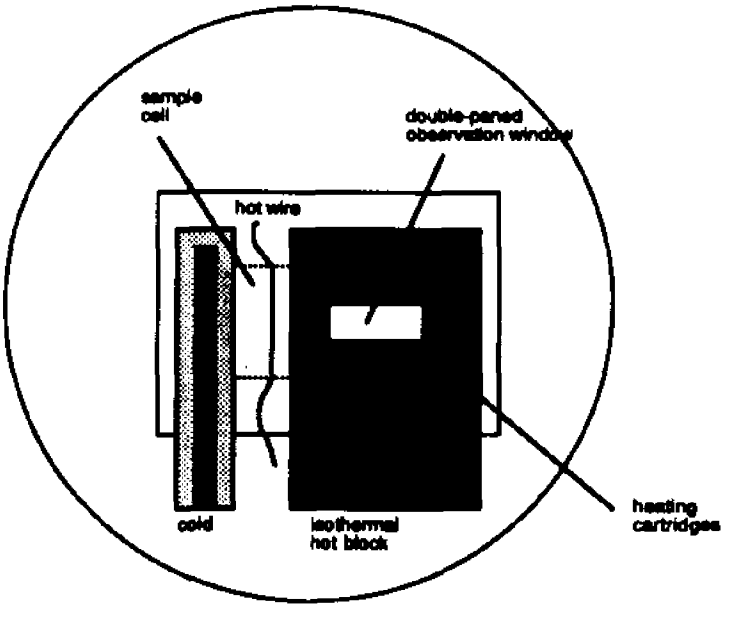
maginfications were used. The usual combinations for the experiments described here were 20x objective with 1x projection lens or 10x objective with 2.5x projection lens. The length scale calibrations have to be done separately. A Ronchi ruling of known spacing is brought into focus on the microscope stage so that the rulings appear as vertical stripes of equal spacing. A horizontal intensity scan is then made with the image digitization board (see section 4.3). The number of pixels per ruling is obtained and the distance/pixel factor can be inferred from the known spacing on the Ronchi ruling. For the 10x objective/2.5x projection lens combination, this factor is 0.96 μm/pixel and the full TV frame (512 x 480 pixels) corresponds to 490 μm x 460 μm. For other objective/projection lens combinations, the length scale is multiplied by the appropriate ratio.

4.2 Temperature controls

The basic design of the temperature control stage follows that of Hunt et al.[89] Two control stages (shown in Fig. 11) were constructed. The design shown in Fig. 11a is intended to be used with the 20x objective for studies of the initial instability. The hot and cold blocks are maintained at temperatures above and below the melting temperature of the sample by resistive heating cartridges (HotWatt Inc.) which are controlled by proportional controllers (Cole-Palmer Versa Therm model 2156). The proportional controllers provide temperature stability of 0.05 ^{o}C. An equilibrium crystal/melt interface is first established between the hot and cold blocks, then the hot block is cooled by thermoelectric cooling elements (Cambion Electric Co.). The removal of heat is assisted by running water through the upper hollowed part of the hot block. Due to the short working distance of the 20x objective, this design is open to the atmosphere with no thermal protection surrounding the cell. As a result, the equilibrium interface is



(a) Temperature stage design for observation of initial instability



(b) Temperature stage design for observing dendrite advancing into uniformly undercooled melt

Fig. 10

susceptible to general disturbances in the laboratory. Under normal conditions of the experiment, however, no interface drift can be detected. Another disadvantage of this design is that an initial temperature gradient is necessary to maintain the equilibrium interface. As the hot block is cooled to induce the crystal to grow, the initially imposed temperature gradient becomes a time- as well as space-dependent unknown factor. Because of this, it is difficult to directly compare experimental results with theory.

As indicated in Chapter 2, the recent theory on dendritic growth is mostly developed for a situation of the crystal growing into a well-defined undercooling. For this reason, the design shown in Fig. 11b was developed in which the sample cell is completely enclosed in isothermal temperature blocks. This stage is made from copper with Styrofoam insulation on the outside. Double-paned viewing windows below and above the sample cell completely isolate the sample cell from the atmosphere. The temperature controls are with cartridge heaters positioned symmetrically around the sample cell. The interface is established between the hot and cold blocks which start out with temperatures above and below the melting point of the sample. A hot wire (AC voltage applied to 0.001" diameter Cr wire purchased from Omega Temperature) runs along the melt side of the crystal/melt interface. As the temperature of the hot block is lowered to an amount Δ below the melting temperature of the sample, the hot wire keeps the interface from growing into the undercooled melt. When the current in the wire is turned off, the interface advances and eventually reaches that region of the sample which is maintained at a well-defined temperature by the copper block.

4.3 Image Acquisition Hardware

A set of three computer interfacing boards were purchased from Imaging Technology Inc. (ITI): Analog Processor (AP), Frame Buffer (FB) and Feature Extraction (HF). Installation of the three boards is shown on page 8-7 of the ITI

Technical Manual. The function of each board is controlled by a set of control registers accessible by the host computer. Details of the control registers can be found in Appendices C and D, and in the ITI Technical Manual.

The camera output is received by either a VCR or the image acqusition boards. The hardware configurations for the two cases are shown in Fig. 10. Jumpers J14A and J20B-C on the Analog Processor board have to be configured differently for the two cases (see page 4-21 of the ITI Technical Manual).

The analog processor is a fast A/D converter which digitizes each video frame into 512x480 pixels. Each pixel has intensity value ranging from 0 to 256 ($=2^8$, thus 8 bits/pixel). The 512x480x8 = 2 Mbits of data are stored in the FB board at the real-time video rate of 30 frames per second. Thus the information transfer between the boards is 512x480x8x30 = 60 Mbits/sec.

The frame buffer board stores the digitized picture and allows the host computer to read intensity values at any specified x-y locations of the image. It then passes the digitized picture back to the AP board to be reconverted into analog form for display on a conventional monitor. There appears to be no degradation of image quality in the A/D conversion and D/A reconversion process.

The feature extraction board works on the data stored in the FB board. A "valid feature" is defined as a pixel (x-y pair of coordinates) whose intensity falls within the user-specified range. In these experiments it is usually pixels that have intensities above a certain threshold specified by the user since the interface appears as a bright line over a dark background. The HF board has a buffer that can store a maximum of 4096 x-y pairs. When this buffer is full, a "flag" (one bit on one of the control registers on the HF board) is raised and no further feature extraction takes place until this flag is lowered. During this time the host computer has access to the extracted x-y coordinates via a control register on HF. In principle all of these things can happen in real-time (i.e. data from every frame can be worked on) but in reality some frames

have to be skipped between accessing data from one frame and getting ready to acquire data from the next frame. It was found that an optimal rate was to skip five frames between acquired frames (i.e. acquire every sixth frame, this translates to 5 acquired frames per second).

The PDP-11/73 computer is based on a 22-bit processor. This means there could be 4M ($=2^{22}$) directly addressable memory locations but our particular unit is equipped with 256K of core memory. Assembly Language, however, is still based on 16-bit addressing. Therefore some fancy memory mapping (Extended Memory Management) is implemented to take advantage of the higher memory locations. During image acquisition, a tremendous amount of data is passed to the host computer and it is necessary to make use of Extended Memory. Some elementary details of Extended Memory and its use can be found in Appendix C. The complete details are contained in RT-11 Software Support Manual and RT-11 Programmer's Manual. With the addition of the SYNC-LOCK unit from Macro-Tek Inc., it has become possible to do feature extraction using VCR recording as input. Since the VCR can freeze the picture and advance the tape frame by frame, feature extraction can be performed on every frame at a leisurely pace and computer memory is no longer a limitation.

4.4 Image Acquisition Software

The image digitization boards are controlled by a set of registers addressable as ordinary computer memory locations. Manipulating the functions of the boards is simply a matter of loading 0's and 1's into the correct bits in the registers. This is accomplished by software written in Macro-11 Assembly Language. Only a brief account of the software will be given here. For details of the programs, see Appendices C and D.

The procedure for the experimentation described here can be generally divided into the following steps

(i) image preparation and enhancement

(ii) acquisition of time-dependent data on the interface

(iii) manipulation of data into convenient form

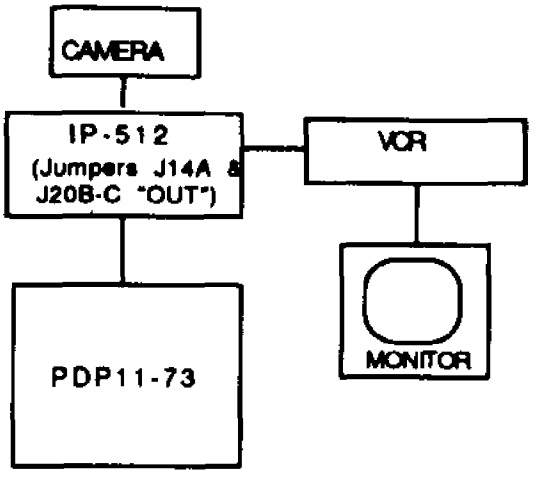
(iv) analysis of data

The computer programs are structured accordingly. Steps (i)-(iii) are performed on the PDP-11/73 with Assembly Language routines. After step (iii), data are transferred to VAX and step (iv) is carried out there with FORTRAN routines. In the following paragraphs, some salient features of the programs in (i)-(iii) will be described. Programs in (iv) for data analysis will be described in Chapter 5.

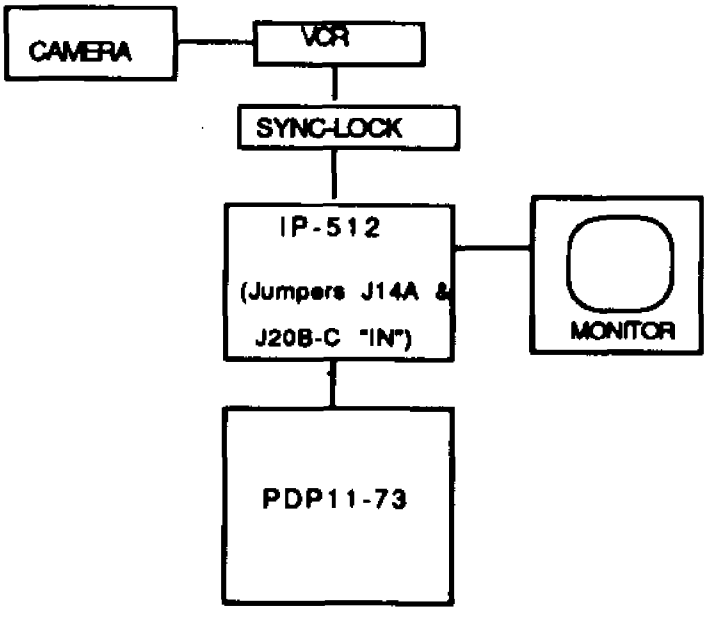
(i) Image preparation: This is in fact a single FORTRAN calling program, PREP.FOR, which performs a host of user-interactive functions listed in its menu; each function is a call to a subroutine. The collection of these subroutines is contained in PREPM.MAC, a MACRO-11 file. The most important function performed by this set of routines is a rudimentary image-enhancement routine named S (for STRETCH). The idea behind this routine is that, although the intensities can be digitized in values ranging from 0 to 256, in a usual picture only a small range is actually present (say values from 50 to 100). Now if this small range is "stretched" to the full dynamic range the hardware is capable of, then contrasts in the image will be enhanced. A linear mapping is done (in the example above, 50-->0 and 100-->256).

Before running the image acquisition routines in part (ii), it is necessary to set an intensity threshold. This informs the HF board that any intensity above this threshold is a valid feature. This function is performed by the command X, which will then prompt the user for the threshold value (between 0 and 256).

(ii) Data acquisition: As alluded to earlier, there are really two ways to acquire data from images. One is to use the camera signal directly as input to the image processing boards, the other is to use the VCR as input. Besides slightly different



(a) Hardware configuration for image acquisition directly from from camera signal



(b) Hardware configuration for image acquisition from VCR tape recording

FIG 11

hardware configurations (see Fig. 10), different programs are needed. The programs are grouped as follows:

Camera input -- XX1X.MAC, XXMAX.MAC, XXI.MAC

VCR input -- SS.MAC, SYNC.MAC, SYNCXY.MAC, SYNXYI.MAC

An important difference is that programs using camera input require Extended Memory on the PDP-11 and therefore must be run under the RT11XM operating system, whereas programs using VCR input can be run under RT11FB. Details of the programs and a practical sample run are contained in Appendices C and D.

(iii) Data manipulation: The data acquired by the programs above are stored in a user-specified file in binary form and with all data from all the frames in that run contained in one file.

It is more convenient to have the numbers in ASCII form and data from different frames separated. A set of FORTRAN programs were written for this task. Since data acquired by different programsin part (ii) are stored somewhat differently, different programs are needed for different situations. The correspondence is as follows:

Camera input: XX1X.MAC, XXMAX.MAC ----------> BINRY.FOR

XXI.MAC, SS.MAC, SYNC.MAC--------> XXIDY.FOR

VCR input: SYNCXY.MAC----------> BINRY.FOR

SYNXYI.MAC ----------> BINRI.FOR

The converted data are stored on floppy disks.

4.5 Sample Cells

As described in Chapter 2, the recent theoretical work on the dendrite problem was mostly developed for the two-dimensional case. Two-dimensionality is achieved if the diffusion of heat or impurtities takes place in the plane of growth. To this end, we

constructed thin cells to approximate this condition. Thin cells are also optimal for optical observation. There are, however, some doubts as to whether two-dimensionality can be achieved in the thermal diffusion case because even though the specimen is thin, the glass cells still may account for significant thermal conduction in the lateral direction.

Construction of the sample cell has gone through several variations of the same theme. Our cells are all constructed from microscope cover slides of thicknesses 0.15 mm or 0.20 mm. The dimensions of the cover slides are 22x22 mm^2 or 22x50 mm^2. The glass slides are cleaned by rinsing in chromerge solution followed by flushing with distilled water and acetone.

Our earliest sample cells were made, following the procedure of Jackson[104], by sealing three sides of two cover slides with a small propane torch. In this sealing process the molten edges of the glass slides "bead" up while resolidifying and there is a natural spacing of ≅20 μm between the two slides. Because of this beading, however, the cell did not sit flat on the temperture stage, thereby reducing the effectiveness of the thermal contact. There is also some question as to how uniform the spacing is between the glass slides. The advantage of this method of manufacture is that these cells are probably free of contamination.

Subsequently the cells were made by attaching the two slides with epoxy, with a spacer in between. The spacers are either a film of aluminum evaporated on one of the glass slides, or Mylar sheets cut to size. The thickness of the evaporated aluminum layer cannot be controlled but UV absorption path-length measurement (with Cumeran solution in the cell) made after evaporation indicated thickness of ≅ 5 μm. The Mylar sheets are obtained from du Pont Chemicals (Wilmington, DE). They come in a variety of thicknesses ranging from 2.5 μm and up. The actual thicknesses used in these experiments are 12 μm (0.0005"), 25 μm (0.001") and 50 μm (0.002"). It is not clear whether the epoxy has any chemical effect on the specimen studied.

The materials are loaded into the thin sample cells by dipping the cell into the melt. Then the melt fills the cell by capillary action. This process is much facilitated if the glass cells are warmed to above the melting temperature of the material, and if only a corner of the cell is dipped into the liquid. For improved cleanliness and purity of the material, a vacuum distillation set-up was assembled which allowed distillation as well as loading of the sample under vacuum. In this procedure, stock succinonitrile is first vacuum distilled for 5-6 times. In the last pass of distillation, the succinonitrile is collected into an evacuated freeze-dry bottle adapted to this purpose. A simple fixture holding the sample cell is inserted through the O-ring sealed opening of the bottle which allows vertical motion of the fixture. When enough succinonitrile has collected in the bottle, the sample cell is lowered into the (molten) succinonitrile.

4.6 Material Preparation

99% pure (Purissima Grade) succinonitrile stock was purchased from Fluka Chemical Corp. The earlier sample cells (on which most of the data in Chapter 5 were obtained) were made without further purification of the material. Later samples were made from multiply zone-refined or multiply vacuum-distilled material. The additional purification procedure produced no noticeable difference in the results of the initial instability but made a difference in the dendrite velocity and apparent dendrite shape.

The thermodynamic and other relevent parameters of succinonitrile are listed below[51]

	Solid	Liquid
chemical formula	$NC(CH_2)_2CN$	
crystalline structure	bcc	
molecular weight (g/mol)	80.09	

density ρ(g/cm^3)	1.016	0.97
melting temperature T_m(K)	331.24	
heat capcity c_p(cal/mol K)	NA	38.25
(erg/cm^3 K)		(1.94 x 10^7)
latent heat of fusion L(cal/mol)	885.1	
(erg/cm^3)	(4.49 x 10^8)	
unit undercooling L/c_p(K)	23.1	
thermal diffusivity D_T(cm^2/s)	1.14 x 10^{-3}	1.16 x 10^{-3}
solid/melt interface energy γ_o (cal/cm^2)	2.14 x 10^{-7}	
(erg/cm^2)	8.95	
$\gamma(\theta) = \gamma_o(1 - \varepsilon \cos n\theta)$		
capillary length $\gamma_o c_p T_m/L^2$ (^{o}A)	27.7	
surface tension anisotropy $\varepsilon=(n^2-1)\beta$	0.105	
where β=shape anisotropy	0.007	

The surface tension, as written in the Gibbs-Thomson relation, T(interface)=$T^o_m(1-\gamma\kappa/L)$, is the surface energy per area. If the surface energy is anisotropic, we write the surface tension as $\gamma(\theta)=\gamma_o(1-\varepsilon \cos n\theta)$. The anisotropy observed in the shape of liquid droplets trapped in solid, however, is <u>not</u> the anisotropy in surface tension. The relationship between the shape anistropy, $R(\theta)=R_o(1+\beta \cos n\theta)$, and the surface tension anisotropy is derived in Appendix B.

5. EXPERIMENTAL PROCEDURE AND DATA ANALYSIS

5.1 Experimental Procedure

Since the instability we want to observe takes place on a time scale of approximately 5 seconds, it is necessary to partially automate the data collection process. The image analysis/data acquisition is performed by the computer programs described in the previous chapter. Appendix C provides sample sessions at the terminal. In this section, some tricks towards the smooth running of the experiment will be pointed out.

Loading the sample cell is much easier if the glass cell is heated to above the melting temperature of the sample. For the cells made with metallic spacers or evaporated metal films, it is sufficient to place the cell in the flame of a Bunsen burner for 2 seconds and then immediately dip a corner of the cell into the melt. For cells made with Mylar spacers or any material that might not stand the heat from the flame, the cell can be heated on a hot plate. For succinonitrile, the hot plate is usually set to 90 °C. After the cell has been loaded, excess succinonitrile on the outside of the cell is immediately wiped off. Acetone is avoided in cleaning the cell lest it may contaminate the material inside.

After the loading procedure, the material inside the sample cell will solidify in a random manner. There will probably be many defects and grain boundaries in the solidified sample. This makes it difficult to establish a smooth equlibrium interface in the temperature gradient. There are several ways to improve this situation. The first is to melt the entire sample and resolidify gradually. This is achieved by melting the entire sample under the hot block and then either slowly pushing it out of the block or slowly turning down the temperature of the block until the sample resolidifies. This again is a random process and several tries may be necessary before a good sample is

obtained. The second way is via annealing. Place the cell inside one the temperature blocks at a temperature below its melting temperature (≅50 °C) for a few hours and then try to establish a solid/melt interface. Sometimes it is sufficient to leave the sample cell overnight under the usual temperature gradient of the experiment (25 C and 70 C). Annealing usually works better, even though it is a longer process. In all of the above, there is no control of how the crystalline axes are oriented within the sample cell. As a result, when the sample cell is placed under the temperature gradient, the crystalline axes may be oriented at an angle to the direction of the temperature gradient. This has the consequence that the dendrites will grow at an angle to the temperature gradient and in some cases this seems to induce a tip-splitting instability; the dendrites never develop a well-defined tip radius. This tilt can be remedied by rotating the entire cell by the appropriate amount, or by repeating the resolidifying procedure until a convenient orientation of the crystalline axes is achieved. Finally, it should be noted that contaminated samples are much more prone to formation of grain boundaries and defects. If a good solid/melt interface cannot be established after repeated solidification and annealing, it is probably best to start with another sample cell.

Finally, there are some tricks to enhancing the image quality and, in particular, the contrast. Make sure that the microscope optics is optimally adjusted and the optical path is free of excessive dust and dirt (see Nikon manual for procedure). In roughly descending order of importance, the contrast of the image can be adjusted by (i) size of the aperture diaphram openning, (ii) tuning of the Nomarski prism, (iii) illumination level of the lamp, (iv) focusing/defocusing of the objective and (v) gain and brightness control on the Dage-MTI camera. The adjustment of the aperture diaphram is the last step of tuning up the optics and is spelled out in the Nikon manual. The aperture should be opened to roughly 85% of the full opening for best contrast. The tuning knob on the Nomarski prism can be used to greatly affect the contrast (this can be appreciated only when the λ-plate on the condenser is at the "in" position). When adjusting for the

optimal contrast, it is best to monitor the image directly from the camera output because this is what really counts in data acquisition.

5.1.1 Measurement of the Initial Instability

When the equilibrium interface has been established and the microscope image has been optimized, one prepares the data acquisition program by running PREP.FOR as shown in Appendix C. Growth at the interface is induced by cooling the the hot block with thermoelectric elements. The heaters in the hot block are turned off and the thermoelectric cooling elements are turned on simultaneously. The Cambion controller for the thermoelectric elements is preset to achieve some final temperature whose precise value is not terribly important because the interface becomes dendritic well before the thermoelectric elements reach their final setting. With the Cambion controller set to 6 on the front dial, the cooling rate at the hot block was measured to be 0.37 C/sec. The cooling rate inside the sample cell near the solid/melt interface, as measured by an imbedded thermocouple of 0.001" diameter (Omega Engineering), was 0.2 C/sec. With this setting, there is a delay of ≅15 seconds before the interface starts moving. Initially the flat interface moves as a stable whole for ≅3 sec before noticeable instability is developed. Because of this delay, and because the data acquisition program (such as XXI.MAC) can only collect data for 5 sec before computer memory is full, the program is not initiated until there is some small but noticeable instability on the interface. During this time it often happens that the contrast of the image deteriorates as the interface becomes more complicated. It is therefore necessary to continuously monitor the image contrast by adjusting the microscope illumination lamp. The alternative way of data acqusition which gets around the problem of computer memory is to record the development of the interface on the VCR and then do the data acquisition from the VCR tape. Appendix C contains a sample session of this method. The raw data acquired is contained in a file named "FEXED.DAT" on the Winchester drive. They are

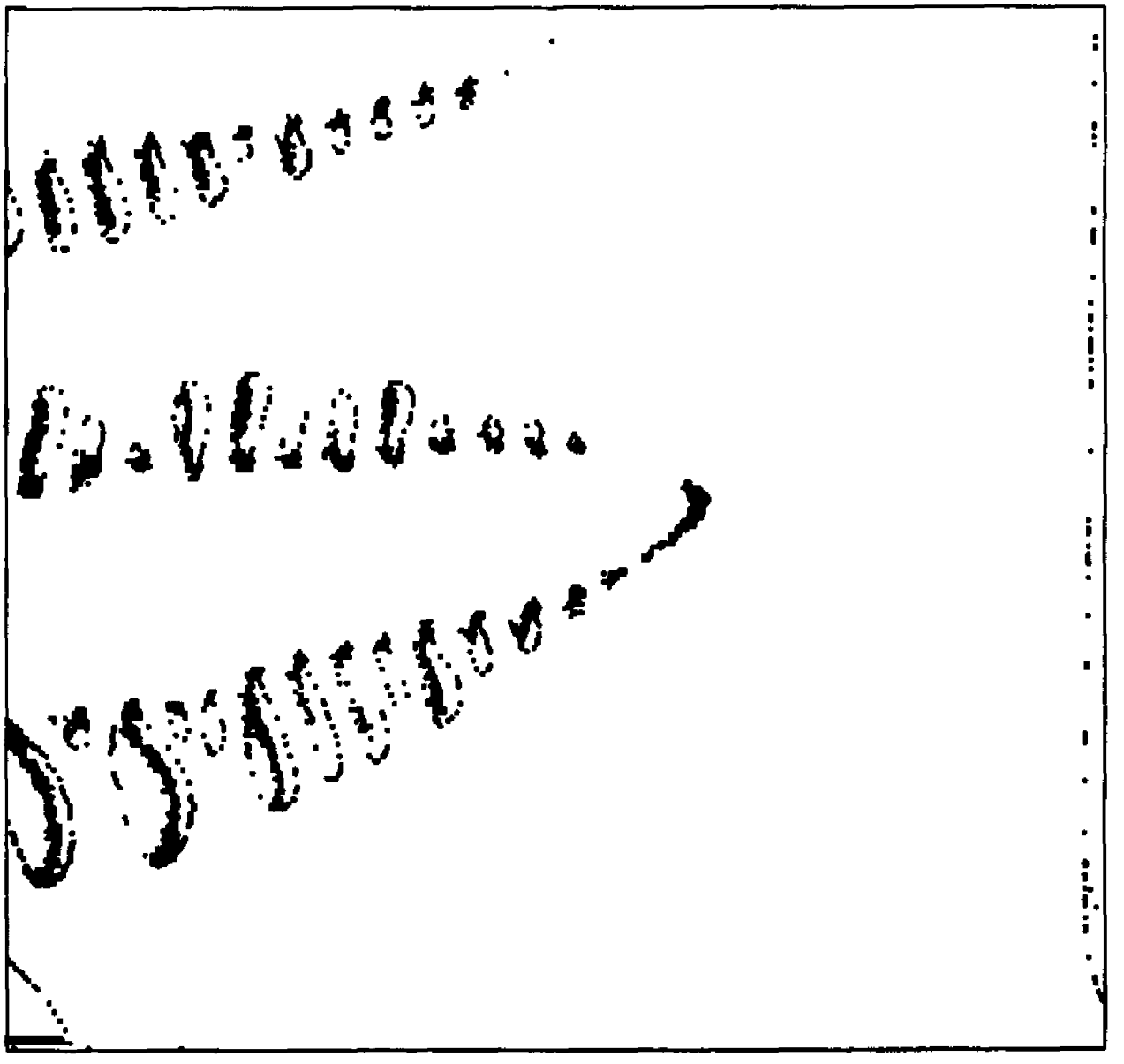


Fig. 12 Image of a dendrite

then converted to ASCII format and transferred to the floppy disk. Once the interface data is on the floppy disk, they are trasferred to VAX via KERMIT for analysis. The analysis programs will be discussed in Section 5.2.

