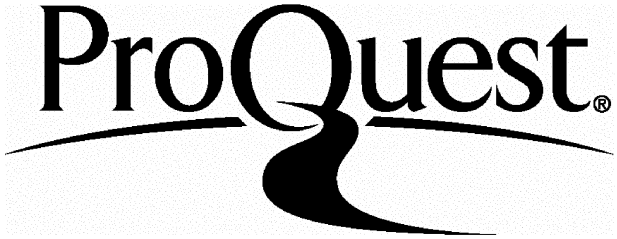
Visual Function in Glaucoma: improving the assessment of computerised visual fields

Ananth Chitur Viswanathan

Submitted in fulfilment of the requirements
for the degree of Doctor of Medicine

Institute of Ophthalmology
University College
London



Abstract

Glaucoma is a leading cause of world blindness. In order to ascertain whether the disease is progressive or stable, glaucoma patients' visual function is monitored at regular intervals by testing the visual field using the technique of automated perimetry. The numerical data obtained, relating to the spatial co-ordinates of the test locations and light sensitivities at those locations, are amenable to sophisticated statistical analysis.

This thesis centres on a method for detecting glaucomatous change in serial visual fields known as pointwise linear regression: univariate linear regression of sensitivity on time is performed for each test location in the visual field. This method has been incorporated into a software package, PROGRESSOR.

Results indicate that PROGRESSOR is superior to previously accepted glaucoma change probability software in the early detection of glaucomatous visual deterioration. A higher level of concordance between expert observers in the assessment of glaucomatous visual progression is found when PROGRESSOR, rather than standard clinical methods, is used.

The optimum frequency of visual field testing is investigated: reducing the frequency of examinations from 3 per year to 1 per year results in failure to detect over half of the deteriorating test locations.

A strong association is found between patient perception of visual disability and objectively measured damage and deterioration in glaucomatous visual fields even in mild to moderate glaucoma.

Image processing techniques have previously been used to increase the reproducibility and predictability of automated visual field tests. These techniques are incorporated into PROGRESSOR. Results presented in this thesis indicate that these benefits are obtained without delayed detection of visual field progression.

The PROGRESSOR software shows great potential in the detection and quantification of glaucomatous visual field deterioration. It promises to be an important part of the management of glaucoma patients in the future.

Table of Contents

List of Tables

List of Figures

Key of Abbreviations

ALTP	Argon laser trabeculoplasty
dB	Decibel
IOP	Intraocular pressure
LF	Long-term fluctuation
MD	Mean deviation
MRI	Magnetic resonance imaging
NTG	Normal tension glaucoma
OHT	Ocular hypertension
POAG	Primary open-angle glaucoma
SD	Standard deviation
SE	Standard error
SF	Short-term fluctuation
SITA	Swedish Interactive Threshold Algorithm

Acknowledgements

I gratefully and wholeheartedly acknowledge the guidance and support of my supervisors Professor Fred Fitzke and Professor Roger Hitchings. I sincerely thank them for actively directing my research and supporting my investigation of areas of interest.

I would like to express my thanks to my colleagues who collaborated in this research:

- Dr. David Crabb for statistical advice and for collaboration in the research described in Chapter 4.2 and Chapter 5.
- Mr. Andrew McNaught for collaboration in the research described in Chapter 4.2 and for collaboration and much of the data collection in the research described in Chapter 5.
- Mr. Mark Westcott, Miss Deborah Kamal and Mr. David Garway-Heath for collaboration in the research described in Chapter 4.2.
- Mr. Darmalingun Poinoosawmy and Dr. Luigi Fontana for collaboration in the research described in Chapter 5.

I am deeply indebted to the International Glaucoma Association, particularly Mr. Ronald Pitts Crick and Mr. David Wright, for the generous funding and travel grant provisions I received as an International Glaucoma Association Fellow while the research described in this thesis was being undertaken.

I thank Mr. Chris Jubb for invaluable technical help during the research described in this thesis.

Finally, my thanks go to Narciss for her love and support.

CHAPTER 1. INTRODUCTION

1.1 Glaucoma

1.1.1 Definition and classification

The term 'glaucoma' encompasses a group of pathological conditions which is distinguished by characteristic patterns of damage to the optic nerve head with concomitant abnormalities of the visual field. The visual field is defined as that portion of space in which objects are simultaneously visible to the steadily fixating eye.[1] Intraocular pressure (IOP) greater than 21mmHg (conventionally taken as two standard deviations above the population mean[2]) is an important characteristic of glaucoma, but raised IOP is neither a necessary nor a sufficient condition for the diagnosis of glaucoma. It is reasonable, however, to suggest that glaucoma is often, but not always, associated with a raised IOP.[3]

Glaucoma may be classified into primary and secondary types. In the majority of cases, glaucoma is unrelated to other ocular or systemic disease and is considered primary, whereas secondary glaucomas are related to conditions such as ocular trauma or inflammation. Primary glaucomas are further classified according to the appearance of the aqueous drainage angle on gonioscopy. If, in a case of glaucoma, the angle is obscured by normal or pathological structures, a diagnosis of closed-angle glaucoma is made: otherwise the diagnosis is open-angle glaucoma. The congenital glaucomas are a comparatively rare sub-group characterised by malformations of the aqueous drainage structures with or without other ocular and systemic congenital anomalies.

The commonest form of glaucoma in the UK is primary open-angle glaucoma (POAG). A substantial minority of POAG patients (around 15% in population based studies) consistently demonstrate intraocular pressures within the normal range.[4] These POAG patients are further classified as having normal tension

glaucoma (NTG). Subjects who have normal visual fields but have risk factors for the development of POAG, such as raised IOP or optic nerve head damage, are typically described as 'POAG suspects' since they are at risk of developing POAG in the future. Another classification term is 'ocular hypertension' (OHT). This refers to individuals with raised IOP in the absence of visual field or optic disc damage.

Untreated POAG leads to progressive irreversible visual loss. Although IOP measurement and evaluation of the optic nerve head are very important in the diagnosis and classification of POAG, it is visual field examination which provides the best continuing measure of a POAG patient's disease status and the best measure of the functional impact of the disease, or of any therapy.

1.1.2 Aetiology

The pathophysiology of POAG is not fully understood. The two main theories to account for the retinal nerve fibre loss seen in glaucoma are the mechanical theory and the vascular theory. The mechanical theory proposes that increased IOP results in direct nerve fibre damage, whilst the vascular theory suggests that an abnormality in blood flow to the optic nerve head is the main cause of this damage.[5]

1.1.2.1 Mechanical theory

The hypothesis that glaucomatous optic neuropathy is produced mechanically by raised IOP was first stated by von Graefe in 1857.[6] Experimentally-induced IOP elevation in primates has been found to produce optic nerve head damage and loss of visual function.[7 8] Quigley has stressed the primary role of raised IOP in the pathogenesis of glaucomatous optic neuropathy, postulating that the characteristic patterns of visual field damage observed in glaucoma result from regional differences in the structure of the scleral lamina cribrosa and hence differing susceptibility of nerve fibres to mechanical damage as they pass through this area.[9]

Both mean and maximum IOP have been reported as closely associated with visual field deterioration[10-17] and optic nerve damage.[18] In a group of NTG subjects worse visual field damage was found in the eye with the higher IOP.[19] However, other studies have failed to demonstrate an unequivocal relationship between IOP and loss of visual function[20-22] and have reported only moderate levels of within-subject correlation between asymmetrical IOP levels and correspondingly asymmetrical visual field damage.[23 24]

1.1.2.2 Vascular theory

In 1858 Jaeger put forward the hypothesis that damage to the optic nerve head in glaucoma was the result of impaired circulation in the short posterior ciliary arteries.[25] The degree of association between systemic arterial blood pressure and glaucoma has recently been investigated.[26 27] Progressive visual field deterioration in the face of IOP levels within the normal range was associated with lower systemic arterial blood pressure than controls, and with relative nocturnal hypotension. It is postulated that vascular risk factors cause hypoperfusion of the optic nerve head leading to glaucomatous damage. Peripheral vasospasm[28] and spontaneous platelet aggregation[29] have also been implicated in the pathogenesis of glaucoma. The results of indirect techniques such as pulsatile ocular blood flow measurement[30] and colour Doppler ultrasonography[31] have been used to infer a disturbance of normal ocular blood flow in NTG. The response of NTG patients to systemically administered calcium channel antagonists[32 33] suggests that vascular mechanisms are important role in the development of NTG.

It is likely that glaucomatous optic neuropathy occurs not as a result of purely mechanical or purely vascular factors but rather from a combination of both. It may be that there are subpopulations within the glaucomatous population in which either vascular or mechanical factors predominate.[34]

1.1.3 Epidemiology

At present glaucoma is the third commonest cause of blindness in the world.[35] It will be the commonest cause of irreversible blindness by the year 2000.[36] It is estimated that approximately 5.2 million people are bilaterally blind from glaucoma: this represents 15% of the total burden of world blindness.[37] There are approximately 250,000 known sufferers in the UK and it is likely that an equal number of cases remain undiagnosed.[38] With the shift towards an older population the medical, social and economic burdens imposed by glaucoma are likely to increase.

1.1.3.1 Prevalence

An early landmark in the epidemiological investigation of POAG was the Ferndale study.[2] This study was notable because a large proportion of a geographically defined population was examined using comprehensive case-finding methods and extensive visual field testing. From the 4231 subjects studied (age range 40-70 years) a prevalence of POAG of 0.5% was obtained. In the Framingham Eye study a sample of 2631 (age range 52-85 years) of the 3977 members of the Framingham (Massachusetts) Heart study population still living in 1973-1975 underwent ophthalmological examinations for cataract, glaucoma, diabetic retinopathy, macular degeneration and visual acuity.[39] The prevalence of POAG was found to be 1.4%. A longitudinal study of 1511 (age range 55-70 years) of the 1963 residents of a Swedish rural and suburban district using visual fields, ophthalmoscopy, slit-lamp and tonometry gave an initial prevalence of POAG of 0.9%.[40]

Three later studies, however, have suggested a higher prevalence of POAG. The Roscommon study, set in a rural community in the West of Ireland, was based on evaluating multiple risk indicators including optic nerve head assessment, visual fields and medical history, in addition to IOP.[38] A total of 2186 people over the age of 50 were examined which represented a 99.5% response rate. The high response rate was achieved by the community basis of the study and vigorous follow up of non-attenders. The prevalence of POAG was estimated at 1.9%. The

prevalence of blindness amongst all POAG patients was 7.3%. The Baltimore Eye Survey [41] in which 2395 black and 2913 white subjects aged 40 years or older from the eastern and south-eastern health districts of Baltimore were studied gave similar estimates of prevalence to the Roscommon study. The Beaver Dam Eye study[42] conducted in Wisconsin on a sample of 4926 subjects (aged 43 years or older) found the overall prevalence of POAG to be 2.1%. Despite differences in examination method and diagnostic criteria for POAG these three recent studies show remarkable concordance in prevalence of POAG against covariates of both age and gender.