5.1.2 Measurements on the Dendrite

In the current set-up, it is only possible to make measurements of the dendrite dimensions by using VCR recordings because the dendrites are too complicated to define a "valid feature" for the Feature Extraction board. The crudest measurements can be obtained simply by laying a ruler or template on the TV monitor, and then the measurement on the ruler can be converted to actual sizes because the microscope/camera have a calibrated length scale (Section 4.1). It is also possible, however, to find the outline of a dendrite by using the feature extraction board. This is accomplished by high-lighting the boundary of a dendrite with the image enhancement routine in PREP.FOR, in the same way one prepares the equilibrium interface. Then with a suitably defined threshold intensity for feature extraction, the program SYNCXY.MAC picks out all the pixels which cross that threshold. SYNCXY.MAC takes VCR output as input, hence feature extraction can be done frame by frame. Once the data has been transferred to the VAX, it can be plotted (Fig. 12) and measurements on the dendrite dimensions can be carried out more precisely. The disadvantage of this procedure is that it is time-consuming, especially if one needs to follow the dendrite shape over a large number of frames.

5.2 Data Analysis on the Initial Instability

We wish to deduce from our data the (spatial) Fourier spectrum of the interface and the growth rate of each Fourier component. During the development of the initial

instability when the growth may still be considered linear, the interface shape y(x) may be Fourier-analyzed as

$$y(x) = y_o + \sum_k a_k(t) \cdot e^{ikx} = y_0 + \sum_k a_k(0) e^{\omega_k t} \cdot e^{ikx} \qquad (5.1)$$

The interface data y(x;t) will allow us to find the Fourier coefficients $a_k(t)$ hence the growth rates ω_k. The raw data from the feature extraction program have to be conditioned before Fourier analysis can be effected. The conditioning goes through the following steps: (i) interpolation of interface location[73], (ii) rotation to get rid of "DC bias and ramp", (iii) B-spline fit to interface shape[91,92]. Interpolation and B-spline fit allow us to obtain significant data during sufficient early stages for the linear stability analysis to be valid.

(i) The raw data is in the form of [y, x, I(x,y)] where I(x,y) is the intensity at the pixel (x,y). Since the interface has a finite width (usually 3-6 pixels), for each scan line y there will be several x values, each having a different value for intensity. Following Dougherty et al.[73], this fact is used to our advantage in deterimining the position of the interface. For each scan line y, a parabola is fitted through the set of intensities [x, I(x)], i.e. $I(x)=b(x-x_0)^2$. The best-fit x_0 is then defined to be the position of the interface for this scan line y. This interpolation procedure is performed by the program **interp.f**. The end result is the set of interpolated interface coordinates [x,y(x)].

(ii) Sometimes there is misalignment of the camera or of the sample cell, the interface may appear as a slightly slanted vertical line. In this case a "DC ramp" is built into the interface data $y(x)=a_0 + bx + y(x)$. When Fourier transformed, the ramp term introduces a 1/k contribution in the Fourier spectrum which is unwanted. The program named **rotat.f** gets rid of the DC contributions a_0+bx by fitting the (interpolated) interface coordinates y(x) to a straight line and then rotating the

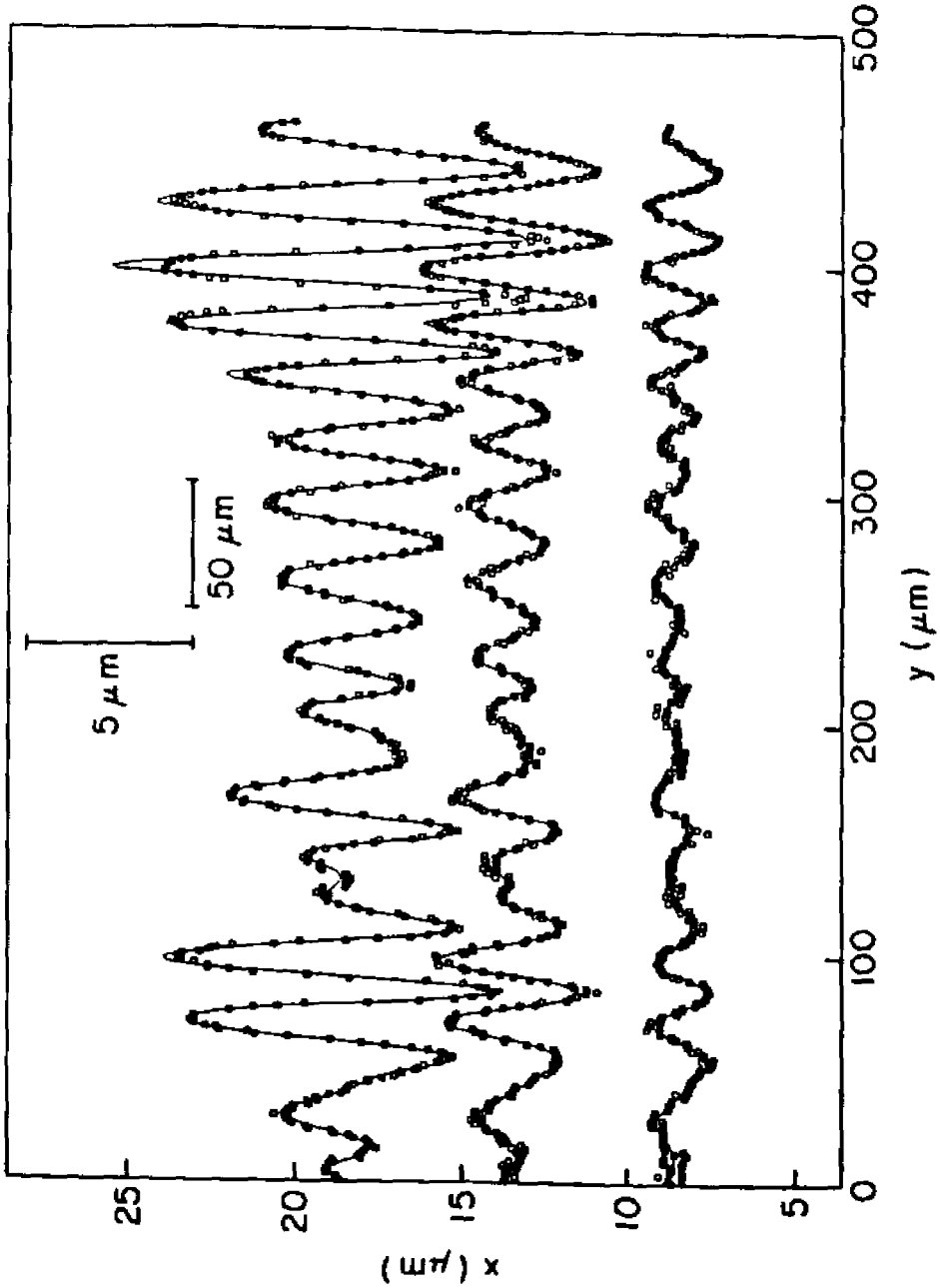


Fig. 13 Evolution of the interface shape

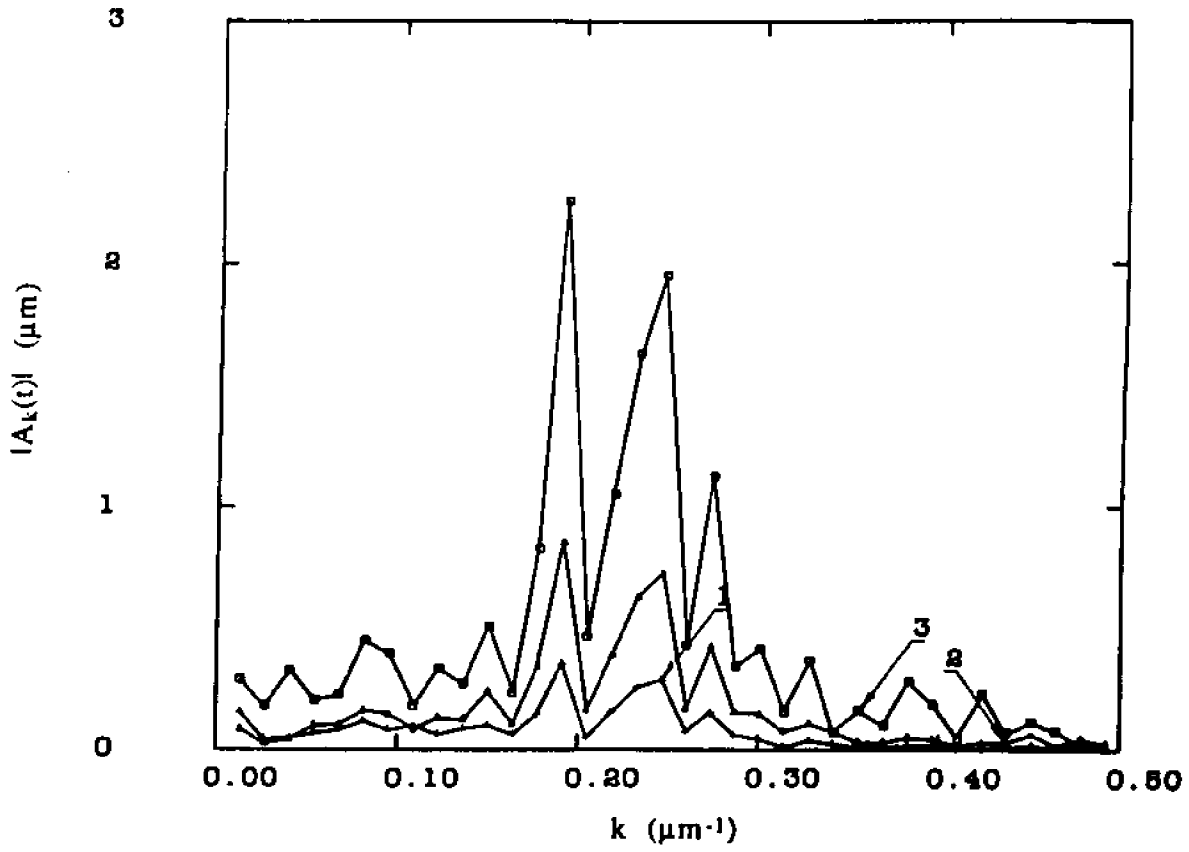


Fig. 14 Fourier spectra of the interfaces shown in Fig. 13

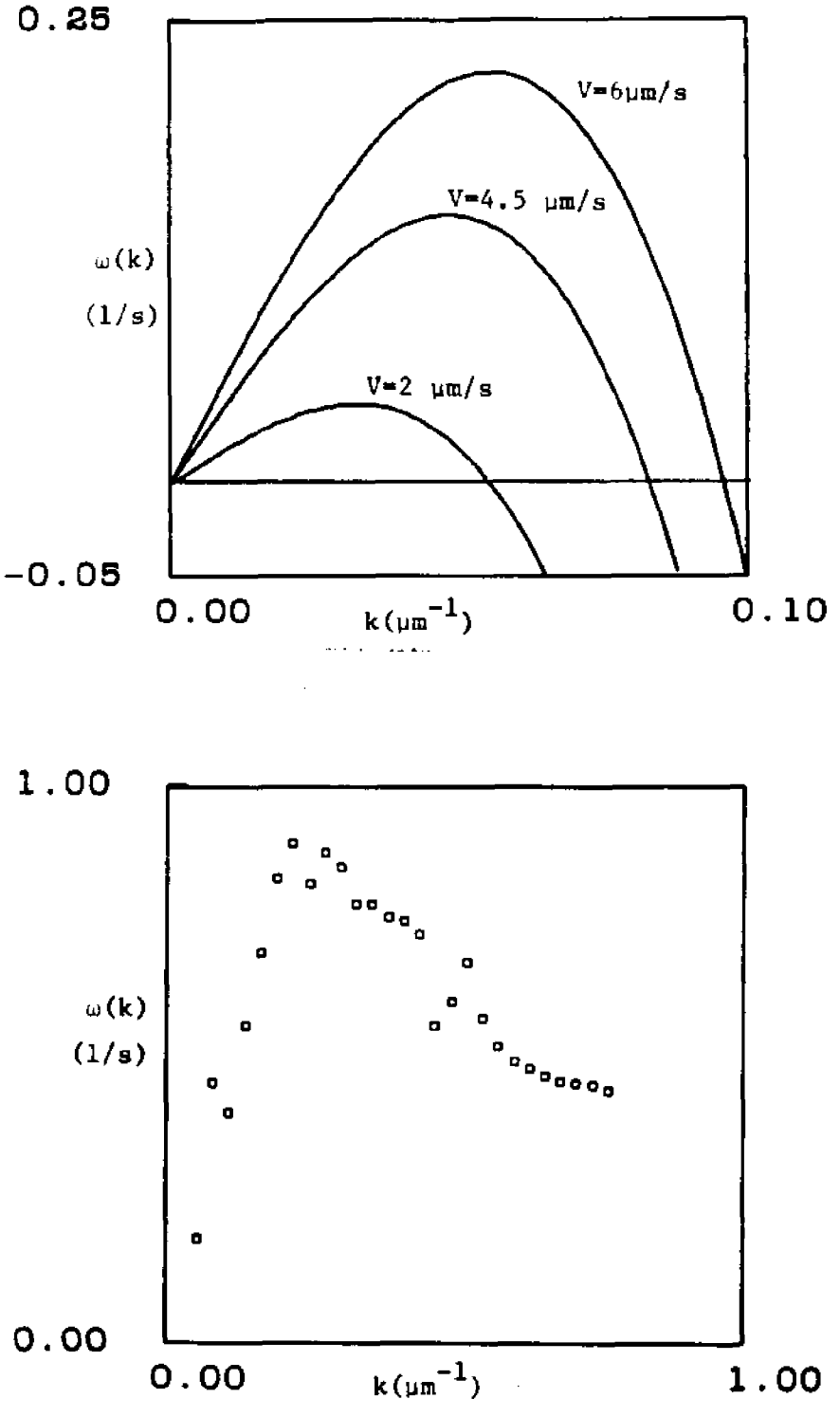


Fig. 15 Growth rates of the Fourier components
(a) Mullins-Sekerka result (Eq. 2.28) plotted for succinonitrile
(b) Experimental result

interface through the slant angle defined by this line. In so doing it has been assumed that the "true" distortion on the interface, $y(x)$, is random in nature and therefore equally likely to be positive and negative. The constant $a_o(t)$ is the mean position of the interface for that time. The instantaneous mean positions of the interface $a_o(t)$ for the duration of the run give us the mean velocity of the interface. For the interfaces shown in Fig. 13, the velocity was measured to be a constant 4.5 μm/s over the 2.2 second span of the 12 frames.

(iii) Next the interface coordinates $y(x)$ are fitted by a least-square cubic B-spline procedure[91,92]. The B-spline is a piecewise polynomial fit over a specified interval of the data with the end points of the interval treated specially so that a smooth, continuous curve results over the entire domain of the data. For our interface data, the best fit results from a least-square fit over 10 data points as an interval. This fitting procedure is performed by a program named **splyn.f**, which calls the standard B-spline subroutines in the UNIX Port library. The program fits the B-spline polynomials over the points $y(x)$ with the least-square procedure. Using the B-spline coefficients thus obtained, it then generates a set of points $\zeta(x_i)$ at regularly spaced points x_i. The points x_i are positioned at spacings of $(x_{max}-x_{min})/n$ where n is a parameter supplied by the user (typically n=400-500). The end result of the program is a set of n points $[x_i, \zeta(x_i)]$. In Fig. 13, the data $y(x)$ is plotted along with the the B-spline generated set $\zeta(x_i)$. Note that the combination of interpolation and B-spline fit enables us to determine the interface coordinates to a high accuracy. In Fig. 13, it can be seen that the interface coordinates are ascertained to an accuracy of ≅0.1μm, which is better than the intrinsic resolution of the microscope!

With the regularly spaced points $\zeta(x_i)$ we perform the Fourier transform. The FFT[91,93] is a standard subroutine which returns real and imaginary parts of the Fourier coefficients a_k. The modulus $|a_k|$ is plotted in Fig. 14 for the interface shapes shown in Fig. 13. The striking feature of the Fourier spectrum is that it appears to be

"noisy". Neighboring points in k-space have very different values of Fourier coefficients while the deterministic Mullins-Sekerka theory can only predict a smoothly varying spectrum. We have interpreted this result as indicating that the growth of individual k components in the initial stages is independent and is driven by stochastic initial conditions. This point will be further expanded on in Chapter 6.

There is an alternative way of performing the Fourier transform devised by Lax and Agrawal[94], which works on the B-spline polynomial coefficients instead of the data points themselves. They call this the "continuous" Fourier transform because in principle one could compute the Fourier coefficients at any arbitrary k value rather than just at regular intervals of the k-space. The continuous Fourier transform works on the principle that, given the B-spline coefficients f_j representing an interval

$$\zeta(x) = \sum_j f_j B_j(x) \tag{5.2}$$

since the Fourier transforms of the polynomials B_j are known functions $F[B_j(x)]$, the Fourier transform can be computed thus

$$F[\zeta(x)] = \sum_j f_j \cdot F[B_j(x)] \tag{5.3}$$

This function is performed by the program **laxft.f** which calls two subroutines **cft.r** and **endcor.r**. These subroutines return the real and imaginary parts of the Fourier coefficients. Note that even though the Fourier coefficient can be computed at an arbitrary point in k-space, the minimum spacing Δk between two adjascent points in k-space still has to obey the uncertainty principle

$$\Delta k (x_{max} - x_{min}) \geq 2\pi \tag{5.4}$$

This precaution has been built into the calling program **laxft.f**.

There is a consolidated program named **spect.f** which performs all of the above functions in one run. **spect.f** calls **subinterp.f, subrot.f, subsplyn.f, subfft.f, subcft.f, cft.r** and **endcor.r**, which in turn perform the interpolation, rotation, B-spline fit, FFT or continuous Fourier transform.

When a time-sequence of interface shapes has been Fourier transformed, we have a set of Fourier coefficients in time $\{a_k(t)\}$ for each k value. In the linear (Mullins-Sekerka) regime, the Fourier coefficients are assumed to grow as $a_k(t)=a_k(0)\ e^{\omega(k)t}$. For each k value, if we fit the set of values $\{\ln a_k(t)\}$ vs time t, the slope will give us the growth rate ω_k. As alluded to earlier, the Fourier spectra are "noisy" which complicates the extraction of the growth rate ω_k. In order to partially smooth out the extreme noise in the Fourier spectrum, we use an ensemble averaging procedure. If the interpretation of the stochastically driven initial conditions is correct, then ideally one could ensemble average the corresponding instances of the interface over many runs to "wash out" the noise. In practice, however, it is impossible to establish, between different runs, interfaces that correspond in time. Another way of doing ensemble averaging is by way of analogy with stationary stochastic processes. If the initial destablization on the interface is stochastic and if we had an infinitely long interface, then the interface shape is much like the time-record of voltage fluctuations across a resistor. If this time-record is divided up into pieces each lasting a time T, provided T is much larger than the correlation time of the fluctuations, each piece can be regarded as a member of the ensemble. Similarly, we can divide our interface into segments each spanning a distance much larger than the apparent correlations length of the interface shape, and regard each piece as a member of the ensemble. A destablized interface of total length 500 μm is usually characterized by a wavelength of 30 μm and typically we cut this interfaces into segments of 200 μm length each. 10 such segments are made on an interface. They are individually Fourier transformed and the resulting spectra averaged together. When this procedure is performed on a time-sequence of interfaces, what we get is the time

evolution of the ensemble-averaged Fourier coefficients, $\{<a_k(t)>\}$. These coefficients are fitted to the exponential time evolution $<a_k(t)> \sim e^{\omega(k)t}$ to obtain the growth rates ω_k. The fit to the exponential time dependence is performed by the subroutine **subexp.f**. There is a consolidated program named **ensemb.f** which performs the ensemble-averaging of the spectra and obtains the growth rates ω_k as described above. **ensemb.f** accepts the raw data as input and in turn performs interpolation, rotation, B-spline fit, ensembe-averaging of Fourier spectra (either FFT or continuous transform), exponential fit to the time dependence of the Fourier coefficients. The output of **ensemb.f** is the set of growth rates ω_k for all the allowed k values. Fig. 15 shows the growth rates ω_k obtained from 12 frames of interface data (3 of which are shown in Fig. 13) by fitting $<a_k(t)>_{FFT} \sim e^{\omega(k)t}$.

5.3 Data Analysis on the Dendrites

As described in Section 5.1.2, our preliminary measurements on the dendrites are rather crude. It is possible to make a fairly good measurement of the tip-radius by either freeze-frame or by using the feature extraction board as described in Section 5.1.2, but velocity of the dendrite tip is difficult to ascertain for the entire duration of a run. This is because typically the dendrites are followed under the microscope by periodically moving the microscope stage. Because of this movement, there is no fixed reference on the VCR footage against which the absolute velocity of the dendrite can be tracked from the beginning to end. The momentary velocity can be measured when the dendrite moves from one end of the screen to the other, at which point the microscope stage is shifted to keep the dendrite in view and a new measurement of the velocity will have to be made. When the cooling rate of the hot block is 3.7 ^{o}C/s as described in Section 5.1.1, after the transient has died out and the dendrites seem to grow at a steady

velocity, the velocity of the dendrites was measured to be 45 $\pm$10% μm/s. The dendrite tip radii, as measured by parabolic templates of known tip radii, are ~ 16 μm.

6. RESULTS AND DISCUSSION

The two most striking results of the data analysis, as shown in Figs. 14 and 15, are the discrepancy with the simple-minded Mullins-Sekerka analysis and the fact that the Fourier spectra of the interfaces are noisy. As alluded to in Section 5.1.1, the noisy spectra are interpreted as due to thermal noise in the system which gives rise to stochastic initial condition and driving force during the growth of the solid. This picture seems to be consistent with some numerical simulations, as will be discussed in Section 6.1. The discrepancy of the initial length scale and the growth rates with the Mullins-Sekerka analysis is probably due to experimental conditions different from what is assumed in the theory, but a quantitative account of the difference seems to be difficult, as will be discussed in Section 6.2.

6.1 The Role of Noise

The deterministic Mullins-Sekerka analysis presented in Section 2.2 treats all k-components equally and there is no account of noise in the system and no mechanism for proving the initial distortions. In real experimental systems, there is always noise present. Because of noise and fluctuations in the initial condition, different k-components may get started differently. Once the instabilities characterized by these k components get "kicked" into action, they should then follow the growth rates predicted by the deterministic equations of Mullins and Sekerka. This situation is analogous to a pencel standing on its end. In this unstable position, all directions of fall are equally likely. Noise determines which direction the pencil will actually fall and once it starts falling in that direction, its motion is determined by Newton's equation. Dougherty et al.[73] recently made a careful study of side-branching activity on a NH_4Br dendrite growing from solution. They came to the conclusion that side-branching[107,108] was

probably due to noise (concentration fluctuations) around the tip. The idea of stochastically driven initial conditions has also been used by Meyer et al.[95] in their study of the initial stage of roll formation in the Rayleigh-Benard system. They pointed out, however, that the noise level needed to fit their data is 6 orders of magnitude larger than the estimated thermal noise in the actual experimental set-up.