A recent meta-analysis of glaucoma prevalence data has stressed the importance of a functional test of the visual field as part of the diagnostic criteria for POAG and excluded studies based on measurement of IOP alone.[43]

1.1.3.2 Incidence

Direct measurement of the incidence of POAG is hampered by uncertainty over the definition of an incident case: in a particular case the characteristic morphological changes in the optic nerve head and retinal nerve fibre layer may, or may not coexist with repeatable visual field loss[44]. The incidence of manifest glaucoma has been estimated for the age band 55-69 years by means of repeated automatic perimetry in a defined general population from a small community in Sweden. It was estimated at 0.24% per year.[45] Further estimates should arise from the Baltimore and Beaver Dam Eye studies as more longitudinal data accumulates.

1.1.3.3 Risk factors

Since the precise causative mechanisms of POAG are unknown there is no clear cut distinction between aetiology and risk factors. Nevertheless, several epidemiological studies have found demographic, ocular and systemic attributes which predispose an individual to POAG.

1.1.3.3 (i) Age

Age appears to be a primary risk factor for POAG. The Beaver Dam Eye study reported a prevalence of POAG of 0.9% in subjects 43 to 54 years of age: this rose to 4.7% in people aged over 75 years.[42] The Roscommon study showed a four and one half times increased risk of POAG in the 70-79 year old age group as compared to the 50-59 age group.[38] The Baltimore Eye Survey found an eightfold increase for the same age groups.[41]

1.1.3.3 (ii) Intraocular pressure

Aside from being one of the putative mechanisms of glaucomatous damage (see section 1.1.2.1) IOP is a major risk factor for POAG. Although only a small number of patients with OHT develop POAG every year, studies on the conversion from OHT to POAG have demonstrated that the overall risk of developing POAG is approximately five times higher in subjects with IOPs greater than 21mmHg than in subjects with lower IOPs[46] and that the higher the IOP at screening, the greater the risk of POAG.[4]

1.1.3.3 (iii) Race

Black Americans are at higher risk of primary open-angle glaucoma than their white neighbours. The Baltimore Eye Survey[41] reported that some age-adjusted prevalence rates for primary open-angle glaucoma were four to five times higher in blacks as compared with whites. Rates among blacks ranged from 1.23% in those aged 40 to 49 years to 11.26% in those 80 years or older, whereas rates for whites ranged from 0.92% to 2.16%, respectively. Similar figures have also been reported in another predominantly black study population in the Barbados Eye study.[47] Black patients with POAG tend to present considerably earlier than their white counterparts, which suggests an earlier onset of disease.[48] This finding, coupled with higher prevalence rates of POAG across all age groups, suggests that the burden of POAG-induced blindness will weigh particularly heavily on the black population.

A population-based, collaborative glaucoma survey was conducted in seven regions throughout Japan, during the years of 1988 and 1989. The total number of subjects

examined was 8,126 out of 16,078 residents aged 40 years or older, representing a participation rate of 50.54%.[49] This research demonstrated a relatively high prevalence of NTG and a low rate of OHT in the Japanese as compared with Caucasians: this might reflect a racial peculiarity in the age-specific trend of the intraocular pressure.

1.1.3.3 (iv) Family history

The Baltimore Eye Survey reported higher age-adjusted associations of primary open angle glaucoma with a family history of glaucoma in siblings (odds ratio = 3.69) than in parents (odds ratio = 2.17) or children (odds ratio = 1.12) of individuals with POAG.[50] The Barbados Eye study identified a family history of open angle glaucoma as a major risk factor for POAG: this association was stronger in men than women.[51] IOP, facility of aqueous outflow and dimensions of the optic nerve head appear to be genetically determined.[52] An epidemiological, clinical and genetic study carried out in the North Western district of Greece (Epirus) suggested that one autosomal dominant gene is the main factor for the heredity of primary open-angle glaucoma in this population.[53] Linkage analysis of 37 members of a family affected with an autosomal dominant form of juvenile open angle glaucoma has mapped the disease-causing gene to chromosome 1q21-q31.[54]

1.1.3.3 (v) Other factors

Although it was previously held that diabetes was associated with POAG, either through common genetic factors or because of diabetic vasculopathy at the optic nerve head,[46 55] the results of large population-based studies are inconsistent on the presence[56] or absence[51 57] of an association between diabetes mellitus and POAG.

It is accepted that local vascular factors may play an important role in the pathogenesis of optic nerve head damage (see section 1.1.2.2). Additionally, studies have demonstrated an association between a decrease in systemic blood pressure, due either to a hypotensive crisis or to anti-hypertensive therapy, and the development of POAG.[46] Although a positive association between migraine and

NTG has been reported[58] results from the Beaver Dam study found no association between POAG and a history of migraine.[59]

There is conflicting evidence on whether myopia is[60 61] or is not[62 63] a risk factor for POAG. Similarly, large population-based studies variously report that gender is[51] or is not[41 42] associated with POAG.

1.1.4 Treatment

The purpose of glaucoma management is to prevent patients' visual function from falling below their visual requirements within their lifetime. Although there is much interest in the areas of neuroprotection and the pharmacological manipulation of optic nerve head blood flow, there is currently no evidence for their *in vivo* efficacy therefore therapeutic options are limited to methods of lowering intraocular pressure. These methods include medical, laser and surgical therapies.

1.1.4.1 Medical therapy

At present, the mainstay of medical treatment in the UK is topical beta-blockers, which reduce the production of aqueous humour. Other topical treatments include various miotics such as pilocarpine, a topical carbonic anhydrase inhibitor (dorzolamide), alpha-agonists (brimonidine and apraclonidine) and a prostaglandin analogue (latanoprost). Each of these groups of drugs lowers intraocular pressure through a different pharmacological mechanism. All the groups of drugs also have different side effects in addition to those caused through allergy to the preservative used in their manufacture.

The majority of patients in the UK receive topical therapy as first line. However, if intraocular pressure is lowered insufficiently to prevent progression of the disease or if side effects are an insuperable problem then consideration must be given to laser or surgical approaches.

1.1.4.2 Laser therapy

Argon laser trabeculoplasty (ALTP) treatment has been shown to reduce intraocular pressure by an average of 30 per cent.[64] It is most effective in eyes with a highly

pigmented trabecular meshwork and in the elderly. However, its efficacy is known to diminish with time and repeated treatments are rarely of benefit. It is also possible that ALTP treatment may prejudice the results of future glaucoma drainage surgery.[65] Thus, ALTP is generally used as a temporising measure in elderly patients in whom a modest lowering of IOP is required.

1.1.4.3 Surgical therapy

The operation of choice for POAG is trabeculectomy.[66] An internal sclerostomy between the anterior chamber and the subconjunctival space is 'guarded' by a partial thickness scleral flap so as to prevent postoperative hypotony. Developments such as argon laser or needle suture lysis and the use of releaseable sutures have allowed the scleral flap to be sutured to its bed relatively tightly. This approach tends to lessen the risk of excess bulk flow of aqueous humour in the early postoperative period with the complications which that process entails. The likelihood of failure due to conjunctival scarring in the later postoperative period may be lessened, in those eyes at particular risk, by the use of adjunctive antimetabolites such 5-fluorouracil and mitomycin C.[67]

Three studies in the UK which have compared medical and surgical treatment as first line therapy have all suggested that primary surgery affords the most effective control of intraocular pressure.[10 12 68] However, these findings may require re-evaluation in the future owing to the recent advent of more powerful topical ocular hypotensive drugs.

Guidelines for the management of POAG have recently been published by the Royal College of Ophthalmologists[69] and Hitchings.[70]

1.2 Visual Field Analysis

1.2.1 Fundamentals of perimetry

The visual field is the region of space from which photons in the visible spectrum can enter the eye or eyes leading to the perception of light. It has long been realised

that human visual function is not uniform across the visual field. As early as 150 B.C. Ptolemy attempted to measure the visual field. Damian, writing in the fifth century, describes 'sharp central vision' as opposed to 'blurred peripheral vision.'[71] From these beginnings techniques for measuring visual fields have greatly increased in complexity: the need for precise, quantifiable, reproducible estimates of the visual field has resulted in a variety of perimeters which are used to examine the 'hill of vision'[72] under standardised conditions.

1.2.2 Development and history

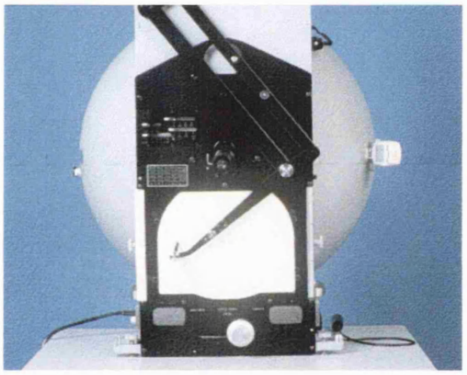
1.2.2.1 Early perimetry

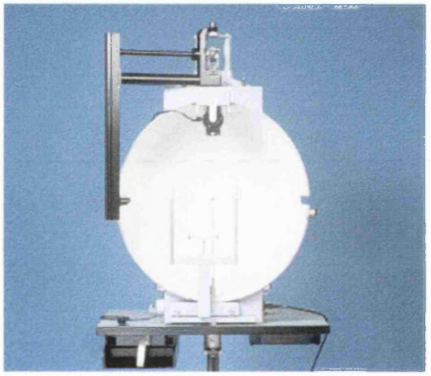
The first description of a quantitative perimetric technique (strictly 'campimetric' rather than perimetric as a flat screen rather than a bowl was used) was by von Graefe in 1856.[73] A large illuminated test object was moved in at various meridians from the periphery of a flat board while the patient fixated on a central target. When the patient reported that the test object was seen a mark was made on the board. In this way the perimeter of the visual field was mapped. The following year Aubert and Foerster[74] described a perimetric technique where a test stimulus was presented on an arc. Several meridians could be tested by rotating the arc. The first perimeter hemispherical bowl was devised in 1872.[75] In 1889, Bjerrum illustrated the characteristic arcuate distribution of glaucomatous visual field loss using a tangent screen examination.[76] Using this type of test Rönne (1909) described the 'nasal step' pattern of visual field loss in glaucoma and related it to the anatomical distribution of the retinal nerve fibre layer.[77] In 1931, Traquair used similar techniques to discriminate between different stages in the progression of glaucomatous visual field loss.[72]