In the absence of noise, the Mullins-Sekerka analysis assumes the Fourier amplitudes grow exponentially in the linear regime, $a_k(t) \sim e^{\omega t}$. The simplest way to incorporate noise is to take the phenomenological approach of the Langevin equation and write the evolution of the k-th Fourier component as

$$\frac{da_k(t)}{dt} = \omega_k a_k(t) + f_s(t) \tag{6.1}$$

$$\langle f_s(t) f_s(t') \rangle = 2s^2 \delta(t - t') \tag{6.2}$$

where $f_s(t)$ is a zero-mean Gaussian (white) noise term with standard deviation s, and ω_k is the usual Mullins-Sekerka growth rate as given by Eq. 2.28. This stochastic differential equation is numerically integrated on the computer as

$$a_k(t = N\Delta t) = \sum_{j=0}^{N-1} \omega_k a_k(j\Delta t)\Delta t + f_s(j\Delta t)\Delta t \tag{6.3}$$

with $f_s(t)$ generated by a zero-mean Gaussian random number generator whose rms amplitude s is a variable parameter. At t=0, all Fourier coefficients start from $a_k(0)=0$ and they "random walk" until such a time that a sufficient amplitude has been built up, so that the deterministic growth term $\omega_k a_k(t)$ dominates from there on. Fig. 16 shows a particular Fourier amplitude as a function of time for three different noise levels of the simulation. The random-walk feature at small times is quite apparent.

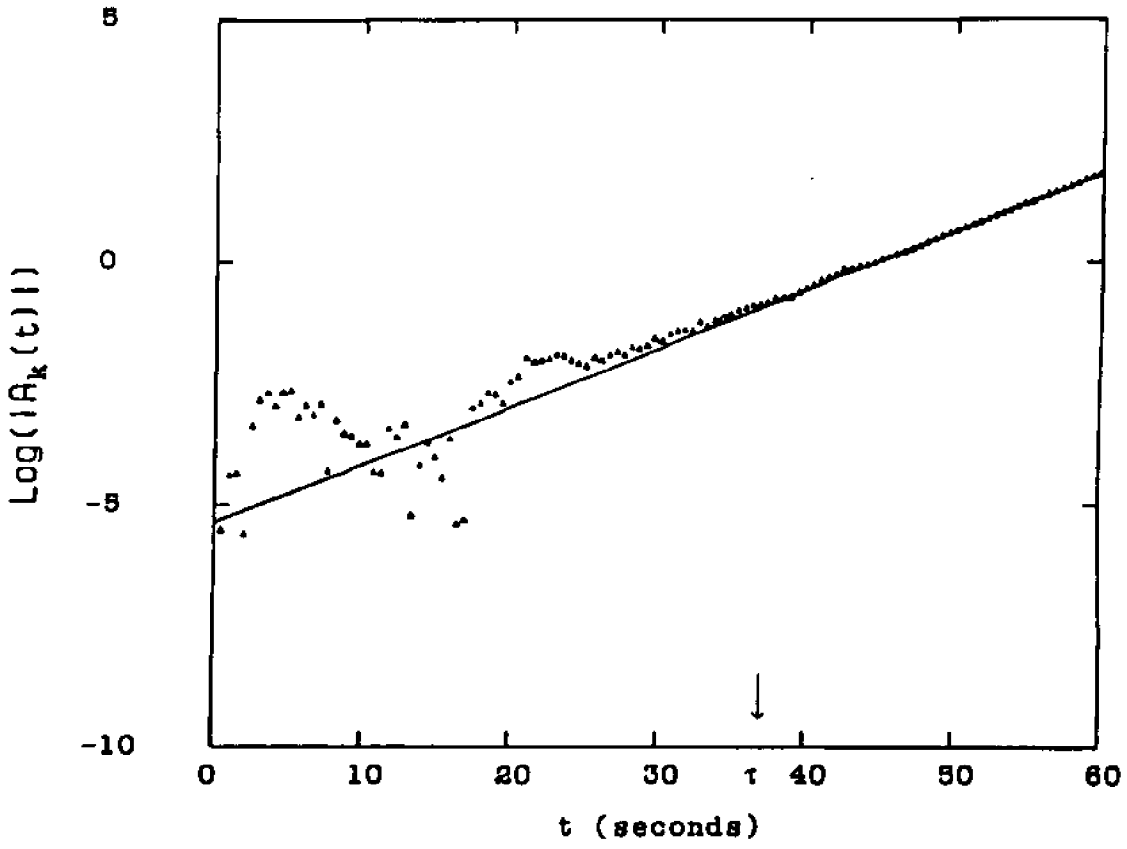


Fig. 16 Development of a Fourier coefficient according to the Langevin equation

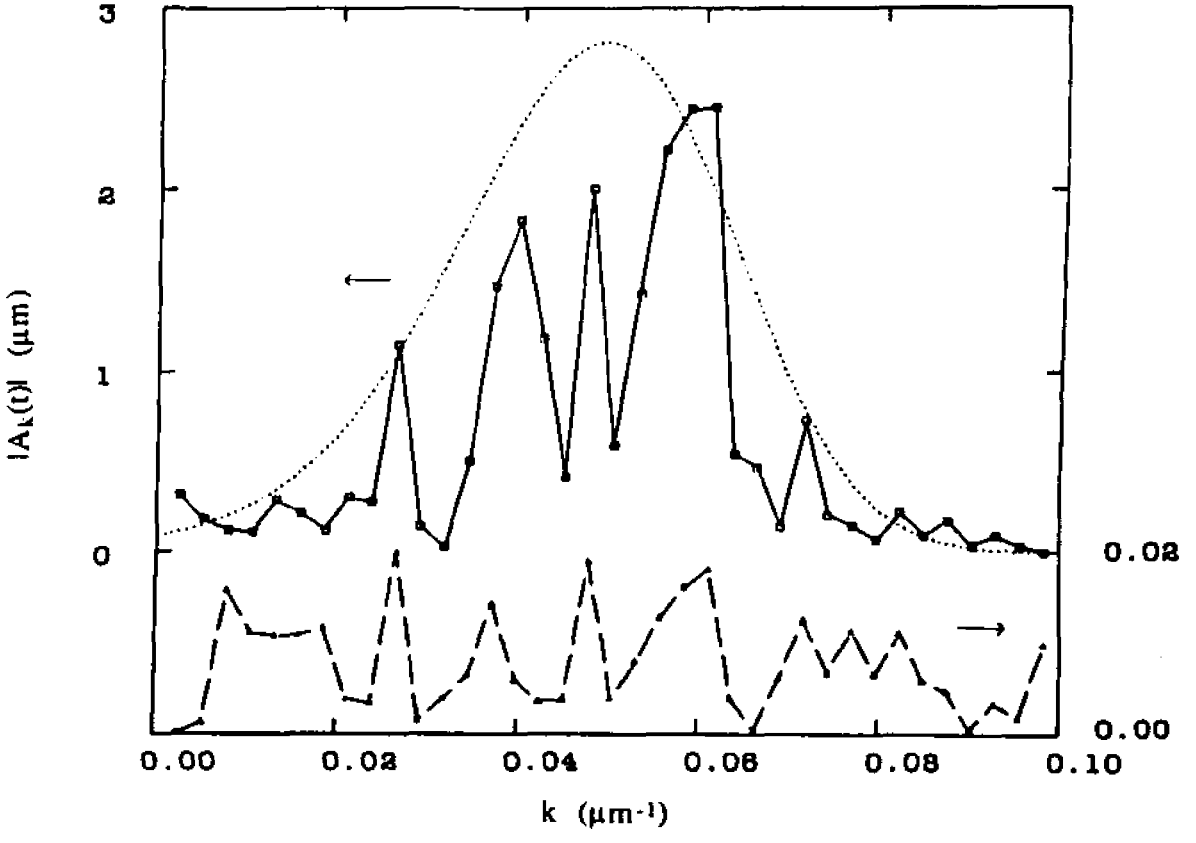


Fig. 17 Fourier spectrum as simulated by the Langevin equation

Fig.17 shows Fourier spectra, at two different times, obtained from such a simulation with s=0.02 μm/s. At t=2.5 sec the spectrum is noisy with no apparent effect of the deterministic growth term. At t=20 sec, we obtain a noisy spectrum resembling spectra obtained from actual experiments.

The origin of the stochastic noise term is most likely thermal fluctuations in the system. A rough estimate of the magnitude of this effect can be found from equilirium statistical mechanics, where the temperature fluctuation ΔT in a volume V is given by[96]

$$(\Delta T)^2 = \frac{k_B T^2}{c_p V} \qquad (6.4)$$

The fluctuation term $f(t)$ has dimension of velocity, so we need to translate this temperature fluctuation into a velocity fluctuation. For a nonfaceting crystal like succinonitrile, the growth kinetics at the crystal-melt interface is linear, $V_g = \mu_l \Delta T$ where μ_l is a kinetic coefficient and ΔT the undercooling ahead of the interface. The kinetic coefficient is a difficult parameter to establish experimentally, but we take the value of 17 cm/sec K for succinonitrile from the literature[97]. The temperature fluctuation depends on the volume of the system. We take as the characteristic length the wavelength λ corresponding to the maximally unstable k-component. From Eq. (2.28) we get this length as $\lambda = (l d_0)^{1/2}$, where $l = 2D/V$ is the thermal diffusion length and d_0 is the capillary length. Using v=4.5 μm which is the observed growth velocity, we get λ=194 μm. Putting this into the temperature fluctuation expression and then translating it into velocity fluctuation, we get 3.2 μm/s, which is within two orders of magnitude of the noise amplitude of (0.02 μm/s) required to produce the spectra shown in Fig. 17. This result, though only a rough estimate, is supportive of the idea that the noisy spectrum obtained from experimental data was due to stochaticity in the system.

Some additional remarks are in order. Firstly, we have assumed no k-dependence on the noise term $f_s(t)$ since we do not see any reason why it should be k-dependent. With a Lorentzian k-dependence built into the noise term, where the width of the Lorentzian was taken to be $k_c=(ld_0)^{-1/2}$, we saw no difference in the resultant simulated spectrum. Secondly, we found that the "cross-over" time from the random-walk part to the deterministic growth part of the amplitudes does not depend on the rms noise level s.

6.2 The Initial Length Scale

As can be seen readily on Fig. 15, the k-value charaterizing the maximally unstable mode as predicted by the Mullins-Sekerka analysis is 0.05 μm^{-1} whereas the experimental results show a peak centered around 0.25 μm^{-1}. The discrepancy could be due to several facts which will be discussed, in descending order of importance, in the following paragraphs.

Chief among the experimental difficulties is the fact that the theoretical analysis starts from a steady state planar interface growing into a uniformly undercooled melt, which is clearly not the case in the experiment. The only way to achieve an initially planar solid-melt interface is by imposition of a temperature gradient as we have done. As the hot block is cooled to induce the growth, the melt ahead of the interface is not uniformly undercooled. In fact, there is a (time-dependent) positive temperature gradient ahead of the interface which acts to stablize the interface. To incorporate this time-dependent temperature gradient into a theoretical analysis is no trivial matter, but the more immediate question is what effect <u>any</u> positive temperature gradient will have on the stability of an interface. A stability analysis[98] can be made of a geometry in which an external (positive) temperature gradient G is imposed and the interface is assumed to grow at speed V controlled by thermal diffusion. This analysis shows that the

interface is always stable. The fact that our interface goes unstable means that either there is a temperature inversion somewhere along the growth, or that the growth is actually controlled by chemical diffusion and the intability is driven by constitutional undercooling.

To estimate the effect of constitutional supercooling, we not only need to know what impurities are in the sample, but also their diffusion constants. Neither of these is available to us. But based on Trivedi's data[69] for acetone in succinonitrile, we estimated that the diffusion length $l_{chem}=2D_{chem}/V$ is lowered by 2 orders of magnitude, while at the same time the capillary length d_0 is increased by 2 orders of magnitude. The Mullins-Sekerka length $2\pi/k_c=2\pi(ld_0)^{1/2}$ is thus unchanged. This is indeed our observation when 5-50% volume fractions of acetone were deliberately added to our succinonitrile stock, the length scale of the initial instability was not much changed, although the dendrites had finer tips and side-branches.

There are some doubts as to whether our sample cells can really be considered two-dimensional. If the growth takes place by thermal diffusion control, then two-dimensionality is a good approximation when thermal diffusion takes place in the plane of growth. The thermal conductivity of succinonitrile melt is 5.32×10^{-4} cal/cm s K (thermal diffusivity is 1.16×10^{-3} cm^2/s), the thermal conductivity of most commercial glasses is approximately 2×10^{-3} cal/cm s K (thermal diffusivity - $3\text{-}8 \times 10^{-3}$ cm^2/s, several times larger than that of succinonitrile). This means that heat conduction through the glass actually cannot be ignored! The effect of lateral heat conduction is to increase the effective thermal diffusion length in the plane of growth because the in-plane heat flux $j_{in\text{-}plane}=\kappa \nabla T$ is reduced. The Mullins-Sekerka length $k_c=(ld_0)^{-1/2}$ is corrected downward, which is in the wrong direction for explaining the discrepancy in Fig. 15.

Finally, there may be wetting effects between the material and glass. It is difficult to quantitatively estimate the effect of wetting on the length scale of the

Mullins-Sekerka instability. Fattinger et al.[79] studied dendritic solidification of dimethyl sulfoxide in narrow gaps and found that for cell thickness below 2 μm, wetting effects are important and the dendritic morphology breaks up into droplets. Our sample cells are always between 5 and 50 μm thick and within this range the morphology is always regular dendritic. For cell thickness above 50 μm, the dendrites appear to be three-dimensional while the length scale for the initial instability was unchanged.

6.3 Dendrite Dimensions and σ^*

Several succinonitrile solidification experiments were performed to measure the tip radii and velocities of dendrites. The runs were made under identical conditions with initial end-point temperatures set at 22 °C (room temperature) and 70 °C, and the controller to the Peltier heaters set to 7. 15-20 sec after the initial instability, the dendrites were moving with fairly constant velocity and inter-dendrite spacing. The velocities measured on these runs was 45 $\pm$10% μm/s, where the error represents the range of velocities measured. The tip radii measured were 16 $\pm$ 5% μm. With these parameters, we can compute the stability parameter $\sigma^* = l d_0/\rho^2 = 0.032 \pm 10\%$ (l=2D/v*=0.46 cm, d_0=17.7 x 10^{-8} cm). The value of σ^* as predicted by microscopic solvability for pure succinonitrile growing in two-dimensional geometry[99] is 0.024. Given the limited precision of our measurements on V* and ρ^* as well as Glicksman's measurement on the surface tension anisotropy, our value of σ^* is in reasonable agreement with theory. Dougherty and Gollub[75] recently measured V*, ρ^* and ε for NH_4Br and found $\sigma^* = 0.081$. The theoretical prediction of σ^* for their system is 0.065. In Glicksman's measurement[50,51] of three-dimensional dendrites of succinonitrile, he found $\sigma^* = 0.0195$. The microscopic solvability theory (developed subsequent to Glicksman's original measurements) predicts a value of σ^* in good agreement with his measurements.

APPENDIX A

Derivation of the Gibbs-Thomson Relation

Consider a solid in equilibrium with its melt. In the absence of curvature, the equilibrium temperature and pressure are T_m and p_o, where T_m is the bulk melting temperature. In general, the change in the Gibbs free energy can be written as

$$dg = - sdT + Vdp \tag{A.1}$$

When interface curvature is present, interface temperature and pressure are slightly different from T_m and p_o. For small deviations of temperature and pressure, entropy and volume may be assumed to be constants. We can thus integrate Eq. (A.1) to get

$$g = - s(T - T_m) + V(p - p_0) + K \tag{A.2}$$

where K is an integration constant. So the solid and melt Gibbs free energies are

$$\begin{aligned} g_s &= - s_s(T_s - T_m) + V_s(p_s - p_0) + K_s \\ g_l &= - s_l(T_l - T_m) + V_l(p_l - p_0) + K_l \end{aligned} \tag{A.3}$$

The integration constants K_s and K_l are equal since, in the absence of curvature, the solid and liquid coexist at temperature T_m and pressure p_o (i.e. when $T_s=T_l=T_m$ and $p_s=p_l=p_o$, $g_s=g_l$).

Quite generally, there is a pressure difference accross a curved surface due to surface tension. This can be derived by considering the forces acting on an elemental area $dA=dxdy$ of a (spherical) surface (Fig. 18a). If the surface tension is γ, the forces acting on the sides of length dx is γdx. The component normal to the area is thus $\gamma dx d\theta = \gamma dxdy/R$. Similar analysis is made on the side of length dy. So the total force normal to the elemental area is $2\gamma dxdy/R$. This perpendicular force must be balanced by a pressure difference; $\Delta p dxdy = 2\gamma dxdy/R$, thus $\Delta p = \gamma(2/R)$. More generally, when the

elemental area is not on a sphere, the pressure difference is $\Delta p = \gamma\kappa$, where κ is the curvature. We apply this result to our curved solid-liquid interface:

$$p_s = p_l + \Delta p = p_l + \gamma\kappa \tag{A.4}$$

Because of this increase in pressure, the Gibbs free energy of the solid g_s is increased, and hence the equilibrium interface temperature is lowered (see Fig. 18b) by an amount ΔT

$$T_{interface} = T_s = T_l = T_m - \Delta T \tag{A.5}$$

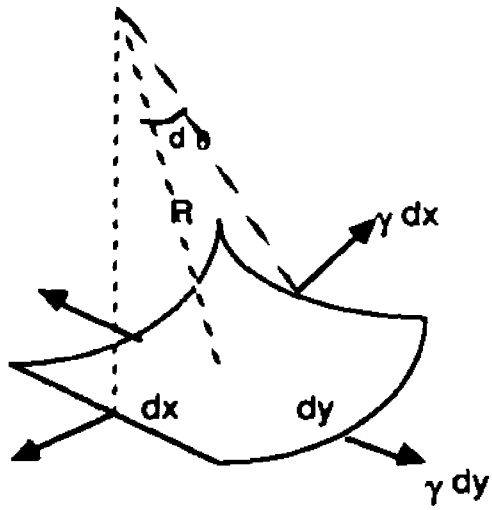
Now put Eqs. (A.4) and (A.5) into Eq. (A.3), with $p_l = p_o$

$$g_s = -s_s[(T_m - \Delta T) - T_m] + V_s[(p_o + \gamma\kappa) - p_o] + K = s_s\Delta T + V_s\gamma\kappa + K$$
$$g_l = -s_l[(T_m - \Delta T) - T_m] + V_l(p_o - p_o) + K = s_l\Delta T + K$$

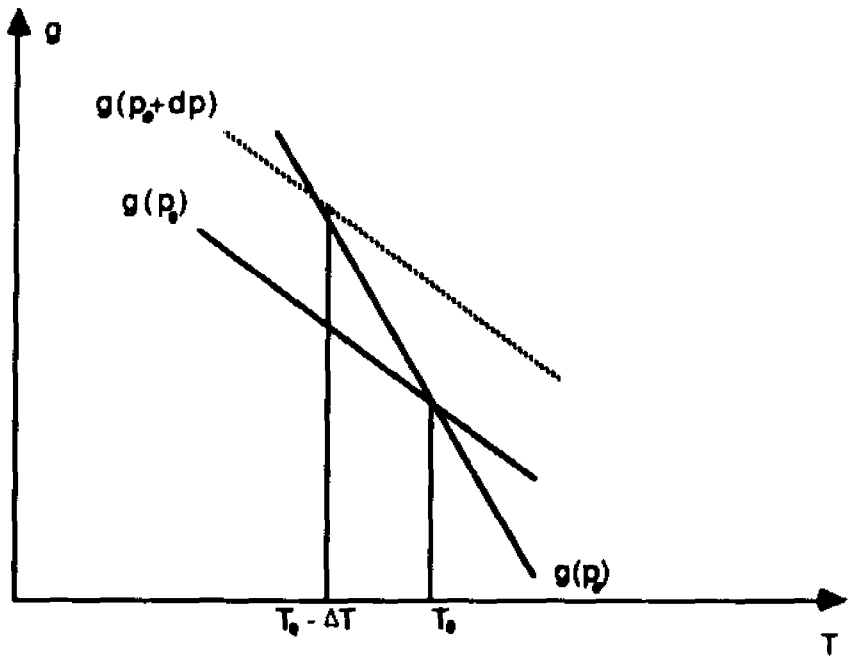
At equilibrium, equating g_s and g_l, we get $s_l\Delta T = s_s\Delta T + v_s\gamma\kappa$, hence

$$\Delta T = \frac{v_s}{s_l - s_s}\gamma\kappa = \frac{T_m}{L}\gamma\kappa \tag{A.6}$$

Thus the interface temperature $T_{interface} = T_m - \Delta T = T_m(1 - \gamma\kappa/L)$, which is the sought-after relation.



(a) Force diagram for an elemental surface area under surface tension



(b) Gibbs free energy as a function of temperature for a curved solid in equilibrium with its melt

Fig. 18

APPENDIX B

Derivation of $\varepsilon=(n^2-1)\beta$

The anisotropy in surface tension (or surface energy) is written as

$$\gamma = \gamma_0 [1 - \varepsilon \cos(n\theta)] \tag{B.1}$$

where n is the order of rotation symmetry in the (2-dimensional) solid. The anisotropy in the observed shape of the crystal is, however, not characterized by ε because the condition for equilibrium is that the temperature (rather than surface energy) everywhere on the interface is uniform. If the shape anisotropy is written as

$$r(\theta) = r_0 [1 + \beta \cos(n\theta)] \tag{B.2}$$

then we can compute the curvature everywhere on the interface. From standard calculus, the expression for curvature in polar coordinates is

$$\kappa = \frac{2\left(\frac{dr}{d\theta}\right)^2 - r\left(\frac{d^2r}{d\theta^2}\right) + r^2}{\left[\left(\frac{dr}{d\theta}\right)^2 + r^2\right]^{3/2}} \tag{B.3}$$

For interface shape $r(\theta)$ as written above,

$$\frac{dr}{d\theta} = -r_0 \beta n \sin(n\theta) \sim O(\beta)$$
$$\frac{d^2r}{d\theta^2} = -r_0 \beta n^2 \cos(n\theta) \sim O(\beta)$$

Since β is the smallness parameter, we only need to keep terms to order β. The curvature expression thus becomes

$$\kappa = \frac{-r\left(\frac{d^2 r}{d\theta^2}\right) + r^2}{r^3} = -\frac{1}{r^2}\left(\frac{d^2 r}{d\theta^2}\right) + \frac{1}{r}$$

$$= \frac{\beta n^2 \cos(n\theta)}{r_0[1 + \beta \cos(n\theta)]^2} + \frac{1}{r_0[1 + \beta \cos(n\theta)]}$$

$$= \frac{1}{r_0} + \frac{1}{r_0}\beta(n^2 - 1)\cos(n\theta) \qquad (B.4)$$

Putting this in the Gibbs-Thomson relation, $T_{interface}=T_m(1-\gamma\kappa/L)$, we get

$$T_{interface} = T_m\left\{1 - \frac{\gamma_0}{L}[1 - \epsilon \cos(n\theta)]\left[\frac{1}{r_0} + \frac{1}{r_0}\beta(n^2 - 1)\cos(n\theta)\right]\right\} \qquad (B.5)$$

Since β and ϵ are both small terms, we keep terms to order β or ϵ

$$T_{interface} = T_m\left\{\left(1 - \frac{\gamma_0}{r_0 L}\right) + \frac{\gamma_0}{r_0 L}[\beta(n^2 - 1) - \epsilon]\cos(n\theta)\right\} \qquad (B.6)$$

Now by requiring that, at equilibrium, the interface temperature must be isothermal, independent of orientation, we see that the second term in the curly bracket must vanish, hence

$$\epsilon = (n^2 - 1)\beta \qquad (B.7)$$

APPENDIX C

IMAGE ACQUISITION AND ANALYSIS

This manual spells out some of the details of the image analysis hardware and software in operation as of July 8, 1988. It is assumed that you know the rudiments of Assembly Language programming and operation of the PDP11 at monitor level. In order to understand the overall system completely, there is no substitute to reading the relevent manufacturers' manuals.

A. IMAGE ACQUISITION BOARDS

Reference: IP-512 Technical Manual Q-bus version (1985)
(There are 3 copies in the lab. The most complete one is in the binder labeled 'copy #1 with updates'.)

(1) Hardware

Three boards were purchased from Imaging Technology Inc.

AP-512: analog processor (A/D converter for video signals)
FB-512: frame buffer capable of storing one digitized frame
HF-512: histogram/feature extraction module

The configuration of the boards on the backplane of the PDP11/73 can be found on page 8-6 and Appendix A of the Technical Manual. ITI supplies special input/output cables.

The functions of the boards are controlled by a set of registers accessible as ordinary computer addresses. Programming the boards is simply a matter of loading 0's and 1's into these registers. **The factory-preset base addresses have been retained on our boards.** They are:

AP-512: 770020 (octal)
FB-512: 770000
HF-512: 770100

Consult IP-512 Technical Manual for details concerning the control registers.

The two modes of operation (camera input with composite video signal or VCR input with separate synchronization signals) require different jumper configurations on

the AP-512 board (see page 4-21 on the ITI Technical Manual). For camera input, jumpers J14A and J20B-C are 'out'. For VCR input, these two jumpers are 'in'. All other jumpers on AP-512 remain unchanged. For these two modes of operation, different input cables have to be used.