1.2.2.2 Kinetic perimetry

Before the advent of computerised perimeters the most reproducible estimates of the visual field were made with manual hemispherical bowl perimeters such as the

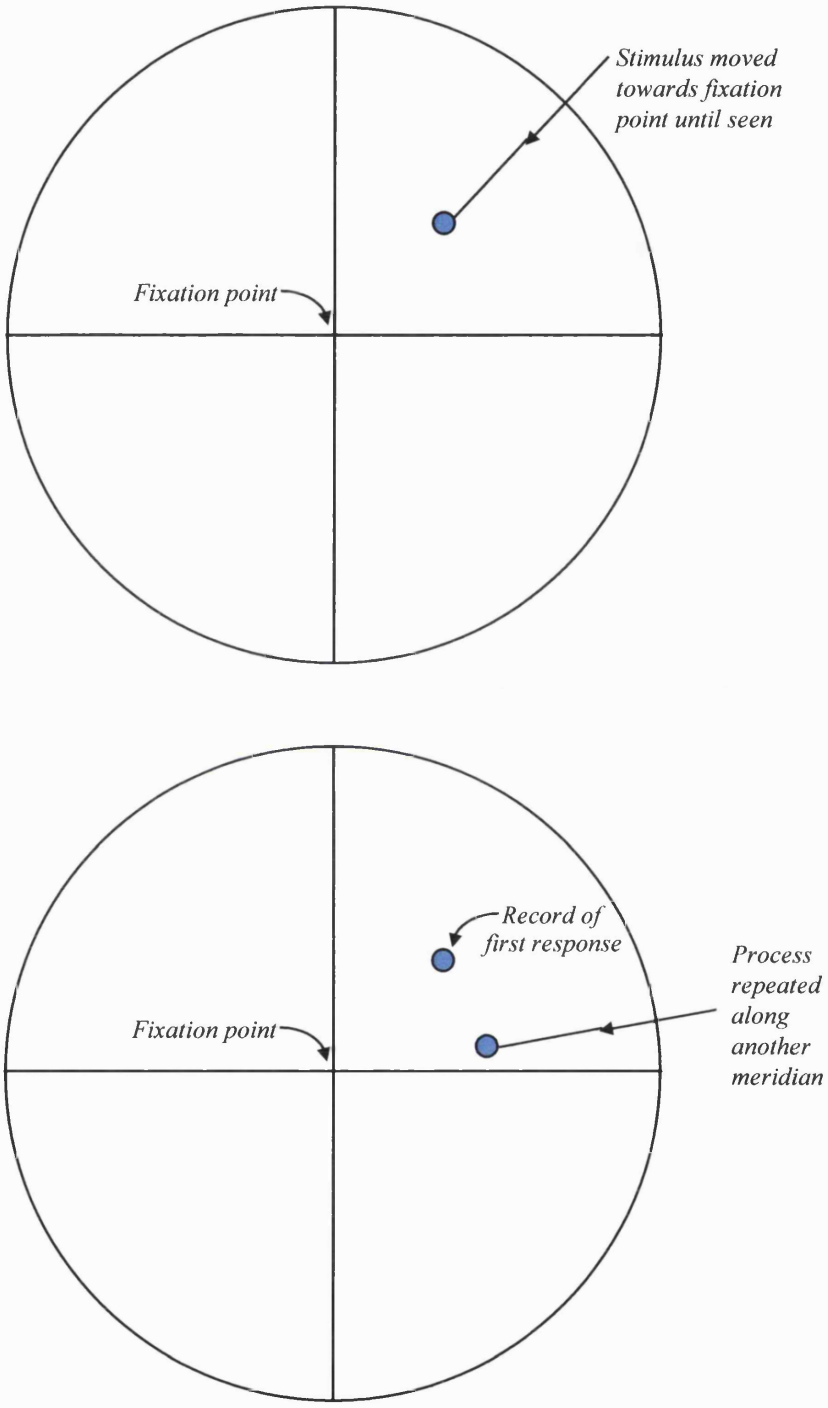
Goldmann perimeter[78] (Figs. 1.1 *a*, *b*). This instrument has the great advantage over the earlier arc perimeters and tangent screens that the visual field may be examined in conditions of constant background illumination, and thus fixed light adaptation state of the eye. This has a critical bearing on the properties of the visual field: for example, in photopic conditions the fovea has a higher light sensitivity than the parafoveal area whereas in scotopic conditions it is relatively depressed. Furthermore, the direct mechanical link between the stimulus control and a plotting pen provided the first visual field measurement with a degree of reproducibility and accuracy.

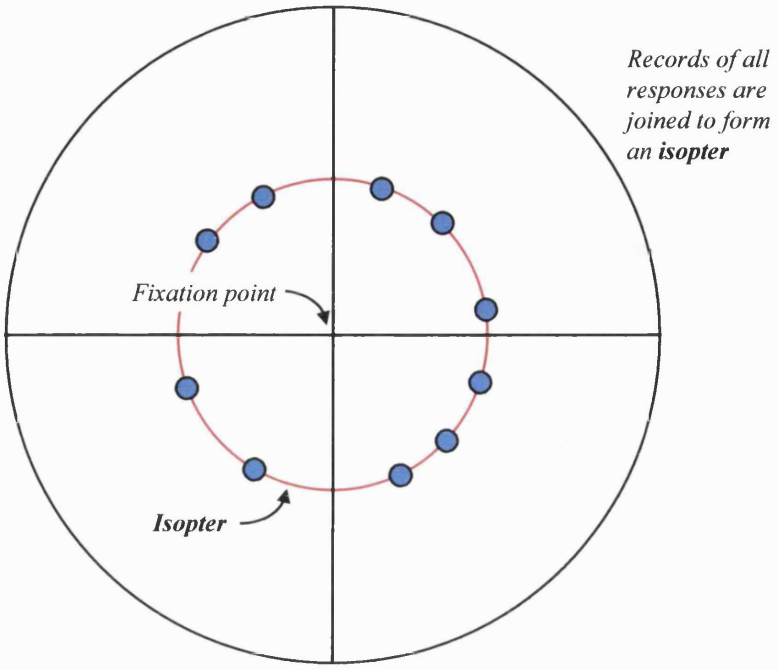




Figures 1.1a and b. *The Goldmann perimeter (Interzeag AG, Schlieren, Switzerland).*

Although the Goldmann perimeter may be used either for static or kinetic testing, its main use is in the latter. Kinetic testing involves moving a target of fixed size and luminance from the periphery of the visual field towards the centre until it is seen. This strategy relies on the fact that, under the mesopic conditions of the test, the central visual field is usually more sensitive to a given target than the peripheral field. The process can be repeated for targets of different sizes and luminances: each target yields an isopter on the visual field chart which corresponds to a contour of the 'hill of vision' (Figs. 1.2 *a*, *b*, *c*).


Stimulus moved towards fixation point until seen
Fixation point
Record of first response
Process repeated along another meridian
Fixation point



Figures 1.2a, b and c. *Diagram of kinetic strategy and the construction of an isopter*

The benefit of the Goldmann perimeter is that an experienced operator can perform fast, flexible testing for a variety of visual field abnormalities in subjects who might not be able to produce reliable tests under the more rigorous conditions imposed by an automated perimeter. Thus, it is invaluable in the determination of the presence, absence, or progression of visual field defects in patients with neurological disease. However, the versatility of manual perimetry may also limit its effectiveness, since it is a source of variation and bias. The examiner may neglect areas of the field which are not thought to be important. The pattern of visual field abnormalities may be 'forced' to comply with preconceived ideas. The result of the test is highly dependent upon the level of training of the examiner. Furthermore, kinetic testing gives poorly reproducible results in the central field and at the edges of gradually deepening scotomas such as those found in glaucoma.(78a) For these reasons, when quantifiable, reproducible results are required, automated perimetry is used.

1.2.3 Automated perimetry

Automated or computerised perimetry differs from manual perimetry in that the decision-making process or strategy of the test is controlled by a computer instead of a human examiner.[79] Automated perimetry has largely replaced manual perimetry in the detection and monitoring of glaucoma. This change has improved the evaluation of visual fields in glaucoma patients.[80-84]

1.2.3.1 Development

The first widely used automated perimeters were the Octopus[85] and Competer.[86] Other early machines were the Scoperimeter[87] and the Peritest.[88] Reviews of these early developments are given by Fankhauser[89] and Wild.[90]
The Humphrey Field Analyzer, on which this thesis centres, was introduced during the 1980s.[91] An account of contemporary perimeters is given by Lachenmayr and Vivell[71] and Henson.[3] The latest versions of the Humphrey Field Analyzer (Humphrey Instruments Inc., San Leandro, CA) and the Octopus perimeter (Interzeag, AG Schlieren-Zurich, Switzerland) are currently the most widely used instruments in research and clinical practice.

1.2.3.2 Static testing

The term 'static testing' encompasses a variety of disparate testing strategies, all of which share the feature that the stimuli presented to the subject do not move. Static testing attempts to estimate the luminance sensitivity at fixed test locations, rather than moving a stimulus until it is seen. The combination of automated perimetry with static testing has the advantages that it is operator independent and yields numerical data relating to the spatial co-ordinates of the test locations and sensitivities at those locations: these data are amenable to sophisticated statistical analysis, unlike the results of kinetic perimetry. Static testing strategies may be divided into full threshold strategies and suprathreshold strategies.

1.2.3.2 (i) Full threshold strategies

If kinetic testing may be imagined to give rise to a contour map of the hill of vision, full threshold static testing produces a grid of numbers representing estimates of the height of the hill at various fixed points. The most widely used strategy for estimating the sensitivity at a given test location is the 4-2 dB 'two reversal staircase' bracketing strategy (Fig. 1.3). An initial stimulus is presented whose brightness depends on the results of tests on nearby points, or on age-matched normal values. If this is seen, the next presentation at that location is 4 dB less intense. If this is also seen, the following stimulus at that location is reduced by a further 4 dB in intensity, and so on until the subject fails to see a stimulus presentation. This is the 'first reversal'. The stimuli following this are increased in intensity by 2 dB each time until the subject again reports a stimulus as seen. This is the 'second reversal'. The sensitivity is estimated as the mean of the final and the penultimate presentation intensities. If the initial presentation is not seen, intensities of subsequent presentations are increased by 4 dB until one is seen ('first reversal') then decreased by 2 dB until one is missed ('second reversal').

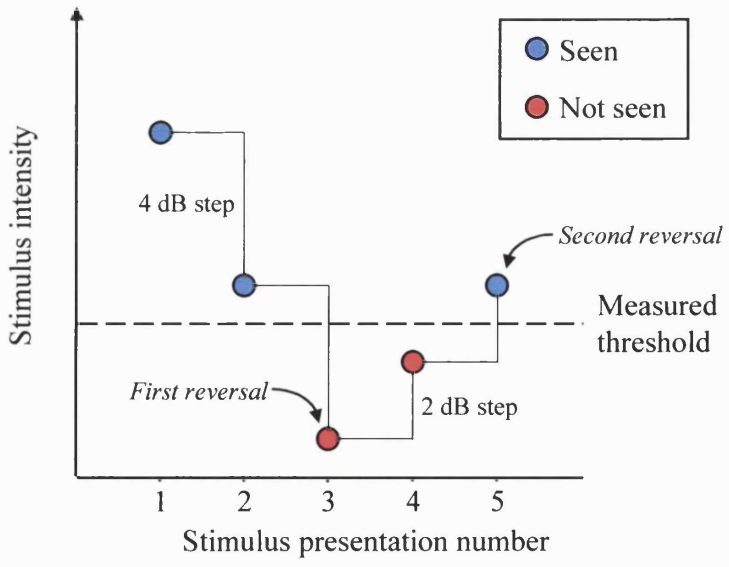


Figure 1.3. *Diagram of 4-2 dB 'two reversal staircase' bracketing strategy*

1.2.3.2 (ii) Suprathreshold strategies

Pure suprathreshold strategies are simpler and usually faster than full threshold strategies. Instead of ascribing a numerical estimate of sensitivity to a test location, they merely record whether it is normal or abnormal. This is done by presenting a stimulus calculated to be slightly more intense than the patient's sensitivity threshold. This calculation requires an estimate of the patient's threshold: the various methods by which this is obtained provide the names for the various suprathreshold strategies. The *fixed intensity suprathreshold test* assumes that all patients have a similar threshold. Thus the same test intensity is used for all patients. The *age-related suprathreshold test* takes account of the fact that sensitivity declines with age by approximately 1 dB per decade, and uses a table of normal data along with the patient's age to calculate the test intensity. The *threshold-related suprathreshold test* precedes the suprathreshold test with a determination of the patient's threshold, usually by means of a brief full threshold examination using a greatly reduced set of test points. The test intensity is then calculated relative to this threshold.

The amount by which the stimulus of a suprathreshold test is more intense than the patient's threshold, the *suprathreshold increment*, must be such that a 'missed' point truly represents abnormality (i.e. the stimulus must not be too dim) and a 'seen' point truly represents normality (i.e. the stimulus must not be too bright). Most perimeters use a suprathreshold increment of between 4 dB and 6 dB. In addition, most suprathreshold strategies in use compensate for the relatively reduced sensitivity of the peripheral visual field compared to the central field under the conditions in which perimetry is performed (lower photopic). These *eccentrically compensated* strategies derive an approximation of the hill of vision from their estimates of the patient's threshold, and thus vary the stimulus intensity to maintain a constant suprathreshold increment for all test locations.

1.2.3.2 (iii) Full threshold versus suprathreshold strategies

Full threshold strategies provide more detailed information than suprathreshold strategies since they indicate the depth of scotomas rather than merely their

presence or absence. However, full thresholding is more time-consuming, which has important implications for resource allocation. It is also more wearisome for the patient, which can affect the reliability of test results. Many patients exhibit pronounced learning effects: their performance in full threshold testing improves with repeated attempts (see 1.3.2.2).

For these reasons, suprathreshold testing is best used when a rapid distinction between normality and abnormality is required, whereas full thresholding is better for the long-term follow-up of patients whose visual fields are known to be abnormal and may be slowly deteriorating. For example, suprathreshold testing is commonly used to 'screen' for glaucoma, but patients with established glaucoma are usually monitored with full threshold tests.

1.2.4 Humphrey Field Analyzer

Data for the studies in this thesis was obtained exclusively from patients tested with the Humphrey Visual Field Analyzer 630 (Humphrey Instruments Inc., San Leandro, CA). This machine is described in detail by Haley[92] and Werner[93] and is shown in Figure 1.4.

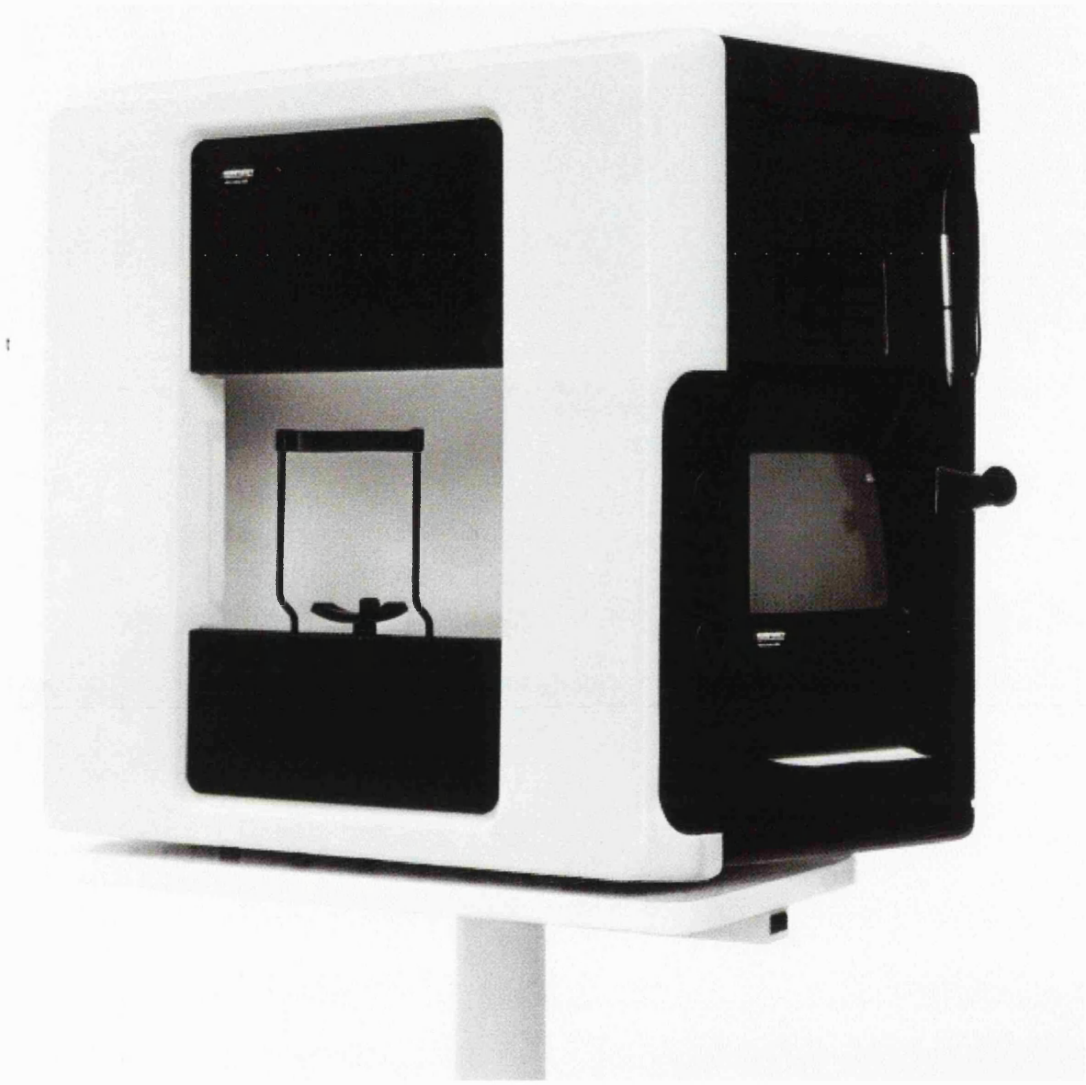


Figure 1.4 *Humphrey Field Analyzer 630 (Humphrey Instruments Inc., San Leandro, CA).*

1.2.4.1 Test patterns

The most commonly used testing programs for monitoring the visual fields of glaucoma patients are the 24-2 and the 30-2, which perform full threshold estimation (see 1.2.3.2 (i)) at locations within the central 24° and 30° of the field respectively. The tested locations are 6° apart, with none lying on the horizontal or vertical meridian. It has been suggested that routine testing at eccentricities beyond 30° is unproductive in POAG.[94] However, the nasal step is a notable exception to

this.[95] Two nasal locations within the standard 24-2 program lie within this area out to 30°.

The program which is most frequently used for testing the binocular visual field is the Esterman Binocular Functional Test. This is a suprathreshold test (see 1.2.3.2 (ii)) which extends to approximately 75° from fixation on the horizontal, 35° above fixation and 60° below fixation. This test is discussed further in Chapter 5.

1.2.4.2 Graphical display of results

The Humphrey Field Analyzer produces printouts which contain numerical and graphical data which varies depending on the exact test performed. The most important features of the printouts are described below.

1.2.4.2 *(i) Numerical threshold data*

Numbers representing the thresholds measured at all the test locations are displayed diagrammatically as in Fig. 1.5. The intersection of the two axes represents fixation, and each number represents the threshold at that test location. For this particular figure (the Humphrey 24-2 strategy), the separation of the locations, both vertically and horizontally, is six degrees.