Now a few words about video image digitization (see page 2-3 of the ITI Technical Manual for more details). Standard (RS-170) composite video signal ranges from +0.143 V (black) to +0.714 V (white). An 8-bit digitizing board (AP-512) divides this range into 2^8=256 'gray levels' via what is called a 'look-up table' (LUT). Ordinarily the LUT is just a linear ramp with 0.143 V = Gray Level 0 and 0.714 V = Gray Level 256, but LUT can be programmed to any desired correspondence of voltage to digital intensity. This flexibility is used to our advantage in a rudimentary image enhancement routine called, for want of a better name, 'stretching'. Imagine that your video image consists mostly of intensities in a narrow range, say from 0.4 to 0.5 V. In the unprogramed LUT, these will be converted to gray levels (roughly) 150 to 200. This means that only a small part of the digitization range is being utilized. Now if you would program the LUT in such a way that 0.4 V= gray level 0 and 0.5 V= gray level 256 (with linear interpolation in between), the contrasts in the image will be greatly enhanced. This is what is done when you use the command S (for 'stretch') in PREP.FOR.

(2) Software

In order to understand the logical flow of any of the image-acquisition programs, it is necessay to understand the timing sequence with which the boards operate. On this point, alas, the ITI Technincal Manual is sadly inadequate. It is described on pages 5-16 and 8-21 of the Manual but reading these pages will just get you more confused. I shall undertake to make it more clear but unless you are going to write a program or modify an existing one, you should skip this section for now. Please refer to Fig. 19 constantly while reading the following paragraphs.

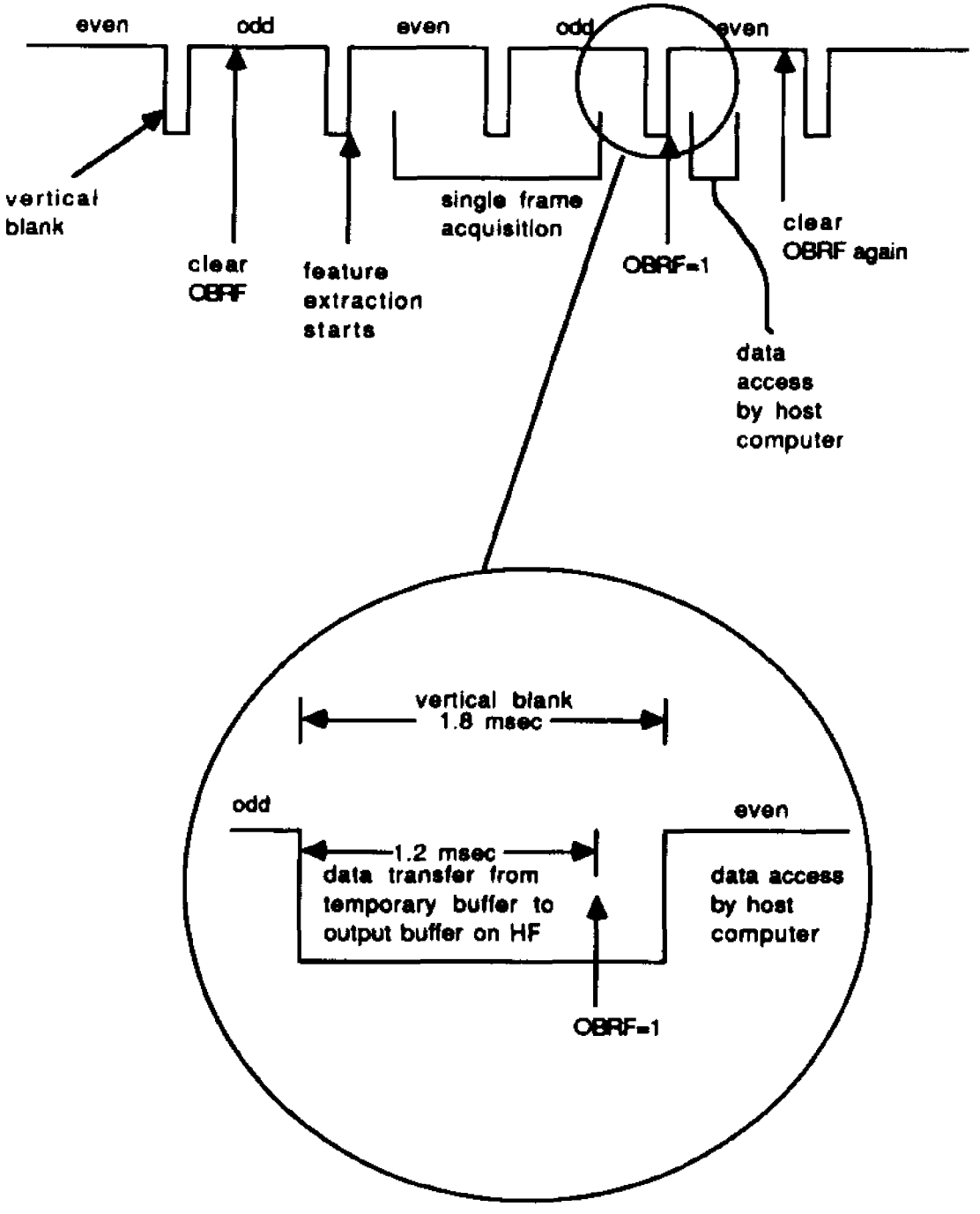


Fig. 19 Timing sequence of the Feature Extraction Board

(a) 'Composite video' scans even lines first and then odd lines. The even and odd 'fields' are then interlaced to form a 'frame'. The fields are done at a rate of 60/sec (thus the frames are 30/sec). The beginning of each field is marked by a 'vertical blank' of 1.8 msec duration. On the FB-512 board, there is a single bit in a register that keeps track whether the video is now in an even or odd field. To confuse the programmer, it is made in such a way that 1=even field, 0=odd field.

(b) The feature extraction operation is initiated by clearing one bit in a control register of the HF-512 board. It is known as the 'OBRF' bit for 'Output BuffeR Full'. When it is set equal to 0, feature extraction begins on the next even field and it remains 0 during the acquisition period. It is automatically set to 1 when either one of two conditions is satisfied: (i) the output buffer of the HF board (capacity=4096 pixels) is full, or (ii) end of the user specified aquistion period (it could be either one field or one frame).

(c) Feature extraction must start in an even field. During the acquisition period, the acquired features (x-y pairs) are stored in a temporary buffer (not accessible to user). **At the beginning of the next even vertical blank**, data is transferred to the output buffer of the HF board. Note that this transfer always takes place after the end of an odd field. This means that if you start feature extraction in an even field (as you must) and specify the acquisition period to be one field, you still have to wait through the odd field for your data to show up in the output buffer.

(d) The data transfer (to the output buffer of the HF board) starts at the falling edge of the even vertical blank and is completed in 1.2 msec. Remember that the vertical blank itself is 1.8 msec, so there is no 'spill over' into the next field. At the end of this data transfer, OBRF is automatically set to 1.

(c) After the OBRF has been set, the host computer can access the output buffer. Mere access takes only a fraction of a field. There might be some spill-over into the next field, but as long as you clear the OBRF within the next even field, data from the

spilled-over field will not be lost (remember that there is a temporary buffer working while you access the output buffer).

The above is the basic timing sequence. Because of some 'peculiarities' which the manufacturer cannot explain, the data does not appear in the output buffer immediately after the acquired frame but is delayed by one frame. In reality, one therefore can only do feature extraction every other frame. Furthermore, in all of our programs, the host computer does not merely access the acquired data but also does some computation as numbers are plucked out of the output buffer. This requires time and therefore means missing some more frames. All told, the XXI.MAC program and its sisters in the real-time-acquisition mode acquire a frame and skips five frames. This translates to a rate of 5 acquired frames per second. The SYNC-LOCK was purchased to get around this problem (and also the problem of limited computer memory) because it allows image acquisition from VCR which can freeze frame and advance single frames.

B. THE PDP11/73 COMPUTER

Reference: RT-11 Software Support Manual (DEC, April 1983)
Chapter 4 on Extended Memory

RT-11 Programmer's Reference Manual (DEC, 1984)
Chapter 2 on Program Requests

The PDP-11/73 is a 22-bit machine, which means in principle it could have 2^{22}=4 M of memory (our unit actually has 256K of memory). The MACRO-11 Assembly Language only allows 16-bit addressing, thus 2^{16}=64 K of memory is directly addressable by the programmer. In the jargon of computerniks, the 64 K is known as 'virtual address space' and the actual 256 K is called 'physical address space'. Now the goal is to make use of all that extra memory that is physically available but cannot be addressed directly. The trick is to use a coded address rather than the address itself. Thank goodness the programmer does not really have to think about the coded address. A

group of program requests do most of the work for you. Program requests are simply software supplied by the manufacturer. Some of the better known program requests are .PRINT, . TTYOUT, .EXIT etc. Among the lesser known program requests are the ones that allow you to access Extended Memory. These program requests are described in Chapter 2 of the RT-11 Programmer's Reference Manual with some good examples of their usage. To understand these example, you need to know something about how the RT11 (physical) memory is organized. I shall sketch very loosely (and in half-truth) how Extended Memory works. For the whole truth, you should consult Chapter 4 of the Software Support Manual.

As indicated in Fig. 20, the virtual address space is divided into 8 'pages' of 8 Kbyte each. Page 0 is taken up by system programs to handle errors etc. and is therefore unavailable to the user. The remaining space can be either directly addressed or 'mapped' onto Extended Memory. The program requests mapping virtual memory to physical memory use the page as the basic unit. A page in virtual memory is mapped to a 'region' in physical memory. The region can be from 8K (1 page) to 192 K (24 pages) in size. So imagine the region in Extended Memory being accessed one page at a time with a sliding 'window' up and down the region. In using program requests to access Extended Memory, you specify the page (in virtual address) which will be used as the sliding window, the starting address of the region in physical memory, the size of the region etc., then the rest is done automatically for you.

As of this writng, all of our image acquisition programs make use of the Extended Memory in the manner described above. Although our PDP11/73 is equipped with 256 K byte of memory, only one region (192 K byte) is used because I have not been able to get more than one region to work at the same time. For a program like XXI.MAC, 192 K of memory allows the storage of 23 frames of data. At 5 frames/sec, this translates to a mere 5 sec of run time, bearly adequate for our purpose. The SYNC-LOCK, working in conjunction with the VCR, is meant to get arount this problem.

Programs that make use of Extended Memory must be run under the RT11XM operating system. As of this writing, the PDP11 boots itself on the RT11FB operating system. After the system has been booted up, just issue the command 'BOOT RT11XM' to get into the Extended Memory operating system. Of the active programs listed in the next section, some require RT11XM and others can be run on RT11FB:

XX1X, XXMAX, XXI, SS : RT11XM

SYNC, SYNCXY, SYNXYI: RT11FB

PREP, XXIDY, BINRY, BINRI: FORTRAN programs

RT11XM is downward compatible with RT11FB, this means that all programs, Extended Memory or not, can run under the RT11XM operating system.

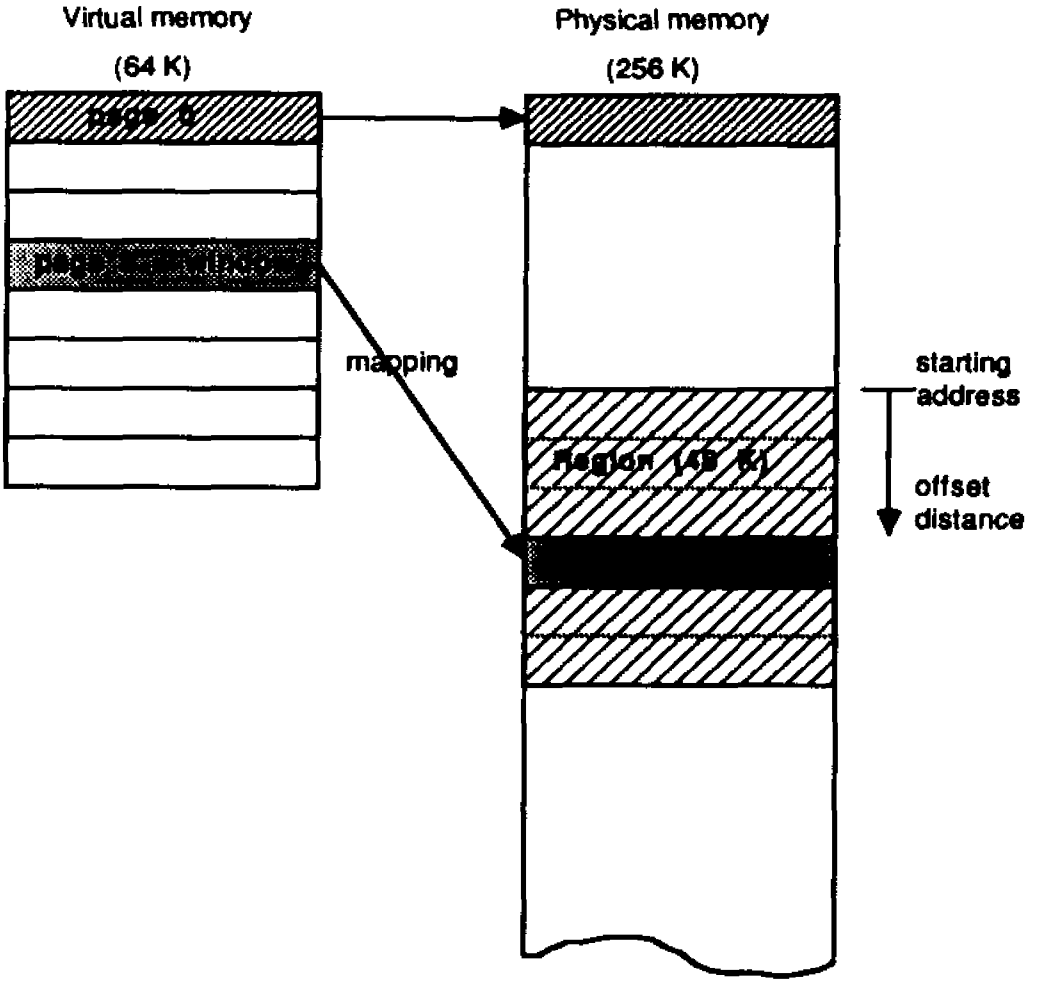


Fig. 20 Memory organization of PDP11/73

An example of a dynamic window of 8Kbyte in virtual memory mapped to a region of 48 Kbyte in physical memory.

APPENDIX D

CURRENTLY ACTIVE PROGRAMS AND THEIR FUNCTIONS

PREP Prepares the video picture for feature extraction. This includes some rudimentary image enhancement, setting the threshold intensity etc. This program must be used before any of the following image acqusition routines can be run.

XX1X Stores the coordinate of the first pixel in each scan-line that has intensity value above the threshold.
Note: The nemonics is that **XX**... refers to 'feature e**X**traction with 'E**X**tended Memory'.

XXMAX Stores the coordinate of the maximum intensity pixel in each scan line.

XXI Here **I**=Intensity. This is a feature extraction program (with Extended Memory) which stores the x-y coordinates and the corresponding intesity values. This is the program with which much of the data in this thesis was obtained.

SS 'Single Shot'. Slight modification of **XXI** to allow the program to prompt the user after acquiring each frame. No additional frame is acquired until the user gives the proper input. Like **XXI**, it store x-y coordinates and corresponding intensities. This program is meant to be used with the Sync-Lock.

SYNC Essentially the same with **SS** but indicates the number of frames acquired. Stores x-y coordinates with the corresponding intensities. This program does not require Extended Memory on the PDP11. Intended for the Sync-Lock.

SYNCXY Same as **SYNC** but does <u>not</u> store intensity values. Intended for the Sync-Lock. Does not require Extended Memory on PDP11.

SYNXYI Essentially the same as **SYNC** but stores data in a different format. Store x-y coordinates and corresponding intensities. Intended for the Sync-Lock and does not require Extended Memory on the PDP11.

All the image acquisition programs above store the extracted data in a binary format on the Winchester disk of the PDP11. To make it more convenient for data analysis, they are then converted to ASCII format by the following three programs. See the flow chart (Fig. 21) for usage.

XXIDY Converts data extracted by **XXI**, **SS** and **SYNC**.

BINRY Converts data extracted by **SYNCXY**, **XX1X**, **XXMAX**

BINRI Converts data extracted by **SYNXYI**.

The data analysis programs on VAX are all written in FORTRAN 77. See the comment statements in the programs for detailed descriptions of their functions.

interp.f Interpolation by parabola of the interface position. Needs data acquired by **XXI, SS** or **SYNC.**

rotat.f Rotate interface to get rid of 'DC ramp'.

splyn.f Least-square cubic B-spline fit of interface coordinates.

laxft.f 'Continuous Fourier Transfor' by Lax and Agrawal.

lspec.f FFT routine. Input data (x,y) must have equally spaced x.

spect.f Consolidation of the 5 program above. Can do either FFT or continuous Fourier Transform upon user input. Input data file contains raw data like the input to **interp.f.**

enavsp.f 'ENsemble AVeraged SPectrum'. Breaks up the interface into pieces, performs Fourier analysis (FFT or continuous) on each piece and average them. Input is raw data file like the input to **interp.f.**

enavspwk.f 'ENsemble AVeraged SPectrum $\omega(k)$'. Starting from a SEQUENCE of raw data files, deduce the growth rates $\omega(k)$ from the ensemble averaged spectra. Input data files are a sequence of raw data files like those to **interp.f.** Note that the time interval between files (each file containing interface coordinates from one frame) is assumed to be 0.2 sec. This is appropriate for data acquired by **XXI** because it acquires 5 frames per second. If any other time interval is used, you must change this constant in **enavspwk.f.**