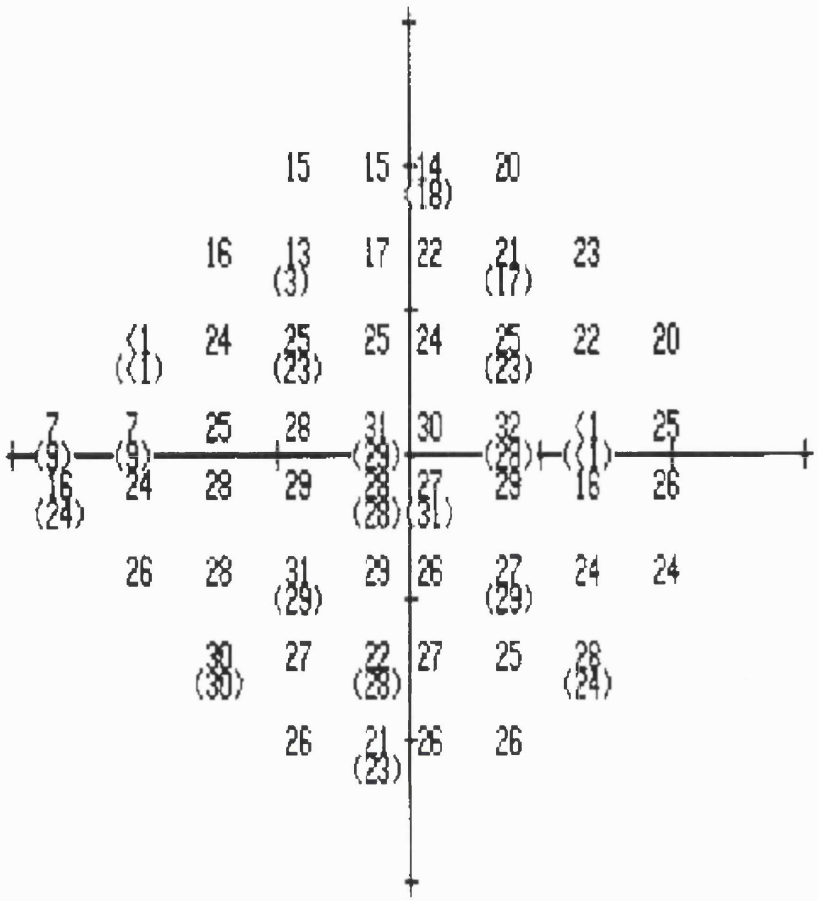


Figure 1.5 *Humphrey Field Analyzer raw sensitivity plot (24-2 strategy)*

Numerical displays such as that in Fig. 1.5 may show either the 'raw' thresholds at each location, or they may show the difference in sensitivity between the measured threshold and a threshold obtained from a normal, age-matched database. For the Humphrey Field Analyzer, this latter display is called the Total Deviation plot.[96 97] The Humphrey Field Analyzer also displays a numerical Pattern Deviation plot: this is a plot in which any generalised shift away from the age-matched norm is removed, and only more localised defects are shown.

1.2.4.2 *(ii) Gray scales*

Although the numerical displays mentioned above provide much useful information, they are rather dry and difficult to interpret at a glance. The gray scale provides an image which is more readily understood: thresholds are first interpolated to give a denser grid of values than the numerical display, then these are coded according to a key (Fig. 1.6*b*) and displayed as in Fig. 1.6*a*, which is the gray scale corresponding to the numerical display in Fig. 1.5. The Humphrey Field Analyzer also provides gray scale displays for the Total Deviation[98] and Pattern Deviation plots. The appearance of the gray scale is critically dependent on the method of interpolation used to generate it.[99]

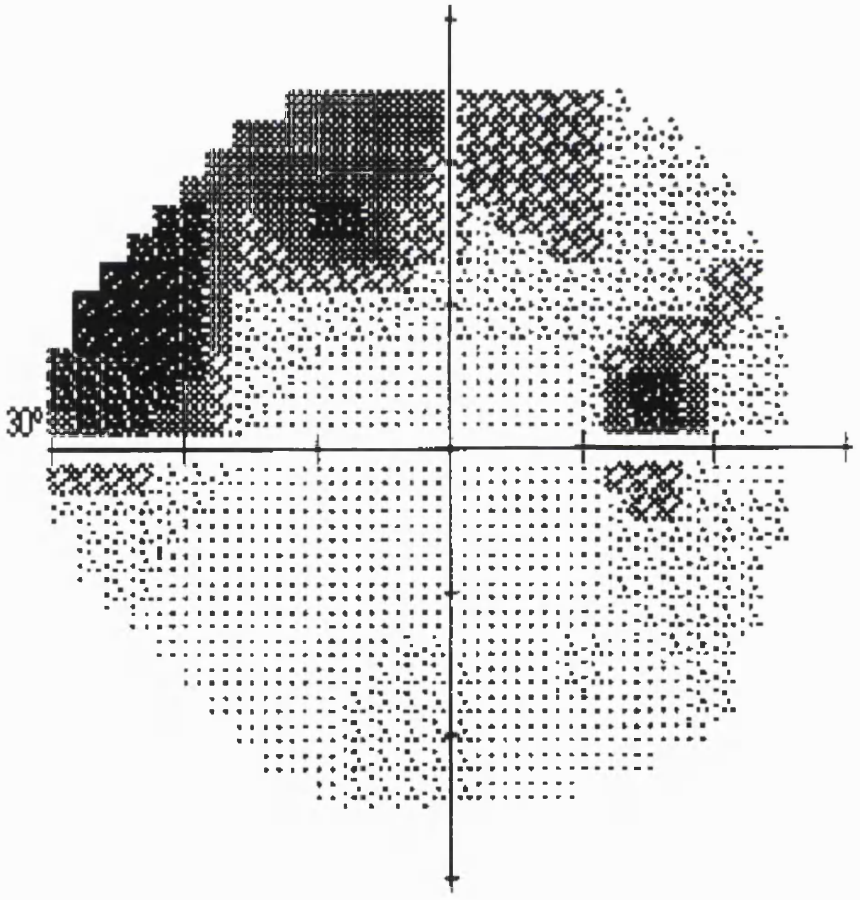


Figure 1.6a *Humphrey Field Analyzer gray scale derived from the numerical output of Figure 1.5*

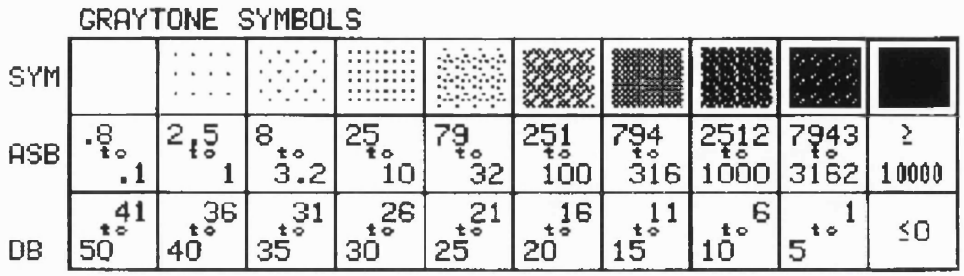


Figure 1.6b *Humphrey Field Analyzer gray scale legend*

1.2.4.2 *(iii) Suprathreshold tests*

Since only one of two categories ('seen' or 'not seen') is attached to each test location, numerical displays are not appropriate: each location is instead represented by a symbol indicating whether the stimulus at that location was seen or not (see Figure 5.1).

1.2.4.3 Global indices

Information relating to the amount of visual field loss, and whether the loss is generalised or focal, is encapsulated in a set of summary measures known as global indices. These were first defined for the Octopus perimeter as mean defect, short-term fluctuation, loss variance and corrected loss variance.[100 101] The corresponding measures for the Humphrey perimeter are mean deviation, short-term fluctuation, pattern standard deviation and corrected pattern standard deviation.[96] The Octopus indices will be described first as they are more readily understood and the Humphrey indices are more complex versions of them.

Mean defect is calculated by taking the difference between the threshold measured at each test location and a value obtained from a normal age-matched database. The mean defect is the mean of these differences. It is more sensitive to generalised or diffuse loss than to small, focal scotomas. The validity of the mean defect is critically dependent upon the validity of the comparison between the test data and the normal data set. In other words, the sample used to compile the normal database must be truly representative of the population from which the subject under test is

drawn. This comment applies to all measures which involve a comparison with a 'normal' database.

Short-term fluctuation is a measure of the intratest variation. It is calculated by testing a set of locations more than once during each visual field examination and analysing the variance of these repeat measures.

Loss variance is an index of the amount of focal loss in the field. It depends on the fact that, if there are areas of the field with large defects (i.e. differences from normal values) and other areas with small defects, the variance of the defects will be larger than if the field were uniformly normal or uniformly depressed.

Since variability is inherent in visual field testing, there will always be some loss variance. The concept of corrected loss variance was introduced to obtain a zero-based measure of focal loss. Corrected loss variance is calculated by subtracting from the value of loss variance a measure of the subject's intratest variability. This measure is the square of the short-term fluctuation.

The global indices provided by the Humphrey Field Analyzer have a similar theoretical basis to the Octopus indices, but there are some important differences. In the Humphrey Field Analyzer, the formula for the calculation of a global index includes a term which accounts for the normal variance at each location. The Humphrey indices for focal loss, pattern standard deviation and corrected pattern standard deviation, are related to the Octopus loss variance and corrected loss variance respectively. However, the Humphrey indices are measures of standard deviation (square root of variance) as their names suggest, and they incorporate measures of normal variance at each location as mentioned previously. In addition, corrected pattern standard deviation includes a constant factor to adjust for the non-uniform fluctuation pattern across the field. The greater complexity of the Humphrey indices does not greatly affect their usefulness as summary measures of visual field behaviour.

The formulae for the Humphrey Field Analyzer global indices are:

Mean deviation (MD)

$$MD = \left\{\frac{1}{n}\sum_{i=1}^{n}\frac{(x_i - N_i)}{s_{1i}^2}\right\} \Big/ \left\{\frac{1}{n}\sum_{i=1}^{n}\frac{1}{s_{1i}^2}\right\}$$

where x_i is the measured threshold and N_i is the normal reference threshold at point i, and s_{1i}^2 the variance of normal field measurements at point i. The number of test points is denoted by n.(96)

Pattern standard deviation (PSD)

$$PSD^2 = \left\{\frac{1}{n}\sum_{i=1}^{n}s_{1i}^2\right\} \times \left\{\frac{1}{n-1}\sum_{i=1}^{n}\frac{(x_i - N_i - MD)^2}{s_{1i}^2}\right\}$$

Short-term fluctuation (SF)

$$SF^2 = \left\{\frac{1}{10}\sum_{j=1}^{10}s_{2j}^2\right\} \times \left\{\frac{1}{10}\sum_{j=1}^{10}\frac{(x_{j1} - x_{j2})^2}{2 \times s_{2j}^2}\right\}$$

The first and second measured thresholds are denoted by x_{j1} and x_{j2} respectively. The normal intra-test variance in point i is denoted by s_{2j}^2.

Corrected pattern standard deviation (CPSD)

$$CPSD^2 = PSD^2 - k \times SF^2$$

The constant k (>1) is used to adjust for the non-uniform fluctuation pattern.(96)

Confidence intervals for normal subjects have been evaluated for all four indices.(96) If a calculated value falls outside these limits a probability statement is printed next to the value of the global index concerned.

1.2.4.4 The Glaucoma Hemifield Test

The Glaucoma Hemifield test algorithm is provided by the Statpac statistical program for the Humphrey Field Analyzer. It tests a single visual field for the presence of a glaucomatous defect.[102] Visual field test locations are grouped together in five corresponding pairs of sectors based on the anatomy of the retinal nerve fibre layer. These sectors are symmetrical about the horizontal midline. Each sector is compared to its counterpart in the opposite hemifield in order to detect field loss that is asymmetric about the horizontal meridian. Deviations from the age-corrected normal threshold in the most sensitive areas of the field are used to detect overall glaucomatous loss. Fields are classified as outside or within normal limits, borderline, or as having a general reduction in retinal sensitivity.