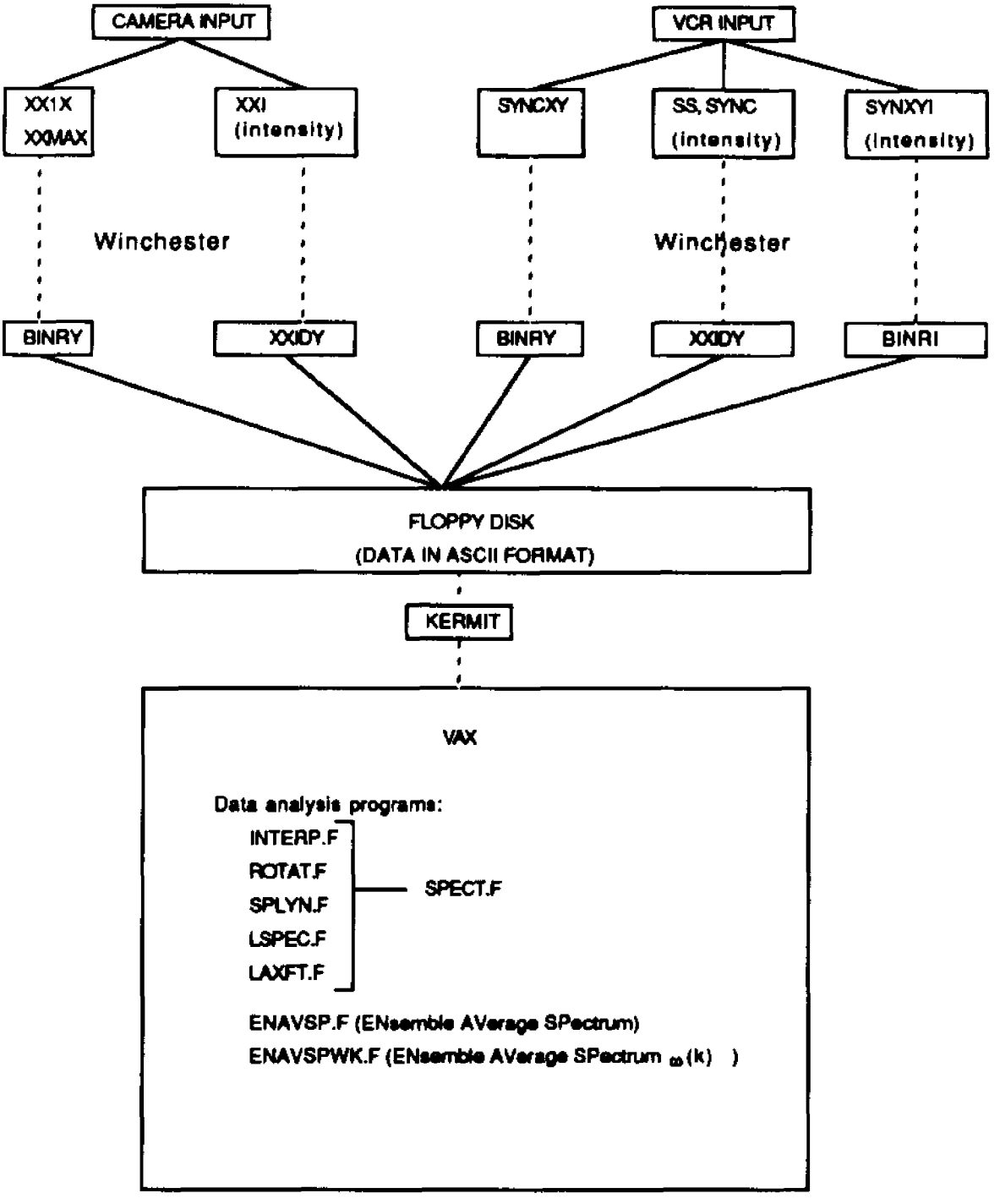


Fig. 21 Flow chart of usage for the feature extraction programs

```
C  PROGRAM BINRI.FOR
C
C  THIS PROGRAM REARRANGES THE NUMBERS AQUIRED BY THE FEATURE
C  EXTRACT PROGRAM "SYNXYI.MAC" AS FOLLOWS:
C       1. READS BINARY NUMBERS FROM DATA FILE.  'FEXED.DAT' IS
C          THE PRESUMED FILE NAME.  IF DATA FILE NAME IS DIFF-
C          FERENT, YOU WILL HAVE TO CHANGE LINE #2 OF THIS
C          PROGRAM.  IT IS EASIER TO MAKE YOUR DATA FILE NAME
C          CONFORM TO 'FEXED.DAT'.
C       2. PUTS THE NUMBERS IN FORTRAN FORMAT AS
C                       X  Y   I(X,Y)
C                       X  Y   I(X,Y)
C                       :  :   :
C       3. BREAKS UP THE FRAMES INTO SEPARATE DATA FILES BY THE
C          NAMES 'FEX.NN' WHERE NN IS THE NN-TH FRAME.
C
C  SUBROUTINE CALLED:
C    THIS PROGRAM MAKES USE OF RT11 SYSTEM LIBRARY ROUTINES SUCH AS
C    'READW', 'IGETC', 'IFETCH'.  SO THE OBJECT CODE OF THIS
C    PROGRAM MUST BE LINKED WITH "SYSLIB.OBJ".
C
C  LAST REVISION:  21 OCTOBER, 1987
C                  HENRY CHOU
C
      INTEGER*2     BUFF(12288),INFILE(4),OUTFILE(12),CR
      DATA          INFILE/3RDK0,3RFEX,3RED ,3RDAT/
      TYPE *,' IF YOU HAVE MORE THATN 20 FRAMES OF DATA, '
      TYPE *,' YOU WILL NEED MORE THAN 1 FORMATTED FLOPPY DISK. '
      TYPE *,' '
      TYPE *,' HIT ANY NUMBER & <CR> TO CONTINUE '
      READ(7,*)CR
      INCHAN=IGETC()
      M=IFETCH(INFILE)
      L=LOOKUP(INCHAN,INFILE)
      IBLK=0
        DO 50 I=1,100
        N=IREADW(8192,BUFF,IBLK,INCHAN)
        IF(N .LT. 0)GOTO 60
        IF(I .GE.10)GOTO 10
        ENCODE(10,5,OUTFILE) I
5       FORMAT('DY:FEX.',I1)
        GOTO 21
10      ENCODE(10,20,OUTFILE) I
20      FORMAT('DY:FEX.',I2)
21      OPEN(UNIT=1,NAME=OUTFILE)
          DO 30 J=4,12288,3
          J1=J+1
          J2=J+2
          IF((BUFF(J) .EQ. 0) .AND. (BUFF(J1) .EQ. 0))GOTO 40
          WRITE(1,25)BUFF(J),BUFF(J1),BUFF(J2)
25        FORMAT(I3,1X,I3,1X,I3)
30        CONTINUE
40      CLOSE(UNIT=1)
```

```
        IBLK=IBLK+48
        IF(IBLK .LE. 960)GOTO 50
        TYPE *,' PLEASE PUT A NEW DISK IN DY:'
        TYPE *,' THEN HIT ANY NUMBER & <CR> TO CONTINUE '
        READ(7,*)CR
50      CONTINUE

60      CALL ICLOSE(INCHAN)
        J=ICLOSE()
        CALL IFREEC(INCHAN)
        STOP
        END
```

```
C  PROGRAM BINRY.FOR
C
C  THIS PROGRAM REARRANGES THE NUMBERS AQUIRED BY THE FEATURE
C  EXTRACT PROGRAM "SYNCXY.MAC" AS FOLLOWS:
C       1. READS BINARY NUMBERS FROM DATA FILE.  'FEXED.DAT' IS
C          THE PRESUMED FILE NAME.  IF DATA FILE NAME IS DIFF-
C          FERENT, YOU WILL HAVE TO CHANGE LINE #2 OF THIS
C          PROGRAM.  IT IS EASIER TO MAKE YOUR DATA FILE NAME
C          CONFORM TO 'FEXED.DAT'.
C       2. PUTS THE NUMBERS IN FORTRAN FORMAT AS X-Y PAIRS.
C       3. BREAKS UP THE FRAMES INTO SEPARATE DATA FILES BY THE
C          NAMES 'FEX.NN' WHERE NN IS THE NN-TH FRAME.
C
C  SUBROUTINE CALLED:
C    THIS PROGRAM MAKES USE OF RT11 SYSTEM LIBRARY ROUTINES SUCH AS
C    'READW', 'IGETC', 'IFETCH'.  SO THE OBJECT CODE OF THIS
C    PROGRAM MUST BE LINKED WITH "SYSLIB.OBJ".
C
C  LAST REVISION:  21 OCTOBER 1987
C                  HENRY CHOU
C
      INTEGER*2     BUFF(8192),INFILE(4),OUTFILE(12),CR
      DATA          INFILE/3RDK0,3RFEX,3RED ,3RDAT/
      TYPE *,' IF YOU HAVE MORE THAN 30 FRAMES OF DATA, '
      TYPE *,' YOU WILL NEED MORE THAN 1 FORMATTED FLOPPY DISK. '
      TYPE *,' '
      TYPE *,' HIT ANY NUMBER & <CR> TO CONTINUE '
      READ(7,*)CR
      INCHAN=IGETC()
      M=IFETCH(INFILE)
      L=LOOKUP(INCHAN,INFILE)
      IBLK=0
        DO 50 I=1,100
        N=IREADW(8192,BUFF,IBLK,INCHAN)
        IF(N .LT. 0)GOTO 60
        IF(I .GE.10)GOTO 10
        ENCODE(10,5,OUTFILE) I
5       FORMAT('DY:FEX.',I1)
        GOTO 21
10      ENCODE(10,20,OUTFILE) I
20      FORMAT('DY:FEX.',I2)
21      OPEN(UNIT=1,NAME=OUTFILE,TYPE='NEW')
          DO 30 J=3,8192,2
          J1=J+1
          IF((BUFF(J) .EQ. 0) .AND. (BUFF(J1) .EQ. 0))GOTO 40
          WRITE(1,25)BUFF(J),BUFF(J1)
25        FORMAT(I3,1X,I3)
30        CONTINUE
40      CLOSE(UNIT=1)
        IBLK=IBLK+32
      IF(IBLK .LE. 960)GOTO 50
```

```
      TYPE *,' PLEASE PUT A NEW DISK IN DY: '
      TYPE *,' THEN HIT ANY NUMBER & <CR> TO CONTINUE '
      READ(7,*)CR
50     CONTINUE

60    CALL ICLOSE(INCHAN)
      J=ICLOSE()
      CALL IFREEC(INCHAN)
      STOP
      END
```

```
;       PROGRAM SYNC
; THIS PROGRAM IS A DESCENDENT OF XXI.MAC.  UNLIKE XXI.MAC,
; THIS PROGRAM IS WRITTEN TO WORK
; WITH THE SYNC-LOCK AND THE VCR.  FEATURE EXTRACTION IS DONE
; ONE FRAME AT A TIME AT THE
; COMMAND OF THE OPERATOR.  THE DATA ACQUIRED FROM EACH FRAME IS
; DIRECTLY WRITTEN INTO A DATA
; FILE ON THE WINCHESTER, THUS THIS PROGRAM DOES NOT MAKE USE OF
; EXTENDED MEMORY.  THE PORTION
; OF THE PROGRAM THAT CONTROLS THE IMAGE PROCESSING BOARDS IS
; ESSENTIALLY THE SAME AS BEFORE,
; NAMELY FEATURE EXTRACTION IS DONE ON A FROZEN FRAME.
; THE EXTRACTED COORDINATES AND THEIR
; CORRESPONDING INTENSITY VALUES ARE STORED.
;
; SINCE THIS PROGRAM DOES NOT MAKE USE OF EXTENDED MEMORY,
; IN PRINCIPLE YOU COULD KEEP
; ACQUIRING FEATRUES UNTIL YOUR DATA FILE FILLS UP THE
; WINCHESTER DISK ( THIS TRANSLATES TO
; APPROXIMATELY 1200 FRAMES ).  IN PRACTICE, THERE ARE
; TWO LIMITATIONS: (1) YOUR PATIENCE
; ( REMEMBER THAT YOU WILL HAVE TO ANALYZE ALL THIS RAW DATA ),
; AND (2) THE RAW DATA WILL
; HAVE TO BE TRANSFERED TO FLOPPY DISK IN FORTRAN FORMAT
; (THIS IS ACCOMPLISHED BY XXI2DY.FOR).
; THE CAPACITY OF EACH FLOPPY DISK ALLOWS 23 FRAMES OF DATA.
;  IF YOU ACQUIRE TOO MANY FRAMES,
; YOU WILL HAVE MANY FLOPPY DISKS TO HANDLE.  THE RULE OF THUMB
; IS THAT YOU SHOULD PLAN TO
; ACQUIRE ROUGHLY 20 FRAMES.  FOR A LONG RUN, PLAN 40 FRAMES.
;
;       SUBROUTINES CALLED:     OCDEC.MAC
;
;       NOTES ON USAGE:     1.  THIS PROGRAM RELIES ON 'PREP.FOR' TO SET UP WINDOW,
;                               THRESHOLD, ETC. FOR IT.
;                           2.  AFTER A (BINARY) DATA FILE HAS BEEN MADE ON ER,
;                               WINCHESTER, THE FORTRAN PROGRAM 'XXIDY.FOR'
;                               IS USED TO CONVERT THE NUMBERS TO FORTRAN
;                               FORMAT AND TO SAVE THEM ON THE FLOPPY DISK.
;
;
;       LAST REVISION:      18 OCTOBER, 1987
;                           HENRY CHOU
;
            .TITLE SYNC

SYNC:  .MCALL .PRINT,.CSIGEN,.WRITW,.CLOSE,.EXIT,.TTYIN,.TTYOUT

       CLRB   @#170023              ;CONTROL REGISTER OF AP BOARD
       MOV    #170100,R1            ;R1 = HF BOARD BASE ADDRESS
       MOV    #140000,@#170010      ;CONTINUOUS ACQUIRE
```

```
;********** PROMPT THE OPERATOR FOR INPUT **********

        .PRINT  #FILE                   ;ASK FOR DATA FILE NAME
        .CSIGEN         #ENDCRE,#EXT,#0         ;LOAD DEVICE HANDLER
        MOV     #1,FCONT                ;FRAME COUNTER STARTS FROM 1
FRM:    MOV     FCONT,R3                ;MOVE FRAME COUNTER TO R3
        JSR     PC,OCDEC                ;PRINT OUT FRAME #
        .PRINT  #PROMPT
LF:     .TTYIN
        CMP     #121,R0
        BEQ     CLS                     ;IF Q THEN CLOSE DATA FILE
        CMP     #12,R0                  ;TEST FOR <LF>
        BNE     LF
        CLR     R0

;**********[1] NOW START THE FEATURE EXTRACTION ON ONE FRAME **********

        BIS     #141000,(R1)            ;SINGLE-FRAME FEX
                                        ;TOGGLE LINE WINDOW MODE
1$:     BIT     #40,2(R1)               ;TEST EVEN/ODD FIELD
        BNE     1$                      ;IF ODD, WAIT
        MOV     #100,2(R1)              ;CLEAR OBRF IN THE EVEN FIELD
2$:     BIT     #20,2(R1)               ;CHECK FOR OBRF=1
        BEQ     2$

        BIC     #140000,@#170010        ;FREEZE FRAME

        MOV     #2,R4                   ;START OF CYCLE, R4 IS A COUNTER
EO:     BIT     #40,2(R1)               ;EVEN/ODD FIELD
        BNE     EO                      ;WAIT IF ODD FIELD
OBR0:   MOV     #100,2(R1)              ;CLEAR OBRF IN THE EVEN FIELD
OBR1:   BIT     #20,2(R1)               ;CHECK FOR OBRF=1
        BEQ     OBR1
        SOB     R4,OBR0

;**********[2] GET THE NUMBERS FROM MEMORY BUFFER**********

        MOV     #DATA,R2                ;R2=BASE ADDRS OF DATA STORAGE SPACE
        MOV     #1041,(R1)              ;OUTPUT BUFFER ACCESS, AUTO INC. 2
        CLR     6(R1)

        CLR     WCONT                   ;CLEAR COUNTER
        MOV     4(R1),R4                ;R4=FEATURE COUNT
        MOV     #600,@#170002           ;INITIALIZE Y VALUE IN FB BOARD
ACC:    MOV     10(R1),TEMPX            ;GET X COORDINATE
        MOV     10(R1),TEMPY            ;GET Y COORDINATE
        CMP     TEMPY,@#170002          ;IS THIS THE SAME Y SCAN AS LAST ?
        BEQ     X                       ;IF YES, DO NOT SAVE IT
```

```
        MOV   TEMPY,(R2)          ;IF NO, STORE THE NEW Y VALUE
        ADD   #1000.,(R2)+        ;TICK-MARK FOR Y VALUE
        INC   WCONT                     ;INCREMENT COUNTER
        MOV   TEMPY,@#170002            ;AND UPDATE THE Y COORD. IN FB BOARD
X:      MOV   TEMPX,@#170000            ;X COORD. IN FB BOARD
        MOV   TEMPX,(R2)+         ;STORE X COORDINATE IN STORAGE SPACE
        INC   WCONT                     ;INCREMENT COUNTER
        MOV   @#170016,(R2)+            ;STORE I(X,Y) IN STORAGE SPACE
        INC   WCONT                     ;INCREMENT COUNTER
        SOB   R4,ACC                    ;ACCESS NEXT PAIR OF EXTRACTED FEATURE

;**********[3] MOVE DATA TO WINCHESTER DISK **********

WR:     .WRITW      #RWBK,#0,#DATA,WCONT,BCONT
        ADD   #16.,BCONT          ;UPDATE THE BLOCK COUNTER
        INC   FCONT               ;UPDATE THE FRAME COUNTER

        BIS   #140000,@#170010    ;BACK TO CONTINUOUS ACQUIRE
        BR    FRM                 ;NEXT FRAME

;*************************************************************

CLS:    .CLOSE #0                 ;CLOSE THE FILE
        .EXIT

EMTBK:      .BLKW 2
RWBK: .BLKW 6
EXT:    .WORD 0,0,0,0
TEMPX:      .BLKW 1
TEMPY:      .BLKW 1
FCONT:      .BLKW 1                     ;FRAME COUNTER
BCONT:      .BLKW 1                         ;BLOCK COUNTER FOR .WRITW
WCONT:      .BLKW 1                         ;WORD COUNTER FOR .WRITW
DATA: .BLKW 4096.

FILE:   .ASCIZ / NAME OF DATA FILE ?/
PROMPT:     .ASCII /    HIT <CR> TO ACQUIRE ANOTHER FRAME, /
QUIT:   .ASCIZ / OR ENTER Q TO QUIT /
ENDCRE      =.
        .END SYNC
```

```
;       PROGRAM SYNCXY
;
; THIS PROGRAM IS A MODIFICATION OF SYNC.MAC.  THE ONLY DIFFERENCE IS THAT
; SYNC.MAC ACQUIRES THE X-Y PAIRS AS WELL AS THE CORRESPONDING INTENSITY
; VALUES AND STORES THEM INTO A DATA FILE ON WINCHESTER.  THIS PROGRAM
; STORES ONLY THE X-Y PAIRS AND NOT THE INTENSITY VALUES.  SEE COMMENTS
; IN SYNC.MAC FOR MORE INFORMATION.
;
;       LAST REVISION:      21 OCTOBER, 1987
;                           HENRY CHOU
;
            .TITLE SYNCXY

SYNCXY:     .MCALL .PRINT,.CSIGEN,.WRITW,.CLOSE,.EXIT,.TTYIN,.TTYOUT

        CLRB    @#170023            ;CONTROL REGISTER OF AP BOARD
        MOV     #170100,R1          ;R1 = HF BOARD BASE ADDRESS
        MOV     #140000,@#170010    ;CONTINUOUS ACQUIRE


;********** PROMPT THE OPERATOR FOR INPUT **********

        .PRINT  #FILE               ;ASK FOR DATA FILE NAME
        .CSIGEN     #ENDCRE,#EXT,#0         ;LOAD DEVICE HANDLER

        MOV     #1,FCONT            ;FRAME COUNTER STARTS FROM 1
FRM:    MOV     FCONT,R3            ;MOVE FRAME COUNTER TO R3
        JSR     PC,OCDEC            ;PRINT OUT FRAME #
        .PRINT  #PROMPT
LF:     .TTYIN
        CMP     #121,R0
        BEQ     CLS                 ;IF Q THEN CLOSE DATA FILE
        CMP     #12,R0              ;TEST FOR <LF>
        BNE     LF
        CLR     R0

;**********[1] NOW START THE FEATURE EXTRACTION ON ONE FRAME **********


        BIS     #141000,(R1)        ;SINGLE-FRAME FEX
                                    ;TOGGLE LINE WINDOW MODE
1$:     BIT     #40,2(R1)           ;TEST EVEN/ODD FIELD
        BNE     1$                  ;IF ODD, WAIT
        MOV     #100,2(R1)          ;CLEAR OBRF IN THE EVEN FIELD
2$:     BIT     #20,2(R1)           ;CHECK FOR OBRF=1
        BEQ     2$

        BIC     #140000,@#170010    ;FREEZE FRAME
```

```
        MOV   #2,R4             ;START OF CYCLE, R4 IS A COUNTER
EO:     BIT   #40,2(R1)         ;EVEN/ODD FIELD
        BNE   EO                ;WAIT IF ODD FIELD
OBR0:   MOV   #100,2(R1)        ;CLEAR OBRF IN THE EVEN FIELD
OBR1:   BIT   #20,2(R1)         ;CHECK FOR OBRF=1
        BEQ   OBR1
        SOB   R4,OBR0

;*********[2] GET THE NUMBERS FROM MEMORY BUFFER**********************

        MOV   #DATA,R2          ;R2=BASE ADDRS OF DATA STORAGE SPACE
        MOV   #1041,(R1)        ;OUTPUT BUFFER ACCESS, AUTO INC. 2
        CLR   6(R1)

        MOV   4(R1),WCONT
        ASL   WCONT                     ;WCONT=WORD COUNT (X-Y PAIRS)
        MOV   4(R1),R4          ;R4=FEATURE COUNT
ACC:    MOV   10(R1),(R2)+      ;GET X COORDINATE
        MOV   10(R1),(R2)+      ;GET Y COORDINATE
        SOB   R4,ACC                    ;ACCESS NEXT PAIR OF EXTRACTED FEATURE

;*************[3] MOVE DATA TO WINCHESTER DISK ********************

WR:     .WRITW      #RWBK,#0,#DATA,WCONT,BCONT
        ADD   #32.,BCONT        ;UPDATE THE BLOCK COUNTER
        INC   FCONT             ;UPDATE THE FRAME COUNTER

        BIS   #140000,@#170010  ;BACK TO CONTINUOUS ACQUIRE
        BR    FRM               ;NEXT FRAME

;*************************************************************************

CLS:    .CLOSE #0               ;CLOSE THE FILE
        .EXIT


EMTBK:      .BLKW 2
RWBK: .BLKW 6
EXT:    .WORD 0,0,0,0
FCONT:      .BLKW 1                     ;FRAME COUNTER
BCONT:      .BLKW 1                         ;BLOCK COUNTER FOR .WRITW
WCONT:      .BLKW 1                         ;WORD COUNTER FOR .WRITW
DATA: .BLKW 8192.

FILE:   .ASCIZ / NAME OF DATA FILE ?/
PROMPT:     .ASCII /    HIT <CR> TO ACQUIRE ANOTHER FRAME, /
QUIT:   .ASCIZ / OR ENTER Q TO QUIT /
ENDCRE      =.
        .END SYNCXY
```

```
;       PROGRAM SYNXYI
; THE THREE PROGRAMS SYNC.MAC, SYNCXY.MAC AND SYNXYI.MAC ARE SISTERS
; OF EACH OTHER.  THEY PERFORM FEATURE EXTRACTION IN THE SAME MANNER BUT
; STORE DATA DIFFERENTLY.  THEIR DIFFERENCES ARE DELINEATED BELOW:
;
;       1.  SYNC.MAC -- PICKS UP X-Y COORDINATES AND THE CORRESPONDING
;           INTENSITY VALUES.  DATA IS STORED SEQUENTIALLY IN THE FORM
;
;               Y1  X1  I(X1,Y1)  X2  I(X2,Y1)  X3  I(X3,Y1) _
;               Y2  X1  I(X1,Y2)  X3  I(X2,Y2)  X3  I(X3,Y1) _
;
;           NAMELY THE Y COORDINATE (SCAN LINE) IS STORED ONLY ONCE, AND
;           THEN ALL THE FEATURES AND INTENSITIES ON THAT SCAN LINE FOLLOW.
;           THIS IS DONE TO SAVE MEMORY SPACE.  IF YOU THEN USE THE FORTRAN
;           PROGRAM 'XXIDY.FOR' TO READ THE RAW DATA FILE, YOU GET ON THE
;           FLOPPY DISK A FILE IN THE FORM
;
;               Y1      0
;               X1      I(X1,Y1)
;               X2      I(X2,Y1)
;               :       :
;               :       :
;               Y2      0
;               X1      I(X1,Y2)
;               X2      I(X2,Y2)
;               :       :
;               :       :
;
;           NOTE:       SYNC.MAC ALLOWS ONLY 4096 WORDS OF MEMORY PER FRAME
;                       TO STORE THE COORDINATES AS WELL AS INTENSITIES.
;                       THIS TRANSLATES TO ROUGHLY 1500 FEATURES (PIXELS)
;                       PER FRAME.  IF YOUR PICTURE HAS MORE FEATURES PER
;                       FRAME THAN 1500, THE PROGRAM PROBABLY WILL CRASH.
;                       IN THAT CASE USE SYNXYI.MAC, SEE BELOW.
;
;       2.  SYNCXY.MAC -- PICKS UP X-Y COORDINATES BUT DOES NOT STORE INTEN-
;           SITIES.  THE DATA IS STORED SEQUENTIALLY IN THE FORM
;
;               X1  Y1  X2  Y2  X3  Y3 _
;
;           IF YOU THEN USE THE PROGRAM 'BINRY.FOR' TO READ THE RAW DATA FILE,
;           YOU GET A DATA FILE ON THE FLOPPY DISK IN THE FORM
;
;               X1      Y1
;               X2      Y2
;               X3      Y3
;               :       :
;
;           THIS PROGRAM ALLOWS 4096 X-Y PAIRS TO BE PICKED UP (THIS IS THE
;           MAXIMUM NUMBER OF FEATURES THE FEX BOARD IS ABLE TO DO PER FRAME.
;
```

```
;       3. SYNXYI.MAC -- PICKS UP X-Y COORDINATES AND STORES THE INTENSITY
;          VALUES AS WELL.  DATA IS STORED IN THE FORM
;
;              X1  Y1  I(X1,Y1)  X2  Y2  I(X2,Y2) ...
;
;          IF YOU THEN USE THE PROGRAM 'BINRI.FOR' TO READ THE RAW DATA FILE,
;          YOU GET A DATA FILE ON THE FLOPPY DISK IN THE FORM
;
;              X1      Y1      I(X1,Y1)
;              X2      Y2      I(X1,Y1)
;              :       :       :
;              :       :       :
;
;          THIS PROGRAM ALLOWS 4096 X-Y PAIRS AND THEIR INTENSITIES TO BE
;          PICKED UP AND STORED.  THIS IS THE MAXIMUM AMOUNT OF INFORMATION
;          ONE CAN GET WHEN USING THE FEATURE EXTRACTION AND FRAME BUFFER
;          BOARDS.
;
;       SUBROUTINES CALLED:     OCDEC.MAC
;
;       NOTES ON USAGE:   1.  THIS PROGRAM RELIES ON 'PREP.FOR' TO SET UP WINDOW
;                             AND THRESHOLD ETC.
;                         2.  AFTER A (BINARY) DATA FILE HAS BEEN MADE ON THE
;                             WINCHESTER, THE FORTRAN PROGRAM 'BINRI.FOR' IS
;                             USED TO CONVERT THE NUMBERS TO FORTRAN FORMAT AND
;                             TO SAVE THEM ON THE FLOPPY DISK.
;
;       LAST REVISION:    21 OCTOBER, 1987
;                         HENRY CHOU
;
            .TITLE SYNXYI

SYNXYI:     .MCALL .PRINT,.CSIGEN,.WRITW,.CLOSE,.EXIT,.TTYIN

        CLRB    @#170023            ;CONTROL REGISTER OF AP BOARD
        MOV     #170100,R1          ;R1 = HF BOARD BASE ADDRESS
        MOV     #140000,@#170010    ;CONTINUOUS ACQUIRE


;*********** PROMPT THE OPERATOR FOR INPUT **********************

        .PRINT  #FILE               ;ASK FOR DATA FILE NAME
        .CSIGEN     #ENDCRE,#EXT,#0         ;LOAD DEVICE HANDLER

        MOV     #1,FCONT            ;FRAME COUNTER STARTS FROM 1
FRM:    MOV     FCONT,R3            ;MOVE FRAME COUNTER TO R3
        JSR     PC,OCDEC            ;PRINT OUT FRAME #

        .PRINT  #PROMPT
LF:     .TTYIN
```

```
        CMP   #121,R0
        BEQ   CLS               ;IF Q THEN CLOSE DATA FILE
        CMP   #12,R0            ;TEST FOR <LF>
        BNE   LF
        CLR   R0

;**********[1] NOW START THE FEATURE EXTRACTION ON ONE FRAME **********

        BIS   #141000,(R1)      ;SINGLE-FRAME FEX
                                ;TOGGLE LINE WINDOW MODE
1$:     BIT   #40,2(R1)         ;TEST EVEN/ODD FIELD
        BNE   1$                ;IF ODD, WAIT
        MOV   #100,2(R1)        ;CLEAR OBRF IN THE EVEN FIELD
2$:     BIT   #20,2(R1)         ;CHECK FOR OBRF=1
        BEQ   2$

        BIC   #140000,@#170010  ;FREEZE FRAME

        MOV   #2,R4             ;START OF CYCLE, R4 IS A COUNTER
EO:     BIT   #40,2(R1)         ;EVEN/ODD FIELD
        BNE   EO                ;WAIT IF ODD FIELD
OBR0:   MOV   #100,2(R1)        ;CLEAR OBRF IN THE EVEN FIELD
OBR1:   BIT   #20,2(R1)         ;CHECK FOR OBRF=1
        BEQ   OBR1
        SOB   R4,OBR0

;*********[2] GET THE NUMBERS FROM MEMORY BUFFER**********

        MOV   #DATA,R2          ;R2=BASE ADDRS OF DATA STORAGE SPACE
        MOV   #1041,(R1)        ;OUTPUT BUFFER ACCESS, AUTO INC. 2
        CLR   6(R1)

        MOV   4(R1),WCONT       ;4(R1)=FEATURE COUNT ON HF BOARD
        ASL   WCONT                     ;WCONT=NUMBER OF WORDS NEEDED TO
        ADD   4(R1),WCONT       ;     STORE X,Y AND I(X,Y)

        MOV   4(R1),R4          ;R4=FEATURE COUNT
ACC:    MOV   10(R1),(R2)       ;GET X COORDINATE
        MOV   (R2)+,@#170000            ;PUT X IN FB BOARD REGISTER
        MOV   10(R1),(R2)       ;GET Y COORDINATE
        MOV   (R2)+,@#170002            ;PUT Y IN FB BOARD REGISTER
        MOV   @#170016,(R2)+            ;STORE I(X,Y) IN STORAGE SPACE
        SOB   R4,ACC                    ;ACCESS NEXT PAIR OF EXTRACTED FEATURE

;**********[3] MOVE DATA TO WINCHESTER DISK **********

WR:     .WRITW      #RWBK,#0,#DATA,WCONT,BCONT
        ADD   #48.,BCONT        ;UPDATE THE BLOCK COUNTER
        INC   FCONT             ;UPDATE THE FRAME COUNTER
```

```
        BIS     #140000,@#170010    ;BACK TO CONTINUOUS ACQUIRE
        BR      FRM                 ;NEXT FRAME

;*********************************************************************

CLS:    .CLOSE #0                   ;CLOSE THE FILE
        .EXIT

EMTBK:          .BLKW 2
RWBK: .BLKW 6
EXT:    .WORD 0,0,0,0
FCONT:          .BLKW 1                     ;FRAME COUNTER
BCONT:          .BLKW 1                         ;BLOCK COUNTER FOR .WRITW
WCONT:          .BLKW 1                         ;WORD COUNTER FOR .WRITW
DATA: .BLKW 12288.

FILE:   .ASCIZ / NAME OF DATA FILE ?/
PROMPT:         .ASCII /    HIT <CR> TO ACQUIRE ANOTHER FRAME, /
QUIT:   .ASCIZ / OR ENTER Q TO QUIT /
ENDCRE          =.
        .END SYNXYI
```

```
        .TITLE XX1X
; THIS PROGRAM PERFORMS FEATURE EXTRACTION WHEREIN THE FIRST PIXEL IN EACH
; SCAN WHICH CROSSES THE VALID FEATURE THRESHOLD IS STORED IN EXTENDED
; MEMORY.  THIS PROGRAM RELIES ON 'PREP.FOR' TO SET UP WINDOW, VALID FEATURE
; INTENSITY ETC. FOR IT.
;
; THE PROGRAM PERFORMS ITS TASK IN THE FOLLOWING SEQUENCE:
;
;       1. SET UP EXTENDED MEMORY MAPPING
;       2. START FEATURE EXTRACTION (ON A FROZEN FRAME).
;          NOTE THE THRESHOLD INTENSITY ETC. HAVE BEEN SET BY RUNNING 'PREP.FOR
;       3. ACCESS THE FEATURES FROM THE FEX BOARD.  ONLY THE FIRST PIXEL
;          IN EACH SCAN LINE IS STORED.
;       4. OPEN A FILE ON WINCHESTER DISK TO STORE THE DATA.
;
; SUBROUTINES CALLED:  NONE
;
; LAST REVISION: 5 DECEMBER, 1986
;                HENRY CHOU
;

XX1X: .MCALL .PRINT,.CSIGEN,.WRITW,.CLOSE,.EXIT,.GMCX
       .MCALL .MAP,.UNMAP,.CRAW,.WDBBK,.ELAW,.CRRG,.RDBBK,.ELRG,.WDBDF,.RDBDF

       CLRB   @#170023          ;CONTROL REGISTER OF AP BOARD
       MOV    #170100,R1        ;R1 = HF BOARD BASE ADDRESS
       MOV    #140000,@#170010  ;CONTINUOUS ACQUIRE

;**************[1] SET UP EXTENDED MEMORY MAPPING**************

; DEFINE UNITS:
; 32.-UNIT = GROUP OF 32 WORDS = 2**5  WORDS
; BLOCK    =      256 WORDS = 2**8  WORDS
; PAGE     =     4096 WORDS = 2**12  WORDS = 128  32.-UNITS = 16 BLOCKS
; WINDOW   =          8192 WORDS = 2**13  WORDS = 256  32.-UNITS = 32 BLOCKS
; REGION   =    96256 WORDS =          =3008  32.-UNITS

       APR  =     2             ;PAR/PDR FOR BUFFER (VIRTUAL WINDOW)
       WINDW=     WDB+W.NBAS;WINDOW BASE ADDRESS
       WDSIZ=     1024.         ;BUFFER SIZE=1024. WORDS
       RGSIZ=     3008.         ;REGION SIZE=3008. 32.-UNITS
       WRNID=     WDB+W.NRID ;REGION ID ADDRESS

       .WDBDF
       .RDBDF

RGMAP:      .CRRG  #EMTBK,#RDB       ;CREATE REGION (96K WORDS)
       MOV    RDB,WRNID         ;MOVE REGION ID TO WINDOW
       .CRAW #EMTBK,#WDB              ;CREATE A WINDOW (1024 WORDS)
```

```
;**********[2] NOW START THE FEATURE EXTRACT**********************

FRM:  BIS    #141000,(R1)          ;SINGLE-FRAME FEX
                                   ;TOGGLE LINE WINDOW MODE
7$:   BIT    #40,2(R1)             ;TEST EVEN/ODD FIELD
      BNE    7$                    ;IF ODD, WAIT
      MOV    #100,2(R1)            ;CLEAR OBRF IN THE EVEN FIELD
8$:   BIT    #20,2(R1)             ;CHECK FOR OBRF=1
      BEQ    8$

      BIC    #140000,@#170010      ;FREEZE FRAME

      MOV    #2,R4                 ;START OF CYCLE, R4 IS A COUNTER
EO:   BIT    #40,2(R1)             ;EVEN/ODD FIELD
      BNE    EO                    ;WAIT IF ODD FIELD
OBR0: MOV    #100,2(R1)            ;CLEAR OBRF IN THE EVEN FIELD
OBR1: BIT    #20,2(R1)             ;CHECK FOR OBRF=1
      BEQ    OBR1
      SOB    R4,OBR0

;********[3] GET THE FEATURES FROM FEX BOARD BUFFER***************

      .MAP   #EMTBK,#WDB               ;MAP WINDOW TO REGION
                                   ;WITH OFFSET INTO REGION=OFSET
                                   ;AND LENGTH TO MAP = 1024 WORDS
      .GMCX  #EMTBK,#WDB               ;GET MAPPING STATUS

      MOV    WINDW,R2              ;R2=WINDOW BASE ADDRESS
AQ:   MOV    #1041,(R1)            ;OUTPUT BUFFER ACCESS, AUTO INC. 2
      CLR    6(R1)

      MOV    4(R1),R4              ;FEATURE COUNTS
      MOV    #600,LASTY            ;INITIAL VALUE OF LAST Y
X:    MOV    10(R1),TEMPX          ;GET A X-COORDINATE
Y:    MOV    10(R1),TEMPY          ;GET A Y-COORDINATE
      CMP    TEMPY,LASTY           ;SAME AS LAST Y ?
      BEQ    DUP                   ;YES, THEN SKIP THIS PAIR