1.3 Change in visual field

The detection and quantification of change in visual fields is one of the most important and problematic areas of glaucoma management. An early, accurate measure of whether a field series shows progressive damage is essential: significant deterioration is likely to prompt a change in treatment, whereas confirmed stability provides more convincing reassurance than lowered intraocular pressure (IOP) alone that therapy is successful.

1.3.1 Measurement of change

1.3.1.1 Automated *versus* manual perimetry

As already mentioned in 1.2.2.2, a benefit of manual perimetry is that an experienced practitioner can perform fast, flexible testing for a variety of visual field abnormalities in subjects who might not be able to produce reliable tests under the more demanding conditions imposed by an automated perimeter. Manual perimetry has been used to attempt to deduce the overall rate of progression in

open-angle glaucoma estimated from cross-sectional prevalence of visual field damage in a large, population based study (the Baltimore Eye Survey).[103]
However, for reasons already discussed in 1.2.2.2, the subjective nature of manual perimetry also limits its effectiveness, since it is a source of variation and bias. Thus, over the past decade, visual field series consisting of full threshold tests obtained by automated perimetry have become the standard starting point for attempts to estimate visual field change in glaucoma.

1.3.1.2 Clinical judgement *versus* numerical analysis

The most commonly used method of deciding whether a visual field series shows progression or not is for a clinician to inspect the series visually and use clinical judgement to form an opinion. Unfortunately, this approach is flawed. Human observers, even experienced ones, are not able to detect visual field progression reliably using clinical judgement alone.[104] One possible reason for this is that the standard output of most automated perimeters contains insufficient information relating to progression or stability. When clinicians are presented with visual field series which have been processed so as to highlight areas of possible deterioration, the level of agreement about progression is greater.[105] Thus, although clinical judgement is by definition at the root of clinical decision making, the simple visual assessment of field series presented as the standard output of an automated perimeter is an inadequate basis for this judgement. Rather, clinical decisions concerning visual field progression should be based on the results of computerised numerical change analysis of visual field series. The most widespread of these analyses are described below.

1.3.1.3 Summary measures

One group of methods relies on estimates of change in summary measures of the field such as regression analysis of the mean defect value,[17] mean deviation,[106] other global measures,[106] measurement of whole-field and quadrantic sensitivity losses,[107] and trend and regression analysis of various estimates of the sensitivity of the whole field or parts of it.[20 21 108] The Glaucoma Hemifield Test has been

used to detect incident field loss among patients with elevated intraocular pressure.[109] However, the analysis of summary measures, whether based on the whole field or on clusters of points within it, has been found to be 'remarkably poor'[110] and 'of little value'[111] in detecting glaucomatous change. Summary measures largely or completely ignore the detailed spatial information contained within computerised field tests and are insensitive to early localised change.[112] Furthermore, different regions of the visual field may deteriorate at different rates.[21 113 114]

1.3.1.4 Pointwise measures

Techniques which evaluate progression on a point-by-point basis avoid the problems with summary measures described above. They may be divided into two categories: event analyses and trend analyses.

Event analyses rely on detecting a significant change from an established baseline. For example, the Collaborative Normal Tension Glaucoma Study Group initially specified an endpoint, based on sensitivity loss, at which patients would be said to have shown unequivocal deterioration. However, the authors noted a surprisingly large number of patients reaching the endpoint. Statistical analysis revealed that the endpoint chosen would be expected to lead to a false diagnosis of progression 57% of the time. In order to correct for this a requirement for progression to be confirmed on multiple repeat tests was introduced.[115] The most recent report from the Group gives the endpoint as follows: 'a follow-up visual field was said to show progression relative to baseline if it contained 2 or more points that had changed by at least 10 dB relative to the average baseline values for these points; these 2 progressing points had to be adjacent, both could not be peripheral, both could not cross the nasal meridian, and the sensitivity at each deteriorating point had to be less than the minimum of the values of this point in each of the three baseline visual fields. In addition, progression was also deemed to have taken place if at least one of the innermost 4 points showed at least a 10 dB deterioration relative to its average value at baseline, with a value that was less than its minimum value in each baseline field. Progression was considered to be confirmed when four

of five consecutive follow-up fields showed progression relative to the baseline fields, with at least one non-peripheral progressing point (or the one central point) being common to all four fields.'[116]

Trend analyses do not construct a baseline. The behaviour of the visual field is analysed and progression is diagnosed if a significant tendency to deteriorate is detected. When this is done on a pointwise basis the technique of linear regression of sensitivity on time is often used. Test locations are labelled as showing progression if a significant negative regression line is calculated. For example, individual test locations from glaucoma patients have been found to deteriorate significantly at rates of between 1 and 5 dB per year[106 117] when visual field testing is performed annually. The effect of the frequency of visual field testing on the ability to detect glaucomatous visual field deterioration using pointwise linear regression is investigated in Chapter 2.

The differences between event analyses and trend analyses may be further appreciated from a consideration of two commercially available software packages for the Humphrey Field Analyzer: Statpac 2 (Humphrey Instruments, Inc., San Leandro, CA, USA) and PROGRESSOR (Institute of Ophthalmology, London, UK).

1.3.1.4 (i) Statpac 2

The Statpac 2 Glaucoma Change Probability analysis[118] performs a pointwise event analysis. It is the 'native' statistical Glaucoma Change Probability software available as an add-on for the Humphrey Field Analyzer. The program uses the thresholds from the two most reliable of the first three fields in the series as a baseline. Each subsequent field is compared on a point-by-point basis to this baseline. Test locations are labelled with a black triangle (Figure 1.7) if $p < 0.05$ for the null hypothesis of no glaucomatous change compared to a database of stable glaucoma field series.

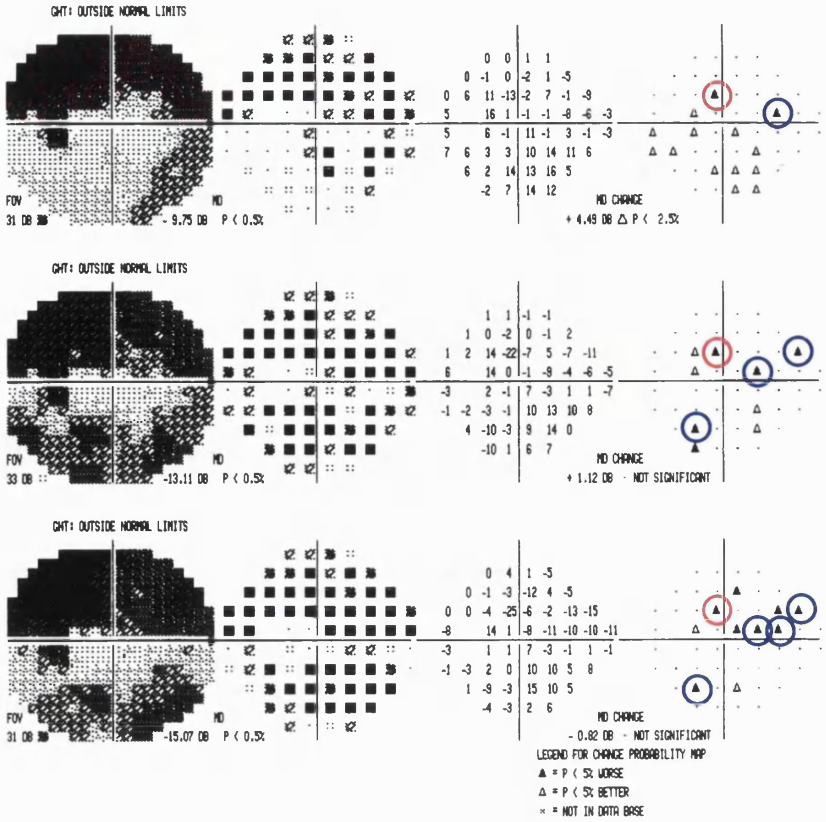


Figure 1.7 *Example of a section of the Statpac 2 Glaucoma Change Probability analysis. The test locations circled in blue are labelled as showing significant deterioration in two out of the three fields shown. The test location circled in red is labelled as showing significant deterioration in each of the three consecutive fields.*

1.3.1.4 (ii) PROGRESSOR

PROGRESSOR[119] is a trend analysis. It is a software package which analyses visual field progression using pointwise linear regression of sensitivity on time. The pointwise linear model has been demonstrated to provide a valid framework for detecting and forecasting glaucomatous loss.[120] This technique has been used for several years to investigate glaucomatous visual field change[121 122] and has recently been re-examined.[106] PROGRESSOR produces a cumulative graphical output as shown in Figure 1.8a. Each test location is shown as a bar graph in which each bar represents one test. The length of the bar relates to the depth of the defect (longer bars represent lower sensitivities) and the colour of the bar relates to the p value of the regression slope (Figure 1.8b). Thus undamaged locations are seen as series of short grey bars, damaged but stable locations are seen as long series of

long grey bars, and progressing locations are seen as series of progressively lengthening bars which change colour as the regression slope becomes more significant.

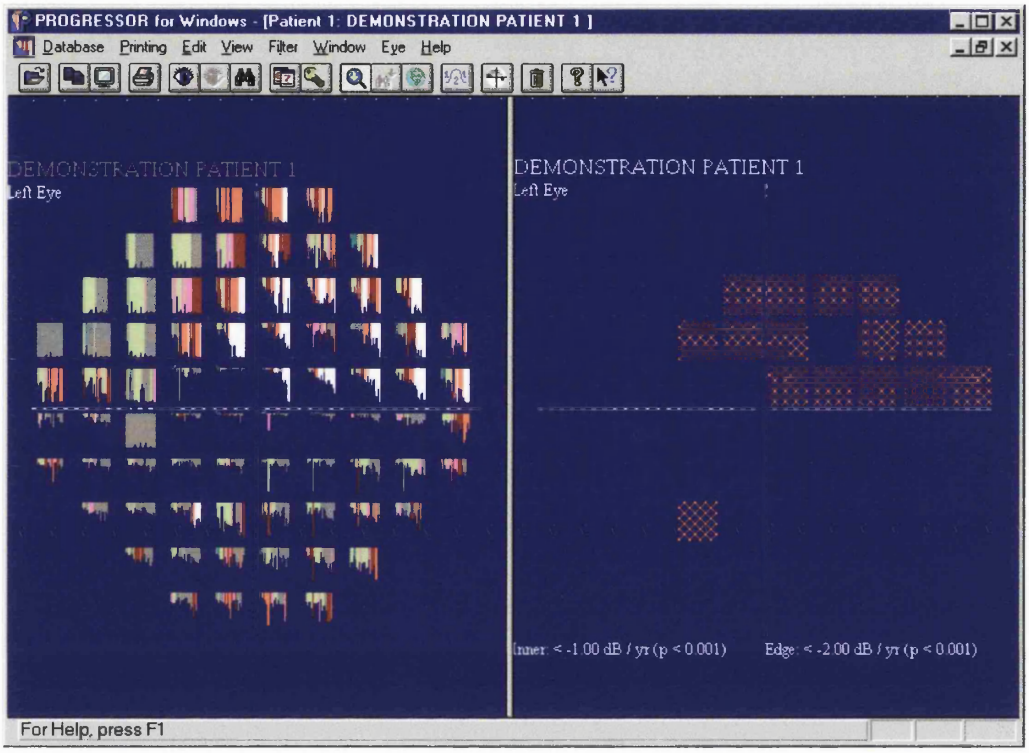


Figure 1.8a *Example of the PROGRESSOR output for the same visual field series as in Figure 1.7. The left pane shows the cumulative graphical output and the right pane shows the test locations which satisfy progression criteria. Note that the pattern of progressing points is similar to that of the circled points in Figure 1.7.*

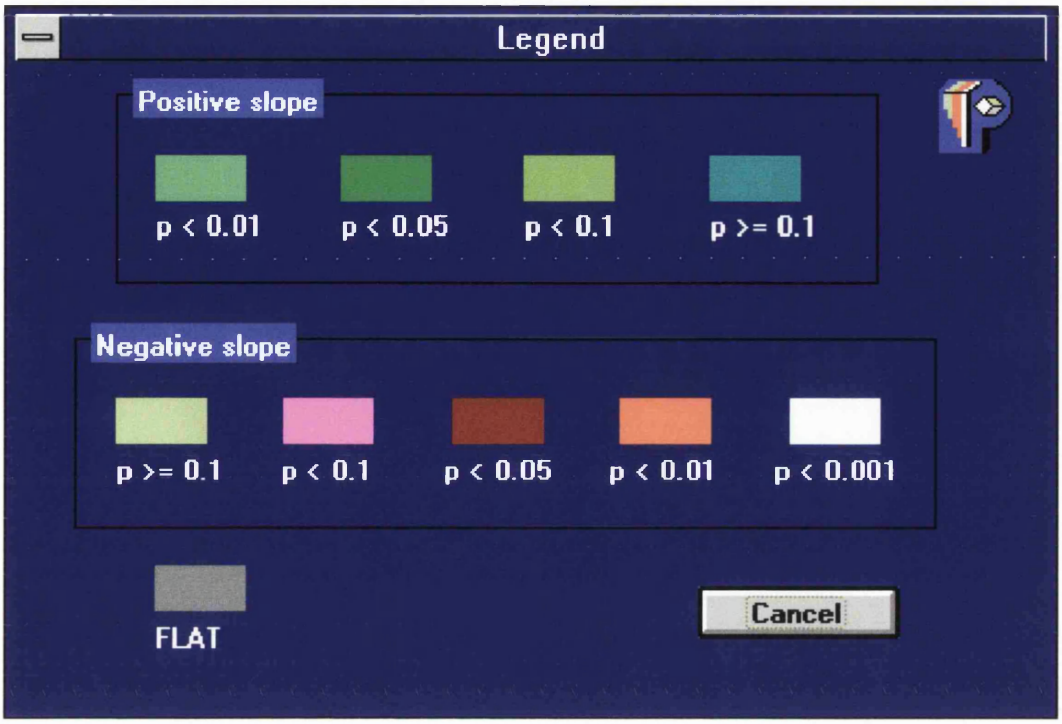


Figure 1.8b *PROGRESSOR legend. This legend relates the colour of a given bar in the PROGRESSOR bar graphs to the p value of the regression slope for that test.*

1.3.1.4 (iii) Comparison of event analysis and trend analysis

Both Statpac 2 and PROGRESSOR provide sensitive methods for the detection of glaucomatous visual field deterioration, whether widespread or focal, but each algorithm has its limitations.

Statpac 2 relies on its database of normal values being a representative sample of the population from which the patient under test is drawn. Only data from the two baseline tests and the test under consideration is analysed: data from any intervening tests is ignored. This is of particular concern because the baseline tests are usually amongst the first attempted by the patient, and thus often the least reliable. By its very nature as an event analysis, Statpac 2 may give an indication that change has occurred but gives no information about the rate of change. This measure of rate is a prerequisite if prevention of clinically significant visual loss is to be achieved since it is through knowledge of the speed of deterioration that the degree of future impairment of visual function may be anticipated. If the rate of

visual field loss suggests that the patient will be visually disabled within their lifetime appropriate therapeutic measures may be instituted.

PROGRESSOR treats individual test locations as statistically independent from one another. This is clearly not the case. Thus points which are labelled as showing statistically significant deterioration must be interpreted judiciously: statistical significance is not analogous to clinical significance. Owing to its reliance on linear trend analysis, PROGRESSOR might be expected to be relatively insensitive to sudden, stepwise decay (though recent work[123] suggests this is not the case).

Statpac 2 and PROGRESSOR have been found to show good agreement about progression or stability, when the appropriate progression criteria are used.[124] Other work has also examined the level of agreement between pointwise linear regression and Statpac 2 in glaucoma.[125] The two techniques were found to show good agreement about which test locations were progressing, if these locations were deteriorating at more than five times the normal age-related decline in sensitivity. If the locations deteriorated at a more moderate rate, however, Statpac 2 failed to detect progression which was diagnosed by pointwise linear regression. The authors ascribed this finding to the fact that locations which have lost a moderate degree of sensitivity have more test-retest variability than points with either high or low sensitivity. These moderately depressed points are thus less easily detected by Statpac 2, which relies on a point falling outside the test-retest confidence limits of the baseline before progression is diagnosed. Conversely, the authors found a number of test locations which showed a high degree of fluctuation throughout the visual field series and could not be adequately described by pointwise linear regression. Statpac 2 labelled a proportion of these locations as progressing. Pointwise linear regression and other trend analyses have been found to agree more closely with human observers than an event-based glaucoma change analysis.[127]

The relative speed of Statpac 2 and PROGRESSOR in their ability to detect progression is investigated in Chapter 4.1.

1.3.2 Errors in measurement

1.3.2.1 Fluctuation

One of the greatest obstacles to the reliable detection of visual field change in glaucoma is the high level of inherent variability between tests. This is known as long-term fluctuation and is a particular feature of glaucoma.[128] The contribution of long-term fluctuation must always be borne in mind before accepting a single field test as showing definite deterioration compared to the previous tests in a series: in general, confirmatory tests should be sought before decisions about patient management are made (see 1.3.1.4 and Chapter 2).

A promising avenue of research is the application of digital image processing techniques, such as those used to process images obtained by magnetic resonance imaging (MRI), to reduce the 'noise' in the results of computerised perimetry. These spatial filtering techniques, such as Gaussian filtering, take account of the interdependence of neighbouring retinal test locations and have been shown to improve the repeatability of automated perimetry by a factor of two[129] and to increase the predictability of glaucomatous decay.[130] Whether this improved repeatability and predictability is gained at a cost of delay in the diagnosis of progression through the blurring of early, focal scotomas is investigated in Chapter 6.

New perimetric strategies such as the Swedish Interactive Threshold Algorithm (SITA) have also been demonstrated to reduce variability. The between-subject pointwise variability was found to be approximately 10% less for SITA than for conventional full threshold perimetry.[131] Within-subject variability is also reduced, resulting in narrower confidence intervals for normal variability.[132] This is most likely a result of the shorter test times associated with this algorithm and consequent reduced visual fatigue. Although narrower confidence limits would be expected to lead to earlier, more reliable detection of visual field deterioration, the SITA algorithm also yields higher pointwise light sensitivities than conventional

full threshold tests[132] and there is some evidence that scotoma size is reduced and that early defects may be missed.[133]

1.3.2.2 Learning effects

Some subjects show a marked increase in sensitivity from the initial test to subsequent tests. This tends to be more marked in the superior and peripheral parts of the field. Other subjects, however, do not show any learning effect and produce reliable tests from the outset. Most learning effects have disappeared after the first two visual field tests.[134] However occasionally learning effects may persist for many years, as shown in Figures 1.9 and 1.10. If learning effects are suspected, the results of visual field tests should be ignored until a stable baseline is achieved. Inclusion of 'pre-learning' fields in a series will delay the recognition of any subsequent deterioration by both event and trend analyses.

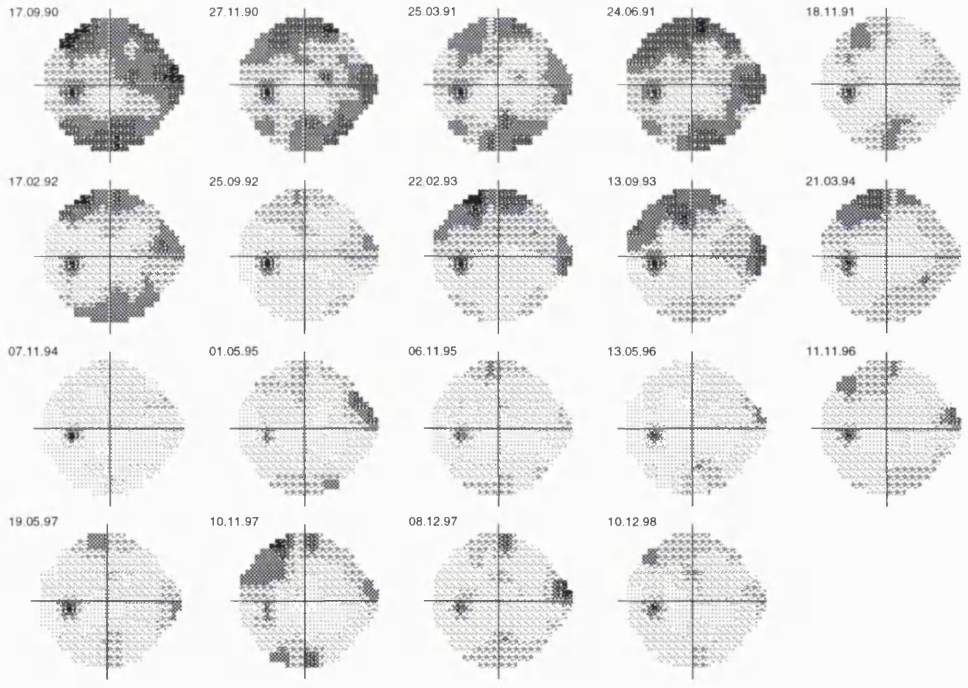


Figure 1.9 *A series of visual fields showing the persistence of learning effects over a period of 8 years.*