      MOV    TEMPX,(R2)+           ;NO, THEN SAVE IN EXD-MEMORY
      MOV    TEMPY,(R2)+
      MOV    TEMPY,LASTY           ;UPDATE LAST Y
DUP:  SOB    R4,X                  ;LOOK AT NEXT X-Y PAIR

      BIS    #140000,@#170010      ;CONTINUOUS ACQUIRE AGAIN
      ADD    #32.,<WDB+8.>             ;UPDATE OFFSET VALUE
      CMP    #3008.,<WDB+8.>           ;ARE WE AT END OF REGION?
      BEQ    DISK
      BR     FRM
```

```
;**********[4] MOVE DATA TO FILE ON HARD DISK**********

DISK:  CLR    R3                    ;R3=BLOCK COUNTER FOR .WRITW
       CLR    <WDB+8.>              ;CLEAR OFFSET VALUE IN WIND. DEF. BLK
       MOV    #94.,R4               ;94 WINDOWS IN A REGION

       .PRINT #FILE                 ;ASK FOR FILE NAME
       .CSIGEN        #ENDCRE,#EXT,#0          ;GET FROM TERMINAL

REMAP:         .MAP   #EMTBK,#WDB              ;REMAP WINDOW FOR WRITW
       .WRITW         #RWBK,#0,WINDW,#1024.,R3 ;WRITE OUT ONE WINDOW WORTH
       ADD    #32., <WDB+8.>                ;UPDATE OFFSET FOR NEXT WINDOW MAP
       ADD    #4,R3                 ;1024 WORDS=4 BLK HAVE BEEN WRITTEN TO DISK
       SOB    R4,REMAP              ;DO 12 WINDOWS = ENTIRE REGION
       .CLOSE #0                    ;CLOSE THE FILE

;*****************************************************************

       .UNMAP         #EMTBK,#WDB
       .ELAW #EMTBK,#WDB
       .ELRG  #EMTBK,#RDB
       .EXIT

RDB:   .RDBBK         RGSIZ
WDB:   .WDBBK         APR,WDSIZ/32.
EMTBK:         .BLKW 2
RWBK: .BLKW 6
EXT:   .WORD 0,0,0,0
LASTY:         .BLKW 1
TEMPX:         .BLKW 1                 ;TEMPORARY STORAGE
TEMPY:         .BLKW 1


FILE:  .ASCII  / NAME OF DATA FILE ?/
ENDCRE         =.
       .END XX1X
```

```
C  PROGRAM XXIDY.FOR
C
C  THIS PROGRAM REARRANGES THE NUMBERS AQUIRED BY THE FEATURE
C  EXTRACT PROGRAM "XXI.MAC" AS FOLLOWS:
C       1.  READS BINARY NUMBERS FROM DATA FILE.  'FEXED.DAT' IS
C           THE PRESUMED FILE NAME.  IF DATA FILE NAME IS DIFF-
C           FERENT, YOU WILL HAVE TO CHANGE LINE #2 OF THIS
C           PROGRAM.  IT IS EASIER TO MAKE YOUR DATA FILE NAME
C           CONFORM TO 'FEXED.DAT'.
C       2.  PUTS THE NUMBERS IN FORTRAN FORMAT IN THE FORM
C
C                       Y-SCAN COORDINATE
C               X-COORDINATE                INTENSITY VALUE
C
C       3.  BREAKS UP THE FRAMES INTO SEPARATE DATA FILES BY THE
C           NAMES 'FEX.NN' WHERE NN IS THE NN-TH FRAME.
C
C  SUBROUTINE CALLED:
C    THIS PROGRAM MAKES USE OF RT11 SYSTEM LIBRARY ROUTINES SUCH AS
C    'READW', 'IGETC', 'IFETCH'.  SO THE OBJECT CODE OF THIS
C    PROGRAM MUST BE LINKED WITH "SYSLIB.OBJ".
C
C  LAST REVISION:  16 FEBRUARY, 1986
C                  HENRY CHOU
C

      INTEGER*2     Y,BUFF(4096),INFILE(4),OUTFILE(12)
      DATA          INFILE/3RDK0,3RFEX,3RED ,3RDAT/
      INCHAN=IGETC()
      M=IFETCH(INFILE)
      L=LOOKUP(INCHAN,INFILE)
      IBLK=0
        DO 900 I=1,23
        N=IREADW(4096,BUFF,IBLK,INCHAN)
        IF(I .GE.10)GOTO 10
        ENCODE(10,5,OUTFILE) I
5       FORMAT('DY:FEX.0',I1)
        GOTO 21
10      ENCODE(10,20,OUTFILE) I
20      FORMAT('DY:FEX.',I2)
21      OPEN(UNIT=1,NAME=OUTFILE,TYPE='NEW')

          J=1
40        IF (BUFF(J) .GE. 1000)GOTO 90
          J1=J+1
          IF((BUFF(J) .EQ. 0) .AND. (BUFF(J1) .EQ. 0))GOTO 100
          WRITE (1,45)BUFF(J),BUFF(J1)
45        FORMAT(I3,1X,I3)
          J=J+2
          IF (J .GT. 4096) GOTO 100
          GOTO 40
```

```
90          Y=BUFF(J)-1000
            WRITE (1,95)Y,JUNK
95          FORMAT(213)
            J=J+1
            IF (J .GT. 4096)GOTO 100
            GOTO 40
100       CLOSE(UNIT=1)
          IBLK  =IBLK+16
900       CONTINUE

          CALL ICLOSE(INCHAN)
          J=ICLOSE()
          CALL IFREEC(INCHAN)
          STOP
          END
```

```
        .TITLE XXMAX
;
; THIS PROGRAM PERFORMS FEATURE EXTRACTION WHEREIN THE COORDINATES OF
; THE MAXIMUM INTENSITY IN EACH SCAN IS SAVE IN EXTENDED MEMORY.  THIS
; PROGRAM RELIES ON 'PREP.FOR' TO SET UP THE VALID FEATURE INTENSITY, WINDOW
; ETC. FOR IT.
;
; THE PROGRAM PERFORMS ITS TASK IN THE FOLLOWING SEQUENCE:
;
;       1. SET UP WINDOW MAPPING TO EXTENDED MEMORY
;       2. START FEATURE EXTRACTION (ON A FROZEN FRAME).
;       3. (A) ACCESS THE EXTRACTED FEATURES.
;          (B) INTERROGATE THE INTENSITIES AT THESE EXTRACTED COORDINATES AND
;              PICK OUT THE MAXIMUM INTENSITY COORDINATES.
;          (C) TRANSFER THE MAXIMUM INTENSITY COORDINATE ON EACH SCAN TO EXTEND
;              MEMORY.
;          (D) CONTINUE UNTIL EXTENDED MEMORY IS FULL (96K WORDS---> 94 FRAMES
;       4. OPEN A FILE ON WINCHESTER DISK TO STORE THE DATA.
;
; SUBROUTINES CALLED:  NONE
;
; LAST REVISION: 14 FEBRUARY, 1987
;                HENRY CHOU
;

XXMAX:      .MCALL .PRINT,.CSIGEN,.WRITW,.CLOSE,.EXIT,.GMCX
        .MCALL .MAP,.UNMAP,.CRAW,.WDBBK,.ELAW,.CRRG,.RDBBK,.ELRG,.WDBDF,.RDBDF

        CLRB   @#170023             ;CONTROL REGISTER OF AP BOARD
        MOV    #170100,R1           ;R1 = HF BOARD BASE ADDRESS
        MOV    #140000,@#170010     ;CONTINUOUS ACQUIRE

;**********[1] SET UP EXTENDED MEMORY MAPPING**********

; DEFINE UNITS:
; 32.-UNIT = GROUP OF 32 WORDS = 2**5   WORDS
; BLOCK    =      256 WORDS = 2**8   WORDS
; PAGE     =     4096 WORDS = 2**12  WORDS = 128  32.-UNITS = 16 BLOCKS
; WINDOW   =          8192 WORDS = 2**13  WORDS = 256  32.-UNITS = 32 BLOCKS
; REGION   =    96256 WORDS =          =3008  32.-UNITS

        APR  =      2               ;PAR/PDR FOR BUFFER (VIRTUAL WINDOW)
        WINDW       =       WDB+W.NBAS;WINDOW BASE ADDRESS
        WDSIZ=      1024.           ;BUFFER SIZE=1024. WORDS
        RGSIZ =     3008.           ;REGION SIZE=3008. 32.-UNITS
        WRNID=      WDB+W.NRID ;REGION ID ADDRESS

        .WDBDF
        .RDBDF

RGMAP:      .CRRG  #EMTBK,#RDB          ;CREATE REGION (96K WORDS)
        MOV    RDB,WRNID            ;MOVE REGION ID TO WINDOW
        .CRAW #EMTBK,#WDB                   ;CREATE A WINDOW (1024 WORDS)
```

```
;**********[2] NOW START THE FEATURE EXTRACT**********

FRM:  BIS    #141000,(R1)          ;SINGLE-FRAME FEX
                                   ;TOGGLE LINE WINDOW MODE
7$:   BIT    #40,2(R1)             ;TEST EVEN/ODD FIELD
      BNE    7$                    ;IF ODD, WAIT
      MOV    #100,2(R1)            ;CLEAR OBRF IN THE EVEN FIELD
8$:   BIT    #20,2(R1)             ;CHECK FOR OBRF=1
      BEQ    8$

      BIC    #140000,@#170010      ;FREEZE FRAME

      MOV    #2,R4                 ;START OF CYCLE, R4 IS A COUNTER
EO:   BIT    #40,2(R1)             ;EVEN/ODD FIELD
      BNE    EO                    ;WAIT IF ODD FIELD
OBR0: MOV    #100,2(R1)            ;CLEAR OBRF IN THE EVEN FIELD
OBR1: BIT    #20,2(R1)             ;CHECK FOR OBRF=1
      BEQ    OBR1
      SOB    R4,OBR0

;**********[3 (A)] GET THE FEATURES FROM FEX BOARD
;**********  (B) INTERROGATE THE INTENSITIES FROM FB BOARD**********
;**********  (C) STORE DATA IN EXTENDED MEMORY
;**********  (D) CONTINUE UNTIL MEMORY IS FULL

      .MAP   #EMTBK,#WDB           ;MAP WINDOW TO REGION
                                   ;WITH OFFSET INTO REGION=OFSET
                                   ;AND LENGTH TO MAP = 1024 WORDS
      .GMCX  #EMTBK,#WDB           ;GET MAPPING STATUS

      MOV    WINDW,R2              ;R2=WINDOW BASE ADDRESS
      MOV    #1041,(R1)            ;OUTPUT BUFFER ACCESS, AUTO INC. 2
      CLR    6(R1)

      MOV    4(R1),R4              ;FEATURE COUNTS
X:    MOV    10(R1),TEMPX          ;GET X-COORDINATE
Y:    MOV    10(R1),TEMPY          ;GET Y-COORDINATE
      CMP    TEMPY,@#170002        ;IS THIS A NEW SCAN?
      BEQ    MAX                   ;IF NO, GO TO PICK OUT MAX I
      MOV    XMAX,(R2)+            ;IF YES, STORE THE (XMAX,YMAX)
      MOV    @#170002,(R2)+        ;FROM THE PREVIOUS SCAN
      MOV    TEMPY,@#170002        ;AND REFRESH THE Y VALUE
      CLR    IMAX                  ;AND SET NEW IMAX=0
MAX:  MOV    TEMPX,@#170000        ;PUT IN NEW X VALUE
      CMP    @#170016,IMAX
      BLE    NEXTX                 ;IF I < IMAX, GO FOR NEXT X-Y PAIR
      MOV    @#170016,IMAX         ;UPDATE IMAX FOR THIS SCAN
      MOV    TEMPX,XMAX            ;AND NOTE THE X COORDINATE
NEXTX:SOB    R4,X
```

```
        BIS     #140000,@#170010    ;BACK TO CONTINUOUS ACQUIRE
        ADD     #32.,<WDB+8.>               ;UPDATE OFFSET VALUE
        CMP     #3008.,<WDB+8.>             ;ARE WE AT END OF REGION?
        BEQ     DISK                ;IF YES, MAKE DATA FILE ON DISK
        BR      FRM1                ;NEXT FRAME

;**********[4] MOVE DATA TO FILE ON HARD DISK**********

DISK:   CLR     R3                  ;R3=BLOCK COUNTER FOR .WRITW
        CLR     <WDB+8.>            ;CLEAR OFFSET VALUE IN WIND. DEF. BLK
        MOV     #94.,R4             ;94 WINDOWS IN A REGION

        .PRINT  #FILE               ;ASK FOR FILE NAME
        .CSIGEN         #ENDCRE,#EXT,#0             ;GET FROM TERMINAL

REMAP:          .MAP    #EMTBK,#WDB                 ;REMAP WINDOW FOR WRITW
        .WRITW          #RWBK,#0,WINDW,#1024.,R3 ;WRITE OUT ONE WINDOW WORTH
        ADD     #32.,<WDB+8.>               ;UPDATE OFFSET FOR NEXT WINDOW MAP
        ADD     #4,R3               ;1024 WORDS=4 BLK HAVE BEEN WRITTEN TO DISK
        SOB     R4,REMAP            ;DO 12 WINDOWS = ENTIRE REGION
        .CLOSE  #0                  ;CLOSE THE FILE

;**********

        .UNMAP          #EMTBK,#WDB
        .ELAW   #EMTBK,#WDB
        .ELRG   #EMTBK,#RDB
        .EXIT

RDB:    .RDBBK          RGSIZ
WDB:    .WDBBK          APR,WDSIZ/32.
EMTBK:          .BLKW 2
RWBK:.BLKW 6
EXT:    .WORD 0,0,0,0

TEMPX:          .BLKW 1
TEMPY:          .BLKW 1
IMAX:   .BLKW 1
XMAX:.BLKW 1

FILE:   .ASCII  / NAME OF DATA FILE ?/
ENDCRE          =.
        .END XXMAX
```

```
;       PROGRAM XXI.MAC
;
;       THIS PROGRAM PERFORMS FEATURE EXTRACTION WHEREIN ALL PIXELS ABOVE A
;       THRESHOLD INTENSITY ARE STORED ALONG WITH THEIR CORRESPONDING INTENSITY
;       VALUES.  THIS PROGRAM MAKES USE OF ONLY ONE REGION (96K WORDS) OF EXTEND
;       MEMORY.
;
;       THE PROGRAM PERFORMS ITS TASK IN THE FOLLOWING SEQUENCE:
;
;               1. SET UP THE MAPPING FOR ONE REGION IN EXTENDED MEMORY.
;               2. START THE FEATURE EXTRACTION ON A FROZON FRAME.
;               3. GET THE EXTRACTED COORDINATES FROM MEMORY BUFFER, AND THEIR
;                  COORESPONDING INTENSITY VALUES.
;                  GO BACK FOR THE NEXT FRAME.
;                  CONTINUE UNTIL THE REGION IN EXTENDED MEMORY IS FILLED.
;               4. PUT THE STORED NUMBERS IN A FILE ON WINCHESTER DISK.
;
;       SUBROUTINES CALLED:     NONE
;
;       NOTES ON USAGE:             1.  THIS PROGRAM RELIES ON 'PREP.FOR' TO SET UP
;                                   WINDOW, THRESHOLD, ETC. FOR IT.
;                                2. AFTER A (BINARY) DATA FILE HAS BEEN MADE ON
;                                   THE WINCHESTER, THE FORTRAN PROGRAM
;                                   'XXIDY.FOR' IS USED TO CONVERT THE NUMBERS
;                                   TO FORTRAN FORMAT AND TO SAVE THEM ON
;                                   FLOPPY DISK.
;
;       LAST REVISION:      25 MAY, 1987
;                       HENRY CHOU
;
            .TITLE XXI

XXI:    .MCALL .PRINT,.CSIGEN,WRITW,.CLOSE,.EXIT,.GMCX
        .MCALL .MAP,.UNMAP,.CRAW,.WDBBK,.ELAW,.CRRG,.RDBBK,.ELRG,.WDBDF,.RDBDF

        CLRB    @#170023            ;CONTROL REGISTER OF AP BOARD
        MOV     #170100,R1          ;R1 = HF BOARD BASE ADDRESS
        MOV     #140000,@#170010    ;CONTINUOUS ACQUIRE

;*********[1] SET UP EXTENDED MEMORY MAPPING**********

; DEFINE UNITS:
; 32.-UNIT = GROUP OF 32 WORDS = 2**5  WORDS
; BLOCK    =      256 WORDS = 2**8  WORDS
; PAGE     =     4096 WORDS = 2**12 WORDS = 128 32.-UNITS = 16 BLOCKS
; WINDOW   =     8192 WORDS = 2**13 WORDS = 256 32.-UNITS = 32 BLOCKS
; REGION   =    96256 WORDS =          =3008 32.-UNITS
```

```
        APR   =     2               ;PAR/PDR FOR BUFFER (VIRTUAL WINDOW)
        WINDW =     WDB+W.NBAS;WINDOW BASE ADDRESS
        WDSIZ =     4096.           ;BUFFER SIZE=4096. WORDS= 1 PAGE
        RGSIZ =     3008.           ;REGION SIZE=3008. 32.-UNITS
        WRNID=      WDB+W.NRID ;REGION ID ADDRESS

        .WDBDF
        .RDBDF

RGMAP:      .CRRG  #EMTBK,#RDB          ;CREATE REGION (96K WORDS)
        MOV   RDB,WRNID             ;MOVE REGION ID TO WINDOW
        .CRAW #EMTBK,#WDB                   ;CREATE A WINDOW (1024 WORDS)

;**********[2] NOW START THE FEATURE EXTRACT**********

FRM:    BIS     #141000,(R1)        ;SINGLE-FRAME FEX
                                    ;TOGGLE LINE WINDOW MODE
7$:     BIT     #40,2(R1)           ;TEST EVEN/ODD FIELD
        BNE     7$                  ;IF ODD, WAIT
        MOV     #100,2(R1)          ;CLEAR OBRF IN THE EVEN FIELD
8$:     BIT     #20,2(R1)           ;CHECK FOR OBRF=1
        BEQ     8$

        BIC     #140000,@#170010    ;FREEZE FRAME

        MOV     #2,R4               ;START OF CYCLE, R4 IS A COUNTER
EO:     BIT     #40,2(R1)           ;EVEN/ODD FIELD
        BNE     EO                  ;WAIT IF ODD FIELD
OBR0:   MOV     #100,2(R1)          ;CLEAR OBRF IN THE EVEN FIELD
OBR1:   BIT     #20,2(R1)           ;CHECK FOR OBRF=1
        BEQ     OBR1
        SOB     R4,OBR0

;**********[3] GET THE NUMBERS FROM MEMORY BUFFER**********

        .MAP    #EMTBK,#WDB             ;MAP WINDOW TO REGION
                                    ;WITH OFFSET INTO REGION=OFSET
                                    ;AND LENGTH TO MAP = 1024 WORDS
        .GMCX #EMTBK,#WDB               ;GET MAPPING STATUS

        MOV     WINDW,R2            ;R2=WINDOW BASE ADDRESS
        MOV     #1041,(R1)          ;OUTPUT BUFFER ACCESS, AUTO INC. 2
        CLR     6(R1)

        MOV     4(R1),R4            ;FEATURE COUNT
        MOV     #600.,@#170002          ;INITIALIZE Y VALUE IN FB BOARD
ACC:    MOV     10(R1),TEMPX        ;GET X COORDINATE
        MOV     10(R1),TEMPY        ;GET Y COORDINATE
```

```
        CMP     TEMPY,@#170002          ;IS THIS THE SAVE Y SCAN AS LAST ?
        BEQ     X                       ;IF YES, DO NOT SAVE IT
        MOV     TEMPY,(R2)              ;IF NO, STORE THE NEW Y VALUE
        ADD     #1000,(R2)+             ;TICK-MARK FOR Y VALUE
        MOV     TEMPY,@#170002          ;AND UPDATE THE Y COORD. IN FB BOARD
X:      MOV     TEMPX,@#170000          ;X COORD. IN FB BOARD
        MOV     TEMPX,(R2)+             ;STORE X COORDINATE IN EXTENDED MEMORY
        MOV     @#170016,(R2)+          ;STORE I(X,Y) IN EXTENDED MEMORY
        SOB     R4,ACC                  ;ACCESS NEXT PAIR OF EXTRACTED FEATURE

        BIS     #140000,@#170010        ;BACK TO CONTINUOUS ACQUIRE
        ADD     #128.,<WDB+8.>          ;UPDATE OFFSET VALUE
        CMP     #2944.,<WDB+8.>         ;ARE WE AT THE END OF REGION ?
        BEQ     DISK                    ;IF YES, MAKE DATA FILE ON WINCHESTER
        BR      FRM                     ;NEXT FRAME

;**********[4] MOVE DATA TO FILE ON HARD DISK**********

DISK:   CLR     R3                      ;R3=BLOCK COUNTER FOR .WRITW
        CLR     <WDB+8.>                ;CLEAR OFFSET VALUE IN WIND. DEF. BLK
        MOV     #23.,R4                 ;23 WINDOWS IN A REGION

        .PRINT  #FILE                   ;ASK FOR FILE NAME
        .CSIGEN         #ENDCRE,#EXT,#0         ;GET FROM TERMINAL

REMAP:          .MAP    #EMTBK,#WDB             ;REMAP WINDOW FOR WRITW
        .WRITW          #RWBK,#0,WINDW,#4096.,R3 ;WRITE OUT ONE WINDOW WORTH
        ADD     #128.,<WDB+8.>          ;UPDATE OFFSET FOR NEXT WINDOW MAP
        ADD     #16.,R3                 ;4096 WORDS=16 BLK HAVE BEEN WRITTEN TO DISK
        SOB     R4,REMAP                ;DO 12 WINDOWS = ENTIRE REGION
        .CLOSE  #0                      ;CLOSE THE FILE

;**********************************************************

        .UNMAP          #EMTBK,#WDB
        .ELAW   #EMTBK,#WDB
        .ELRG   #EMTBK,#RDB
        .EXIT

RDB:    .RDBBK          RGSIZ
WDB:    .WDBBK          APR,WDSIZ/32.
EMTBK:          .BLKW 2
RWBK:   .BLKW 6
EXT:    .WORD 0,0,0,0
TEMPX:          .BLKW 1
TEMPY:          .BLKW 1

FILE:   .ASCII  / NAME OF DATA FILE ?/
ENDCRE          =.
        .END XXI
```

```
        CMP     TEMPY,@#170002          ;IS THIS THE SAVE Y SCAN AS LAST ?
        BEQ     X                       ;IF YES, DO NOT SAVE IT
        MOV     TEMPY,(R2)              ;IF NO, STORE THE NEW Y VALUE
        ADD     #1000,(R2)+             ;TICK-MARK FOR Y VALUE
        MOV     TEMPY,@#170002          ;AND UPDATE THE Y COORD. IN FB BOARD
X:      MOV     TEMPX,@#170000          ;X COORD. IN FB BOARD
        MOV     TEMPX,(R2)+             ;STORE X COORDINATE IN EXTENDED MEMORY
        MOV     @#170016,(R2)+          ;STORE I(X,Y) IN EXTENDED MEMORY
        SOB     R4,ACC                  ;ACCESS NEXT PAIR OF EXTRACTED FEATURE

        BIS     #140000,@#170010        ;BACK TO CONTINUOUS ACQUIRE
        ADD     #128.,<WDB+8.>          ;UPDATE OFFSET VALUE
        CMP     #2944.,<WDB+8.>         ;ARE WE AT THE END OF REGION?
        BEQ     DISK                    ;IF YES, MAKE DATA FILE ON WINCHESTER
        BR      FRM                     ;NEXT FRAME

;**********[4] MOVE DATA TO FILE ON HARD DISK**********

DISK:   CLR     R3                      ;R3=BLOCK COUNTER FOR .WRITW
        CLR     <WDB+8.>                ;CLEAR OFFSET VALUE IN WIND. DEF. BLK
        MOV     #23.,R4                 ;23 WINDOWS IN A REGION

        .PRINT  #FILE                   ;ASK FOR FILE NAME
        .CSIGEN         #ENDCRE,#EXT,#0         ;GET FROM TERMINAL

REMAP:          .MAP    #EMTBK,#WDB             ;REMAP WINDOW FOR WRITW
        .WRITW          #RWBK,#0,WINDW,#4096.,R3 ;WRITE OUT ONE WINDOW WORTH
        ADD     #128.,<WDB+8.>          ;UPDATE OFFSET FOR NEXT WINDOW MAP
        ADD     #16.,R3                 ;4096 WORDS=16 BLK HAVE BEEN WRITTEN TO DISK
        SOB     R4,REMAP                ;DO 12 WINDOWS = ENTIRE REGION
        .CLOSE  #0                      ;CLOSE THE FILE

;*****************************************************

        .UNMAP          #EMTBK,#WDB
        .ELAW   #EMTBK,#WDB
        .ELRG   #EMTBK,#RDB
        .EXIT

RDB:    .RDBBK          RGSIZ
WDB:    .WDBBK          APR,WDSIZ/32.
EMTBK:          .BLKW 2
RWBK:   .BLKW 6
EXT:    .WORD 0,0,0,0
TEMPX:          .BLKW 1
TEMPY:          .BLKW 1

FILE:   .ASCII  / NAME OF DATA FILE ?/
ENDCRE          =.
        .END XXI
```

```
subroutine cft(xl,xu,x,yreal,yimag,nxy,u,nout,mmesh,k,t,ar,ai)
dimension x(1),yreal(1),yimag(1),u(1)
dimension t(mmesh),ar(mmesh),ai(mmesh)
complex phi(10)
ncof = mmesh-k
if (k >4)
        write(5,10)
else if (ncof >nxy)
        write(5,20)
else {
        nab = mmesh-2*(k-1)
        hmesh = (xu-xl)/float(nab-1)
        call umb(xl,xu,nab,k,t,mmesh)
        call dl2sf(x,yreal,nxy,k,t,mmesh,ar)
        call dl2sf(x,yimag,nxy,k,t,mmesh,ai)
        do i = 1,nout {
                tmp = u(i)*hmesh
                call endcor(tmp,phi,k)
                sum = 0.
                sum2 = 0.
                do j = 1,ncof {
                        if (j<k) {
                                re = real(phi(j))
                                aim = aimag(phi(j))
                                }
                        else if (j<nab) {
                                itmp = 2*k-1
                                re = real(phi(itmp))
                                aim = aimag(phi(itmp))
                                }
                        else {
                                itmp = j-nab+k
                                re = real(phi(itmp))
                                aim = aimag(phi(itmp))
                                }
                        tmp = u(i)*t(j)
                        tmq = (re*cos(tmp)-aim*sin(tmp))
                        tmp = (re*sin(tmp)+aim*cos(tmp))
                        sum = sum+(ar(j)*tmq-ai(j)*tmp)
                        sum2 = sum2+(ar(j)*tmp+ai(j)*tmq)
                        }
                yreal(i) = hmesh*sum
                yimag(i) = hmesh*sum2
                }
        }
return
10  format(" spline order should be less than five")
20  format("no of spline coeffs should be less than no of sample pts")
end
```

```
c       program enavsp.f
c
c  This program computes the ensemble-averaged spectrum (FFT or CFT)
c  of an interface.
c  It breaks the interface into several 'windows' (each 'window' of
c  the interface is considered a member of an ensemble), and then
c  computes the Fourier spectrum of each window, and finally the
c  spectra of the windows are added together and averaged.
c
c  INPUT:      File containing raw data of the extracted interface
c              [via XXI.MAC or SYNXYI.MAC]
c
c  OUTPUT:     File (name specified by user) containing the Fourier
c              coefficients in the form
c
c                      k(j)     /a(k)/
c
c  SUBROUTINES CALLED: subinterp.f
c                      subrot.f
c                      subsplyn.f -> PORT library
c                      subfft.f   -> PORT library
c                      subcft.f   -> PORT library, cft_r, endcor_r
c
c  LAST REVISION:  11 June, 1987
c                  Henry Chou

      real x(500),y(500),windx(500),windy(500),k(250),a(250)
      real avspec(250)
      character*20 infile, outfile
      character*1 fc

      print *,' name of raw data file ?'
      read(5,10)infile
10    format(a20)
c-------------------------------------------------------------
c  interpolate interface and rotate to get rid of ramp
c-------------------------------------------------------------
      open(1,file=infile,status='old')
      do 15 j=1,500
        read(1,*,end=16)x(j),y(j)
15    continue
16    nxy=j-1
      close(1)

      call    subrot(x,y,nxy)

      write(6,20)nxy
20    format(' interface consists of ',i5,' points')
      print *,' how many points to smooth over (least square B-spline) ?'
      read(5,*)nsp
```

```
      print *, ' how many regular sample points in x (500 max) ?'
      read(5,*)nn

      call    subsplyn(nxy,nsp,nn,x,y)

      print *, ' how many points in each member of the ensemble ?'
      read(5,*)nwind
      print *, ' how many points to shift at a time ?'
      read(5,*)nshift
      print *, ' use FFT or CFT ( f/c ) ?'
      read(5,30)fc
30    format(a1)
      if (fc .ne. 'c')goto 40
      print *, ' how many points in k space to evaluate Fourier coefficient?'
      read(5,*)nk

c-------------------------------------------------------
c               generate ensemble
c-------------------------------------------------------

40    ncuts=(nn-nwind)/nshift
      do 190 l=1,ncuts
        noffset=(l-1)*nshift
        do 50 m=1,nwind
              windx(m)=x(noffset+m)
              windy(m)=y(noffset+m)
50      continue

        if(fc .eq. 'c')goto 100
c-------------------------------------------------------
c               FFT
c-------------------------------------------------------
        xmin=windx(1)
        xmax=windx(nwind)
        call   subfft(xmin,xmax,nwind,windy,k,a)
        do 80 m=1,nwind/2
              avspec(m)=avspec(m)+a(m)
80      continue
        goto 190
c-------------------------------------------------------
c               CFT
c-------------------------------------------------------
100     call   subcft(windx,windy,nwind,nsp,nwind,nk,k,a)
        do 110 m=1,nk
              avspec(m)=avspec(m)+a(m)
110     continue

190   continue
```

```
c-------------------------------------------------------------
c  make output file
c-------------------------------------------------------------

      print *,' name of output file ?'
      read(5,10)outfile

      open(2,file=outfile)
      jend=nwind/2
      if (fc .eq. 'c')jend=nk
      do 160 j=1,jend
        write(2,*)k(j),avspec(j)/ncuts
160   continue
      close(2)
      stop
      end
```

```
c       program enavspwk.f
c
c  This is a consolidate program for computing the ensemble-
c  averaged spectra and growth rates w(k).  The input files
c  are the B-spline fitted set of interfaces in a time sequence.
c
c  Each interface is broken into several 'windows' (each 'window' of the
c  interface is considered a member of an ensemble), the Fourier
c  spectrum of each window is computed by either FFT or CFT.
c  The ensemble-averaged Fourier spectrum of the interface
c  is computed by averaging together the Fourier spectra of the
c  individual windows.
c
c  We thus get a time sequence of the ENSEMBLE-AVERAGED SPECTRA.  The
c  growth rates are then obtained by fitting the time development of
c  each k component to an exponential in time.
c
c  For an alternative way of ensemble averaging and obtaining the w(k),
c  see the program ENAVWK.F.
c
c  SUBROUTINES CALLED:
c                       subfft.f   -> PORT library
c                       subcft.f   -> PORT library, cft.f, endcor.f
c                       subexp.f
c
c  LAST REVISION:   14 September, 1987
c                       Henry Chou

      real x(500),y(500),windx(500),windy(500),k(250),a(250)
      real avspec(250,20),wk(250)
      integer fm1,fmn,wlen,windows,offset
      character*20 infile, outfile,fname
      character*1 fc

      print *,' name of interface file ? (must be 4 letters long)'
      read(5,10)infile
10    format(a20)
      print *,' which frames to analyze ?'
      read(5,*)fm1,fmn
      print *,' how many points was in each mesh interval of B-spline?'
      read(5,*)nsp
      print *,' how many points in each member of the ensemble ?'
      read(5,*)wlen
      print *,' how many points to shift at a time ?'
      read(5,*)nshift
      print *,' use FFT or CFT ( f/c ) ?'
      read(5,20)fc
20    format(a1)
      if (fc .ne. 'c')goto 25
      print *,' how many points in k space to evaluate Fourier coefficient?'
      read(5,*)nk
```

```
c*************************************************************************
25      do 300 index=fm1,fmn

        write(fname,30)infile,index
30      format(a4,'.',i2.2)
        open(1,file=fname,status='old')
         do 40 j=1,500
         read(1,*,end=45)x(j),y(j)
40       continue
45      nn=j-1
        close(1)

c-------------------------------------------------------
c               generate ensemble
c-------------------------------------------------------

        windows=(nn-wlen)/nshift
        do 200 iwind=1,windows
         offset=(iwind-1)*nshift
         do 50 m=1,wlen
               windx(m)=x(offset+m)
               windy(m)=y(offset+m)
50       continue

         if(fc .eq. 'c')goto 100
c-------------------------------------------------------
c               FFT
c-------------------------------------------------------
         xmin=windx(1)
         xmax=windx(wlen)

         call subfft(xmin,xmax,wlen,windy,k,a)

         nk=wlen/2
         do 80 m=1,nk
               avspec(m,index)=avspec(m,index)+a(m)/windows
80       continue
         goto 200
c-------------------------------------------------------
c               CFT
c-------------------------------------------------------

100      call   subcft(windx,windy,wlen,nsp,wlen,nk,k,a)

         do 110 m=1,nk
               avspec(m,index)=avspec(m,index)+a(m)/windows
110      continue
```

```
c-------------------------------------------------------------------
200     continue

300     continue

        call subexp(avspec,fm1,fmn,nk,wk)

c-------------------------------------------------------------------
c  make output file
c-------------------------------------------------------------------

        print *,' name of output file ?'
        read(5,10)outfile

        open(2,file=outfile)
        do 400 j=1,nk
          write(2,*)k(j),wk(j)
400     continue
        close(2)
        stop
        end
```

```
subroutine endcor(x,y,k)
complex y(15),z,z2,z4,tm
x0 = .2
z = (0.,1.)*x
z2 = z*z
z4 = z2*z2
kk = 2*k-1
if (abs(x)<=x0) {
        tm = (1.+.5*z+z2/6.+z2*z/24.+z4/120.+z4*z/720.)**k
        y(kk) = tm
        if (k==2) {
                y(1) = .5+z/6.+z2/24.+z2*z/120.+z4/720.
                y(2) = cexp(z)*(.5-z/6.+z2/24.-z2*z/120.+z4/720.)
                }
        else if (k==3) {
                y(1) = 1./3.+z/12.+z2/60.+z2*z/360.+z4/2520.
                y(2) = 2./3.+.5*z+7.*z2/30.+z2*z/12.+31.*z4/1260.
                y(4) = cexp(z)*(1./3.-z/12.+z2/60.-z2*z/360.+z4/2520.)
                y(3) = cexp(2.*z)*(2./3.-.5*z+7.*z2/30.-z2*z/12.+31.*z4/1260.)
                }
        else if (k==4) {
                tm = cexp(z)
                y(1) = .25+z/20.+z2/120.+z2*z/840.+z4/6720.
                y(2) = .5+.3*z+7.*z2/60.+z2*z/28.+31.*z4/3360.
                y(3) = .75+.9*z+.625*z2+9.*z2*z/28.+301.*z4/2240.
                y(6) = tm*(.25-z/20.+z2/120.-z2*z/840.+z4/6720.)
                y(5) = tm*tm*(.5-.3*z+7.*z2/60.-z2*z/28.+31.*z4/3360.)
                y(4) = tm**3*(.75-.9*z+.625*z2-9.*z2*z/28.+301.*z4/2240.)
                }
        }
else {
        tm = ((cexp(z)-1.)/z)**k
        y(kk) = tm
        if (k==2) {
                y(1) = (cexp(z)-1.-z)/z2
                y(2) = (1.-cexp(z)*(1.-z))/z2
                }
        else if (k==3) {
                y(1) = (2./(z2*z))*(cexp(z)-1.-z-.5*z2)
                y(2) = (1./(z2*z))*(cexp(2.*z)-4.*cexp(z)+3.+2.*z)
                y(4) = (2./(z2*z))*(cexp(z)*(1.-z+z2/2.)-1.)
                y(3) = (1./(z2*z))*(-1.+4.*cexp(z)-cexp(2.*z)*(3.-2.*z))
                }
        else if (k==4) {
                tm = cexp(z)
                y(1) = (6./z4)*(tm-1.-z-.5*z2-z2*z/6.)
                y(2) = (1.5/z4)*(tm*tm-8.*tm+7.+6.*z+2.*z2)
                y(3) = (1./z4)*(tm**3-4.5*tm*tm+9.*tm-5.5-3.*z)
                y(6) = (6./z4)*(1.-tm*(1.-z+.5*z2-z2*z/6.))
                y(5) = (1.5/z4)*(1.-8.*tm+tm*tm*(7.-6.*z+2.*z2))
                y(4) = (1./z4)*(1.-4.5*tm+9.*tm*tm-tm**3*(5.5-3.*z))
                }
        }
return
end
```

```
c       program interp.f
c  This program fits a parabola to intensity profiles of
c  crystal/melt interface extracted by the 'XXI.MAC' or
c  'XXI2.MAC' programs on the PDP-11.  The data is assumed
c  to be in the format:
c
c                                  y       0
c                                  x       I
c                                  x       I
c                                  :       :
c
c  A minimum of 3 x-coordinates is required for the fit.  If
c  fewer are read, then this y-scan is simply skipped.
c
c  After the least square fit, the program picks out the maximum
c  of the parabola and stores the coordinates in a (user-speccified)
c  data file in the format
c                                  y-coordinate    x-coordinate
c
c  Subroutines called:  none
c
c  Last revision:8 March, 1987
c                       Henry Chou
c
      character*10   infile,outfile
      real           x(10),int(10),xx(480)

      print *,' name of input file ? '
      read(5,10)infile
      print *,' name of output file ? '
      read(5,10)outfile
10    format(a10)
      open(1,file=infile,status='old')
      open(2,file=outfile)

      int(1)=0
40    read(1,*,end=99)i1,i2
      if(i2 .eq. 0)goto 50
      nx=nx+1
      x(nx)=i1
      int(nx)=i2
      goto 40

50    x(1)=x(2)-1.
      x(nx+1)=x(nx)+1
      int(nx+1)=0
      nx=nx+1
      if(nx .lt. 5)goto 90
      xmin=x(1)
```

```
        do 55 j=1,nx
        x(j)=x(j)-xmin
55      continue

       sx=0
       sx2=0
       sx3=0
       sx4=0
       sy=0
       sxy=0
       sx2y=0
       do 80 j=1,nx
        sx=sx+x(j)
        sx2=sx2+x(j)**2
        sx3=sx3+x(j)**3
        sx4=sx4+x(j)**4
        sy=sy+int(j)
        sxy=sxy+x(j)*int(j)
        sx2y=sx2y+(x(j)**2)*int(j)
80     continue

       delta=nx*(sx2*sx4-sx3*sx3)-sx*(sx*sx4-sx3*sx2)+sx2*(sx*sx3-sx2*sx2)
       if (delta .eq. 0.)goto 90
c      a=sy*(sx2*sx4-sx3*sx3)-sx*(sxy*sx4-sx3*sx2y)+sx2*(sxy*sx3-sx2*sx2y)
       b=nx*(sxy*sx4-sx3*sx2y)-sy*(sx*sx4-sx3*sx2)+sx2*(sx*sx2y-sxy*sx2)
       c=nx*(sx2*sx2y-sxy*sx3)-sx*(sx*sx2y-sxy*sx2)+sy*(sx*sx3-sx2*sx2)
       if (c .eq. 0.)goto 90
c      a=a/delta
       b=b/delta
       c=c/delta
       xx(iy)=-b/(2*c)+xmin
90     iy=i1
       nx=1
       goto 40

99     close(1)

       dx=0.962

       do 100 j=1,480
        if(xx(j) .ne. 0.)then
        write(2,*)j*dx,xx(j)*dx
        elseif((xx(j) .eq. 0.) .and. (xx(j-1) .ne. 0.) .and. (xx(j+1) .ne. 0)
&       )then
       xx(j)=(xx(j-1)+xx(j+1))/2
        write (2,*)j*dx,xx(j)*dx
        else
       goto 100
        endif
100    continue
       close(2)
       stop
       end
```

```
c  PROGRAM splyn.f
c
c  This program performs a least square cubic B-spline fit
c  to the input data array (x(i),y(i)). The user supplies
c  the input file name and the "number of points in an inter-
c  val over which a B-spline polynomial is to be fitted. The
c  number of points has to be at least 4, usually 10 is good
c  but it depends on the structure of the data.
c
c  SUBROUITNES CALLED:    Port library
c
c  LAST REVISION:    May 12, 1987
c                    Henry Chou
c
c
      dimension x(1000),y(1000),t(1000),a(1000)
      character*20 infile,outfile
c        x(i)=x-coordinate of data point
c        y(i)=y-coordinate of data point
c        t=mesh arrayA
c        a(i)=coefficient of ith B-spline
      write (6,10)
10    format( ' name of input data file (x,y)?')
      read (5,20)infile
20    format (a20)
      print *, ' number of points in each mesh interval ?'
      read (5,*)nsp
      open (1,file=infile,status='old')
      xmin=800.
      xmax=0.
      nxy=0
      do 50 i=1,1000
        read(1,*,end=55)x(i),y(i)
        if (x(i) .lt. xmin) xmin=x(i)
        if (x(i) .gt. xmax) xmax=x(i)
        nxy=nxy+1
50    continue
55    close(1)
c
      call istkin(2000,2)
c
      call mnpb(x,nxy,nsp,4,t,nt)
c
      call dl2sf(x,y,nxy,4,t,nt,a)
c
      write (6,51)nxy
51    format ( ' original number of data points=',i3)
      write (6,52)
52    format ( ' how many regular mesh points?')
      read (5,*)nn
c
```

```
      call umd(xmin,xmax,nn,x)
c
      call splne(4,t,nt,a,x,nn,y)
c
      write (6,60)
60    format ( ' output file name?')
      read (5,70)outfile
70    format (a20)
      open (2,file=outfile)
      do 80 i=1,nn
        write (2,*)x(i),y(i)
80    continue
      close(2)
      stop
      end
```

```
      subroutine subcft(x,y,nxy,nsp,nn,nk,k,a)

c INPUT:      x(j) = real-space x-coordinates
c             y(j) = interface coordinates
c             nxy  = number of points in y(j)
c             nsp  = number of points to fit through a B-spline
c                    (least square fit)
c             nn   = number of regularized points
c             nk   = number of k-values to evaluate Fourier
c                    coefficients
c
c OUTPUT:     k(j) = wave-vectors ( it goes from 0 to 1 1/um )
c             a(j) = modulus of the Fourier coefficients
c
c SUBROUTINES CALLED:   PORT library
c                       cft.f    ]      [Professor Lax]
c                       endcor.f]
c
c LAST REVISION:        12 May, 1988
c                       Henry Chou

      real x(500),y(500),k(250),a(250),kmax
      real t(500),b(500),bimag(500),yimag(500)
      character*1 yn

c        t=mesh array
c        b(i)=coefficient of ith B-spline


      xmin=x(1)
      xmax=x(nxy)
      do 1 j=1,500
       b(j)=0
       bimag(j)=0
       yimag(j)=0
       a(j)=0
1     continue

      call istkin(2000,2)

      call mnpb(x,nxy,nsp,4,t,nt)

      call d12sf(x,y,nxy,4,t,nt,b)

      call umd(xmin,xmax,nn,x)

      call spline(4,t,nt,b,x,nn,y)
```

```
c  NOTE:  The number of points in k space cannot be arbitrarily set
c         because of the UNCERTAINTY RELATION.  In general, the smallest
c         MEANINGFUL separation in k-space is
c
c                 dk > 2*pi/(xmax - xmin) = 2*pi/(dx)*nn
c

        dk=2*3.14/(xmax-xmin)
        nkmax=1/dk
4       if(nk .gt. nkmax)then
        print *, ' You cannot have so many points in k-space because'
        print *, ' of UNCERTAINTY PRINCIPLE !
        print *, ' The maximum you can have is ',nkmax
        print *, ' enter new nk '
        read(5,*)nk
        goto 4
        endif

5       print *, ' max value in k-space = 1.0 1/um '
        print *, ' do you want another value for kmax (y/n) '
        read(5,6)yn
6       format(a1)
        if(yn .eq. 'n')goto 9
          print *, ' what is your value for kmax ? '
          read(5,*)kmax
          unitk=kmax/nk
          if(unitk .lt. dk)unitk=dk
            do 7 j=1,nk
            k(j)=j*unitk
7           continue
          goto 15

9       do 10 j=1,nk
          k(j)=j/float(nk)
10      continue

15      call cft(xmin,xmax,x,y,yimag,nn,k,nk,nt,4,t,b,bimag)

        do 20 i=1,nk
          a(i)=sqrt(y(i)**2+yimag(i)**2)
20      continue

        return
        end
```

```
      subroutine subfft(xmin,xmax,nxy,y,k,a)

c  INPUT:      xmin=minimum value in real space coordinate
c              xmax=maximum value in real space coordinate
c              nxy =number of data points
c                  (must be acceptible for the FFT routine)
c              y(j)=array containing the interface coordinates,
c                  they are presumed to be equally spaced points.
c
c  OUTPUT:     k(j)=array containing the k-values
c              a(j)=array containing the modulus of the Fourier
c                   coefficients
c
c  SUBROUTINE CALLED:     PORT library
c
c  LAST REVISION:    11 June, 1987
c                    Henry Chou

      real y(1000),yi(1000),k(250),a(250)

      nnp2=nxy+2

      call fftr(nnp2,y,yi)

      dk=2*3.14/(xmax-xmin)

      do 10 j=1,nxy/2
        k(j)=(j-1)*dk
        a(j)=sqrt((y(j)**2+yi(j)**2))/nxy
10    continue

      return
      end
```

```
      subroutine subinterp(infile,scale,y,xx,nxy)

c INPUT:      File name containing raw data
c             scale= length scale in (micrometer/pixel)
c
c OUTPUT:     Arrays y(j) = video scan line where
c                           a valid interface coordinate
c                           was extracted.
c                     xx(j) = interpolated coordinate
c                           corresponding to the y(j)th
c                           scan.
c                     nxy   = number of valid x-y pairs
c
c This program fits a parabola to intensity profiles of
c crystal/melt interface extracted by the 'XXI.MAC' or
c 'XXI2.MAC' programs on the PDP-11.  The data is assumed
c to be in the format:
c
c                                 y      0
c                                 x      I
c                                 x      I
c                                 :      :
c
c A minimum of 3 x-coordinates is required for the fit.  If
c fewer are read, then this y-scan is simply skipped.
c
c After the least square fit, the program picks out the maximum
c of the parabola and stores the coordinates in arrays y(j),xx(j)
c as output.
c
c Subroutines called:  none
c
c Last revision:11 June, 1987
c                      Henry Chou
c
      character*20   infile
      real           x(20),int(20),xx(500),y(500)

      open(1,file=infile,status='old')

      int(1)=0
40    read(1,*,end=99)i1,i2
      if(i2 .eq. 0)goto 50
      nx=nx+1
      x(nx)=i1
      int(nx)=i2
      goto 40
```

```
50      x(1)=x(2)-1.
        x(nx+1)=x(nx)+1
        int(nx+1)=0
        nx=nx+1
        if(nx .lt. 5)goto 90
        xmin=x(1)
          do 55 j=1,nx
          x(j)=x(j)-xmin
55        continue

        sx=0
        sx2=0
        sx3=0
        sx4=0
        sy=0
        sxy=0
        sx2y=0
        do 80 j=1,nx
          sx=sx+x(j)
          sx2=sx2+x(j)**2
          sx3=sx3+x(j)**3
          sx4=sx4+x(j)**4
          sy=sy+int(j)
          sxy=sxy+x(j)*int(j)
          sx2y=sx2y+(x(j)**2)*int(j)
80      continue

        delta=nx*(sx2*sx4-sx3*sx3)-sx*(sx*sx4-sx3*sx2)+sx2*(sx*sx3-sx2*sx2)
        if (delta .eq. 0.)goto 90
        b=nx*(sxy*sx4-sx3*sx2y)-sy*(sx*sx4-sx3*sx2)+sx2*(sx*sx2y-sxy*sx2)
        c=nx*(sx2*sx2y-sxy*sx3)-sx*(sx*sx2y-sxy*sx2)+sy*(sx*sx3-sx2*sx2)
        if (c .eq. 0.)goto 90
        b=b/delta
        c=c/delta
        xx(iy)=-b/(2*c)+xmin
90      iy=i1
        nx=1
        goto 40

99      close(1)

        dx=25.4/28

        nxy=1
        do 100 j=1,480
          if(xx(j) .ne. 0.)then
            y(nxy)=j*scale
            xx(nxy)=xx(j)*scale
            nxy=nxy+1
          elseif((xx(j) .eq. 0.) .and. (xx(j-1) .ne. 0.) .and. (xx(j+1) .ne. 0)
&         )then
        xx(j)=(xx(j-1)+xx(j+1))/2
            y(nxy)=j*scale
            xx(nxy)=xx(j)*scale
            nxy=nxy+1
          else
        goto 100
          endif
100     continue
        nxy=nxy-1
```

```
      subroutine subsplyn(nxy,nsp,nn,x,y)
      dimension x(500),y(500),t(500),a(500)

c INPUT:   nxy=number of valid xy-pairs
c          nsp=number of points to fit each B-spline
c               over in an interval (least-square fit)
c          nn =number of regularized points
c          x(i)=x-coordinate of data point
c          y(i)=y-coordinate of data point
c
c OUTPUT:  x(i)=regularized x-position (scan line)
c          y(i)=interface position, evaluated from the
c               fitted B-splines
c
c SUBROUTINES CALLED: PORT library
c
c LAST REVISION:    11 June, 1987
c                   Henry Chou


      xmin=x(1)
      xmax=x(nxy)

      call istkin(2000,2)

      call mnpb(x,nxy,nsp,4,t,nt)

      call dl2sf(x,y,nxy,4,t,nt,a)

      call umd(xmin,xmax,nn,x)

      call spline(4,t,nt,a,x,nn,y)

      return
      end
```

```
      subroutine subexp(specs,fm1,fmn,nk,wk)

c  subroutine to extract the Mullins-Sekerka "dispersion" curve from
c  a time sequence of Fourier spectra of the developing interface.
c  For each k-vector, the Fourier coefficients /a(k,t)/ ~ exp[w(k)*t].
c
c  Last revision:  9 Sept. 1987
c                  Henry Chou
c
      real specs(250,20),wk(250),time(20),y(20)
      integer fm1,fmn

      sx=0
      sxx=0
      do 10 j=fm1,fmn
        time(j)=0.2*(j-1)
        sx=sx+time(j)
        sxx=sxx+time(j)**2
10    continue

c     open(9,file='linfit.wk')
      do 200 mk=1,nk
      sy=0
      syy=0
      sxy=0
        do 100 m=fm1,fmn
              y(m)=alog(specs(mk,m))
              sy=sy+y(m)
              syy=syy+y(m)**2
              sxy=sxy+time(m)*y(m)
100     continue

        nf=fmn-fm1+1
      delta=nf*sxx-sx**2
      wk(mk)=(nf*sxy-sx*sy)/delta
c       cor=(nf*sxy-sx*sy)/sqrt((nf*sxx-sx**2)*(nf*syy-sy**2))

c     write(9,110)mk,(y(n),n=fm1,fmn),cor
c110  format(i4,10f6.2,f6.2)

200   continue
c     close(9)

      return
      end
```

```
c  PROGRAM laxft.f
c
c  This program performs the "continuous Fourier Transform"
c  a la Lax and Agrawal. The input set of data (x(i), y(i))
c  do not have to be on a regular intervals of x(i). A cubic
c  B-spline fit is first performed on the input data, then
c  the B-spline coefficients are used in the computation of
c  Fourier coefficients in the subroutines "cft.r" and "endcor.r".
c
c
c  SUBROUTINES CALLED:    cft.r
c                         endcor.r
c                         Port library
c
c  LAST REVISION:         June 16, 1987
c                         Henry Chou
c
c
      dimension x(1000),y(1000),t(1000),a(1000)
      dimension yimag(1000),aimag(1000),w(500)
      character*20 infile,outfile
c         x(i)=x-coordinate of data point
c         y(i)=y-coordinate of data point
c         t=mesh array
c         a(i)=coefficient of ith B-spline
      write (6,10)
10    format( ' name of input data file (x,y)?')
      read (5,20)infile
20    format (a20)
      print *, ' number of points in each mesh interval ?'
      read (5,*)nsp
      open (1,file=infile,status='old')
      xmin=800.
      xmax=0.
      nxy=0
      do 50 i=1,1000
        read(1,*,end=55)x(i),y(i)
        if (x(i) .lt. xmin) xmin=x(i)
        if (x(i) .gt. xmax) xmax=x(i)
        nxy=nxy+1
50    continue
55    close(1)
      write(6,*)xmin,xmax
c
      call istkin(2000,2)
c
      call mnpb(x,nxy,nsp,4,t,nt)
c
      call d12sf(x,y,nxy,4,t,nt,a)
c
```

```
      write (6,51)nxy
51    format ( ' original number of data points=',i3)
      write (6,52)
52    format ( ' how many regular sample points?')
      read (5,*)nn
c
      call umd(xmin,xmax,nn,x)
c
      call spline(4,t,nt,a,x,nn,y)
c
      print *, 'how many freguency intervals to evaluate FT ?'
      read (5,*)nw
      do 54 j=1,nw
        w(j)=2*3.14*j/(xmax-xmin)*nn/nw
54    continue
c
      call cft(xmin,xmax,x,y,yimag,nn,w,nw,nt,4,t,a,aimag)
c
      write (6,60)
60    format ( ' output file name?')
      read (5,70)outfile
70    format (a20)
      open (2,file=outfile)
      do 80 i=1,nw
        amod=y(i)**2+yimag(i)**2
        write (2,*)w(i),amod
80    continue
      close(2)
      stop
      end
```

```
      subroutine subrot(x,y,nxy)

c     This program takes a string of x-y values from an input data file
c     and makes a linear least-square fit through the points.  It then
c     rotates the original data to get rid of the ramp and
c     stores the new data in an output file.
c
c     subroutines called:      none
c
c     last revision:           11 June, 1987
c                              Henry Chou
c
      real     x(500),y(500)

      sx=0
      sy=0
      sxx=0
      sxy=0
      do 50 j=1,nxy
        sx=sx+x(j)
        sy=sy+y(j)
        sxx=sxx+x(j)**2
        sxy=sxy+x(j)*y(j)
50    continue

      delta=nxy*sxx-sx**2
      a=(nxy*sxy-sx*sy)/delta
      b=(sxx*sy-sx*sxy)/delta
      cos=1/sqrt(1+a**2)
      sin=a/sqrt(1+a**2)

      do 120 j=1,nxy
        x(j)=(y(j)-b)*sin+x(j)*cos
        y(j)=(y(j)-b)*cos-x(j)*sin
120   continue

      return
      end
```

```
c       rotat.f
c       This program takes a string of x-y values from an input data file
c       and makes a linear least-square fit through the points.  It then
c       rotates the original data to get rid of the ramp and
c       stores the new data in an output file.
c
c       subroutines called:     none
c
c       last revision:          March 24, 1986
c                               Henry Chou
c
        character*10    infile,outfile
        real            x(1000),xp(1000),y(1000)
        print *, ' name of input data file ? '
        read(5,10)infile
10      format(a10)
        open(1,file=infile,status='old')
          do 50 j=1,1000
          read(1,*, end=80)x(j),y(j)
          sx=sx+x(j)
          sy=sy+y(j)
          sxx=sxx+x(j)**2
          sxy=sxy+x(j)*y(j)
          n=n+1
50        continue
80      close(1)
        delta=n*sxx-sx**2
        a=(n*sxy-sx*sy)/delta
        b=(sxx*sy-sx*sxy)/delta
        cos=1/sqrt(1+a**2)
        sin=a/sqrt(1+a**2)
        print *,'  n       a       b    '
        print *, n,a,b
        print *, ' name of output file ? '
        read(5,10)outfile
        open(2,file=outfile)
        do 120 j=1,n
          xp(j)=(y(j)-b)*sin+x(j)*cos
          y(j)=(y(j)-b)*cos-x(j)*sin
          write(2,*)xp(j),y(j)
120     continue
        close(2)
        stop
        end
```

```
c       program spect.f
c  This is a consolidated program which takes the raw data of the
c  extracted interface [via XXI.MAC or XXI2.MAC on the PDP-11] and
c  computes the Fourier spectrum of the interface.
c
c  SUBROUTINES CALLED:  subinterp.f
c                       subrot.f
c                       subsplyn.f -> PORT library
c                       subfft.f   -> PORT library
c                       subcft.f   -> PORT library, cft.f, endcor.f
c
c  LAST REVISION:       11 June, 1987
c                       Henry Chou

      real x(500),y(500),k(250),a(250)
      character*20 infile, outfile
      character*1 fc

      print *, ' name of raw data file ?'
      read(5,10)infile
10    format(a20)
      print *, ' scale of the  picture ( um/pixel ) ?'
      read(5,*)scale
c-------------------------------------------------------
c  interpolate interface and rotate to get rid of ramp
c-------------------------------------------------------
      call    subinterp(infile,scale,x,y,nxy)

      call    subrot(x,y,nxy)

      write(6,20)nxy
20    format(' interface consists of ',i5,' points')
      print *, ' how many points to smooth over (least square B-spline) ?'
      read(5,*)nsp
      print *, ' how many regular sample points in x (500 max) ?'
      read(5,*)nn

      print *,' use FFT or CFT (f/c) ?'
      read(5,30)fc
30    format(a1)
      if(fc .eq. 'c')goto 100

c-------------------------------------------------------
c               FFT
c-------------------------------------------------------

      call    subsplyn(nxy,nsp,nn,x,y)

      xmax=x(nxy)
      call    subfft(xmin,xmax,nn,y,k,a)
      goto 150
```

```
c-------------------------------------------------------
c                CFT
c-------------------------------------------------------

100     print *,' how many points to evaluate Fourier Coefficients (250 max) ?'
        read(5,*)nk

        call    subcft(x,y,nxy,nsp,nn,nk,k,a)

c-------------------------------------------------------
c  make output file
c-------------------------------------------------------

150     print *,' name of output file ?'
        read(5,10)outfile

        open(2,file=outfile)
        jend=nn/2
        if(fc .eq. 'c')jend=nk
        do 160 j=1,jend
          write(2,*)k(j),a(j)
160     continue
        close(2)
        stop
        end
```

REFERENCES

1. V. Weisskopf, Chertok Lecture (1986)
2. Lord J. W. S. Rayleigh, Philos. Mag. **32**, 529 (1916)
3. H. Benard, Ann. Chim. Phys. **7**, ser. 23, 62 (1901)
4. S. Chandrasekhar, *Hydrodynamic and Hydromagnetic Instability* (Dover, New York, 1981)
5. P. Le Gal, A. Pocheau and V. Croquette, Phys. Rev. Lett. **54**, 2501 (1985)
6. R. P. Behringer, Rev. Mod. Phys. **57**, 657 (1985)
7. See articles by Whitehead and Berge in *Fluctuations, Instabilities and Phase Transitions* edited by T. Riste (Plenum, New York 1975)
8. I. Prigogine, *From Being to Becoming* (Freeman, San Francisco 1980)
9. J. Sommeria, S. D. Meyers and H. L. Swinney, Nature **331**, 689 (1988)
10. K. J. Maloy, J. Feder and T. Jossang, Phys. Rev. Lett. **55**, 2688 (1985)
11. Y. Sawada, A. Dougherty and J. P. Gollub, Phys. Rev. Lett. **56**, 1260 (1986)
12. H. E. Stanley and N. Ostrowsky (eds.), *On Growth and Form* (Martinus Nijhoff, Dordrecht 1986)
13. H. Weyl, *Symmetry* (Princeton University Press, 1952)
14. G. W. Baxter and C. D. Andereck, Phys. Rev. Lett. **57**, 3046 (1986)
15. J. P. Gollub and H. L. Swinney, Phys. Rev. Lett. **35**, 927 (1975)
16. P. R. Fenstermacher, H. L. Swinney and J. P. Gollub, J. Fluid Mech. **94**, 103 (1979)
17. D. Bensimon, L. Kadanoff, S. Liang, B. I. Shraiman and C. Tang, Rev. Mod. Phys. **58**, 977 (1986)
18. D. A. Kessler and H. Levine, Phys. Rev. **A33**, 2621 (1986)
19. J. S. Langer, Rev. Mod. Phys. **52**, 1 (1980)
20. C. Herring, Phys. Rev. **82**, 87 (1952)

21. C. Herring in *Structure and Properties of Solid Surfaces* edited by R. Gomer and C. S. Smith (University of Chicago Press 1953)

22. C. Rottman and M. Wortis, Phys. Rep. **103**, 59 (1984)

23. W. Kurz and D. J. Fisher, *Fundamentals of Solidification* (Trans Tech, Switzerland 1984)

24. K. A. Jackson and J. D. Hunt, Trans. Metal Soc. AIME **236**, 1122 (1966)

25. P. Haasen, *Physical Metallurgy* (Cambridge University Press 1986)

26. B. Chalmers, *Principles of Solidification* (Wiley, New York 1964)

27. R. J. Hauy, *Essai d'une theorie sur la structure de cristaux*, Paris 1784 and *Traite de cristallographie*, Paris 1801

28. V. Goldschmidt, Ch. Palache and M. Peacock, Neues Jahrb. Mineral. Geol. Palaeont., Ref. , Beil.-Bd. **63**, 1 (1931)

29. D. H. Donnay, Ann. Soc. Geol. Belg. **18**, B222 (1935)

30. D. P. Woodruff, *The Solid-Liquid Interface* (Cambridge University Press 1973)

31. A. A. Chernov, Sov. Phys. Uspekhi **4**, 116 (1961)

32. H. A. Wilson, Philos. Mag. **50**, 238 (1900)

33. J. Frenkel, Physik. Z. Sov. **1**, 498 (1932)

34. M. Volmer, *Kinetik der Phsenbildung* (Steinkopf Verlag, Dresden and Leipzig 1939)

35. R. Becker and W. Doring, Ann. Physik **24**, 719 (1935)

36. F. C. Frank, Disc. Faraday Soc. **5**, 48 (1949)

37. W. K. Burton, N. Cabrera and C. Frank, Phil. Trans. R. Soc. London, ser. **A243**, 299 (1951)

38. W. B. Hillig and D. Turnbull, J. Chem. Phys. **24**, 914 (1956)

39. K. A. Jackson in *Growth and Perfection of Crystals* edited by Doremus, Roberts and Turnbull (Wiley, New York 1958)

40. K. A. Jackson in *Liquid Metals and Solidification* (American Society of Metals 1958)

41. K. A. Jackson, Mat. Sci. and Eng. **65**, 7 (1984)

42. W. W. Mullins and R. F. Sekerka, J. App. Phys. **34**, 323 (1963)

43. W. W. Mullins and R. F. Sekerka, J. App. Phys. **35**, 444 (1964)

44. R. F. Sekerka in *Crystal Growth, an Introduction* edited by P. Hartman (North Holland, New York 1973)

45. J. W. Cahn in *Crystal Growth* edited by H. S. Peiser (Pergamon Press, Oxford 1967)

46. S. R. Coriell and R. L. Parker, J. App. Phys. **37**, 1548 (1966)

47. M.-A. Lemieux, J. Liu and G. Kotliar, Phys. Rev. **A36**, 1849 (1987)

48. D. A. Kessler, J. Koplik and H. Levine, to appear in Advances in Physics

49. J. S. Langer, Physica **140A**, 44 (1986)

50. M. E. Glicksman, R. J. Schaefer and J. D. Ayer, Metall. Trans. **7A**, 1747 (1976)

51. S.-C. Huang and M. E. Glicksman, Acta Metallurgica **29**, 701 (1981)

52. G. P. Ivantsov, Dokl. Akad. Nauk. USSR **58**, 567 (1947)

53. J. S. Langer and H. Muller-Krumbhaar, J. Cryst. Growth **42**, 11 (1977)

54. J. S. Langer and H. Muller-Krumbhaar, Acta Metallurgica **26**, 1681 (1978)

55. E. Ben-Jacob, N. Goldenfeld, J. S. Langer and G. Schon, Phys. Rev. Lett. **51**, 1930 (1983)

56. E. Ben-Jacob, N. Goldenfeld, J. S. Langer and G. Schon, Phys. Rev. **A29**, 330 (1984)

57. E. Ben-Jacob, N. Goldenfeld, G. Kotliar and J. S. Langer, Phys. Rev. Lett. **53**, 2110 (1984)

58. R. C. Brower, D. A. Kessler, J. Koplik and H. Levine, Scripta Metallurgica **18**, 463 (1984)

59. D. A. Kessler, J. Koplik and H. Levine, Phys. Rev. **A30**, 2820 (1984)

60. D. A. Kessler, J. Koplik and H. Levine, Phys. Rev. **A30**, 3161 (1984)

61. D. A. Kessler, J. Koplik and H. Levine, Phys. Rev. **A31**, 1712 (1984)

62. J. M. van den Broeck, Phys. Fluids **26**, 2033 (1983)

63. D. A. Kessler, J. Koplik and H. Levine, Phys. Rev. **A33**, 3352 (1986)

64. D. A. Kessler, J. Koplik and H. Levine, Phys. Rev. **A33**, 7867 (1986)

65. D. A. Kessler, J. Koplik and H. Levine, Phys. Rev. Lett. **57**, 3069 (1986)

66. D. Bensimon, P. Pelce and B. I. Shraiman, J. Physique **48**, 2081 (1987)

67. B. I. Shraiman, Phys. Rev. Lett. **56**, 2028 (1986)

68. A. Barbieri, D. C. Hong and J. S. Langer, Phys. Rev. **A35**, 1802 (1987)

69. R. Trivedi and K. Somboonsuk, Mat. Sci. and Eng. **65**, 65 (1984)

70. K. Somboonsuk, J. T. Mason and R. Trivedi, Metall. Trans. **15A**, 967 (1984)

71. H. Esaka and W. Kurz, J. Cryst. Growth **72**, 578 (1985)

72. H. Honjo, S. Ohta and M. Matsushita, Phys. Rev. **A36**, 4555 (1987)

73. A. Dougherty, P. D. Kaplan and J. P. Gollub, Phys. Rev. Lett. **58**, 1652 (1987)

74. A. Dougherty and J. P. Gollub, preprint 1988

75. H. Honjo and Y. Sawada, J. Cryst. Growth **58**, 297 (1982)

76. H. Honjo, S. Ohta and Y. Sawada, Phys. Rev. Lett. **55**, 841 (1985)

77. T. Okamoto and K. Kishitake, J. Cryst. Growth **29**, 131 (1975)

78. J. H. Bilgram, M Firmann and W. Kanzig, Phys. Rev. **B37**, 685 (1988)

79. Ch. Fattinger, F. Honegger and W. Lukosz, Phys. Rev. Lett. **57**, 2536 (1986)

80. S. C. Hardy and S. R. Coriell, J. App. Phys. **39**, 3505 (1968)
and J. Cryst. Growth **4**, 569 (1968)

81. K. Suzuki A. Hikata and C. Elbaum, Phys. Rev. Lett. **59**, 2686 (1987)

82. S. De Cheveigne, C. Guthmann and M. M. Lebrun, J. Cryst. Growth **73**, 242 (1985)

83. S. De Cheveigne, C. Guthmann and M. M. Lebrun, J. Physique **47**, 2095 (1986)

84. F. Heslot and A. Libchaber, Phys. Scripta **79**, 126 (1985)

85. J. Bechhoefer and A. Libchaber, Phys. Rev. **A35**, 1393 (1987)

86. P. Oswald, J. Bechhoefer and A. Libchaber, Phys. Rev. Lett. **58**, 2318 (1987)

87. J. Bechhoefer, P. Oswald, A. Libchaber and C. Germain, Phys. Rev. **A37**, 1691 (1988)

88. W. Lang, Zeiss Information **70**, 114 (1969) and **71**, 12 (1969)

89. J. D. Hunt, K. A. Jackson and H. Brown, Rev. Sci. Inst. **37**, 805 (1966)

90. K. A. Jackson, private communication (1984)

91. AT&T Bell Laboratories PORT Scientific Subroutine Library

92. C. De Boor, *A Practical Guide to Splines* (Springer-Verlag, New York 1978)

93. E. O. Brigham, *The Fast Fourier Transform* (Prentice Hall 1974)

94. M. Lax and G. P. Agrawal, Math. of Computation **39**, 535 (1982)

95. C. W. Meyer, G. Ahlers and D. S. Cannell, Phys. Rev. Lett. **59**, 1577 (1987)

96. R. K. Pathria, *Statistical Mechanics* (Pergamon, Oxford 1972)

97. D. E. Ovsienko and G. A. Alfintsev in *Crystal Growth, Properties and Applications* edited by H. C. Freyhardt (Springer-Verlag, Berlin 1980)

98. X. W. Qian, private communication (1988)

99. H. Levine, private communication (1988)

100. D. Wolkind and L. Segal, Phil. Trans. R. Soc. London **268**, 351 (1970)

101. J. S. Langer and L. A. Turski, Acta Metall. **25**, 1113 (1977)

102. A. Karma, Phys. Rev. Lett. **57**, 858 (1985)

103. M. Ben Amar and B. Moussallam, Phys. Rev. Lett. **60**, 317 (1988)

104. K. A. Jackson, private communication

105. S. T. Chui and J. D. Weeks, Phys. Rev. **B14**, 4976 (1976) and J. D. Weeks in *Strongly Fluctuating Condensed Matter Systems* edited by T. Riste (Plenum, New York 1980)

106. M. Ben Amar and Y. Pomeau, Europhys. Lett. **2**, 307 (1986)

107. R. Pieters and Langer, Phys. Rev. Lett. **56**, 1948 (1986)

108. R. Pieters, Phys. Rev. **A37**, 3126 (1988)