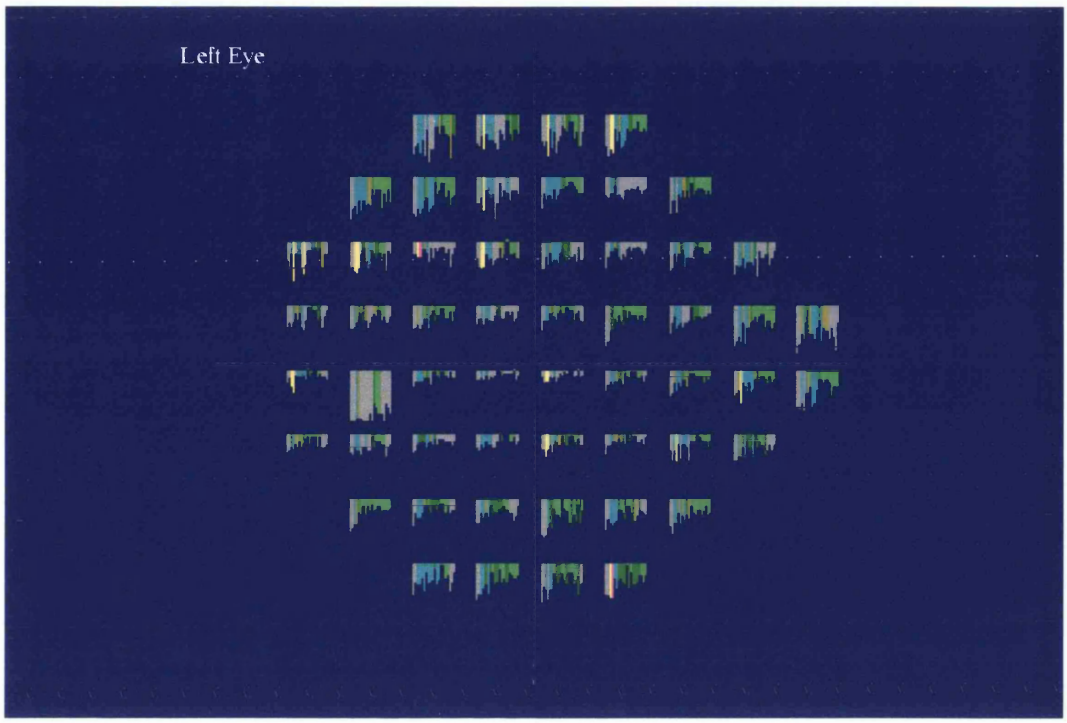


Figure 1.10 *PROGRESSOR output for the same visual field series as shown in Figure 1.9. The progressively shortening green bars indicate improvement in sensitivity over time, which in this patient is attributable to learning effects.*

1.3.2.3 Change criteria

The sensitivity and specificity of any method used to detect field change are critically dependent on the criteria used to differentiate progression from stability. For example, if an event analysis specifies a pointwise $p < 0.05$ probability criterion for the diagnosis of progression, in a 30-2 test grid consisting of 76 test locations, about four locations in each test would be expected to be labelled as progressing when they are actually stable (false positives). The originators of Statpac 2 suggest that a location be repeatedly ascribed a high probability of change over a series of consecutive fields before progression is diagnosed.[118] Another recommendation is that progression should depend on clusters of neighbouring points being labelled as progressing. These proposals will increase the specificity of the analysis at the expense of sensitivity. The change criteria recently reported by the Collaborative Normal Tension Glaucoma Study Group have been discussed in 1.3.1.4.[116]

Change criteria used for trend analyses such as pointwise linear regression should specify a slope criterion as well as a probability criterion: a significant regression line should demonstrate a specific minimum level of deterioration before a test location is labelled as progressing. This lessens the risk of false positives. Furthermore, if a slope criterion is not specified, the physiological age-related decline in sensitivity may be falsely labelled as glaucomatous progression.[125]

At present, there are no universally accepted progression criteria and it is unlikely that any single set of criteria will be applicable to the wide spectrum of disease seen in glaucoma patients and glaucoma suspects. In terms of practical management of the glaucoma patient, the results produced by any set of progression criteria must be examined within the clinical context.

1.3.2.4 Frequency of testing

Even sensitive, reliable methods such as pointwise linear regression are critically dependent on the frequency with which visual field tests are performed: progressive visual field loss cannot be detected unless it is adequately sought. When considering how often to measure the visual field of a glaucoma patient, factors such as the type of glaucoma, the number and results of previous tests, the age of the patient, recent changes in glaucoma therapy, the availability of perimetric resources and the ability of the patient to produce reliable tests must be borne in mind. Reliability is discussed further in 1.3.2.6.

The Collaborative Normal Tension Glaucoma Study Group has reported that an event type of analysis can be used to diagnose visual field progression in a timely, sensitive and specific way.[115] In this analysis, patients are followed every three months with visual field tests. If the criteria for suspected progression are met, the patient returns within one to four weeks for one or two confirmatory tests. If progression is confirmed in this way on two consecutive series of visits three months apart, true progression is diagnosed. However, this study protocol entails a high frequency of testing which would have profound resource implications if it were implemented on a routine basis, since it may involve three tests every three months. The most recent 'four-of-five' confirmatory testing protocol reported by

the Collaborative Normal Tension Glaucoma Study Group has been described in 1.3.1.4.[116]

Chapter 2 of this thesis examines whether performing visual field tests once a year rather than three times a year would lead to a clinically significant delay in the detection of progressive glaucoma.

1.3.2.5 Artefact

There are many factors that may cause an apparent worsening of the visual field which is not truly related to progressive glaucoma. The commonest and most troublesome of these is long-term fluctuation (see 1.3.2.1) but artefacts affecting the visual field, which are usually more easily remedied than long-term fluctuation, must also be considered.

1.3.2.5 (i) Pupil size

A reduction in pupil size results in a decrease in threshold values, especially in the peripheral visual field, and in an increase in the variability of threshold measures. Quantitative data on the effect of pupillary diameter on visual field sensitivity is available.[135-137] Normal variations in pupil size are not thought to be sufficient to influence perimetric sensitivity.[138 139] However, these effects are particularly relevant with regard to miotic therapy for glaucoma: for example, an apparent visual field progression may be found to coincide with the start of a course of pilocarpine drops. If this patient's pupil is dilated with a short-acting topical mydriatic administered before the visual field test (if this is clinically appropriate) the result will be free from the adverse effects of the miotic and a fairer comparison with previous tests may be made. Pupil size should be measured and recorded in the protocol of testing[92] to avoid artefacts when following glaucoma patients over time.

1.3.2.5 (ii) Refractive error

Uncorrected refractive error causes defocusing of the retinal image of the test stimulus. For example, 5 dioptres of simulated hypermetropic blur on a Goldmann

size III stimulus reduces the central sensitivity by approximately 6 decibels. Quantitative data on this phenomenon is available.[140 141] Refractive errors greater than 1.00 dioptre should be corrected. Many perimeters use a testing distance for which a presbyopic correction will also be necessary. Unfortunately, the correction of refractive errors itself poses problems. Many perimeters incorporate a holder for a standard trial lens in order to correct the eye under test, but the small diameter of these lenses may mean that the edge of the lens may encroach upon the visual field and cause a lens rim artefact.[3] This appears as a defect at the edge of the central visual field. It does not usually respect either the horizontal or vertical meridians but may mimic a nerve fibre bundle defect. Lens rim artefacts are more common in the elderly and in hypermetropes. They may be avoided by ensuring that the patient's eye is as close to the correcting lens as possible and well centred behind the lens. Patients' own spectacles should be used if possible (unless they are bi-, multi-, or varifocal or inappropriate to the testing distance) since they give a larger field of view than a trial lens.

1.3.2.5 (iii) Lid / brow artefact

Apparent losses in the superior visual field may result from slight ptosis, prominent eyelashes,[142] dermatochalasis[143] or prominent brows. Incorrect positioning of the patient at the perimeter may be a contributory factor. These artefacts may be extremely difficult to distinguish from incipient arcuate scotomas. If the cause is ptosis, however, the problem is easily solved: if the lid is gently lifted with adhesive tape for the duration of the test, so as to abolish the ptosis but still allow the patient to blink fully, the artefact disappears.

1.3.2.5 (iv) Media opacities

Opacities in the lens (or posterior capsule following extracapsular cataract surgery) may cause progressive visual field deterioration. Media opacities usually cause a generalised reduction of sensitivity across the whole field: this may be difficult to distinguish from diffuse glaucomatous loss. Occasionally localised lens opacities may cause focal visual field defects but these are rarely as well-defined as nerve

fibre bundle defects. Examination of the ocular media will reveal any opacities dense enough to affect the visual field.

The Collaborative Normal Tension Glaucoma Study Group has reported that the data from patients who developed cataract during the trial had to be removed from the analysis before the protective effect of IOP lowering on visual field deterioration in normal tension glaucoma could be demonstrated.(116) The Group also attempted to control for the effects of cataract on Mean Deviation by adjusting the individual patients' Mean Deviation values for the corresponding readings of foveal sensitivity by regression analysis. This was based on the assumption that changes in foveal sensitivity primarily reflect cataract formation rather than the effects of glaucoma. Before these adjustments were made, a significantly different change in Mean Deviation from the time of randomisation to the time of IOP stabilisation was found between the treated and untreated groups. After the adjustments, no difference was found.

1.3.2.5 (v) Macular disease

Advancing age-related macular degeneration may produce field changes reminiscent of progressive glaucomatous paracentral scotomas but this rarely leads to any clinical confusion, as examination of the macula readily reveals the true source of the visual field loss.

1.3.2.6 Reliability

Before a field test may be included in any glaucoma change analysis, some account must be taken of its reliability. This involves attention to the factors mentioned in 1.3.2.5. In addition, automated perimeters calculate reliability indices which are shown on the output display. For example, the Humphrey Field Analyzer calculates fixation losses, false positives and false negatives. Fixation losses are measured by the Heijl-Krakau technique: once the position of the physiological blind spot has been ascertained, stimuli are presented within this area. Any reported as seen represent inaccurate fixation. False positives are recorded when the machine behaves exactly as if a stimulus were about to be presented, but none is, and the

subject reports the (non-existent) stimulus as seen. False negatives are detected when a stimulus known to be above the threshold at a particular location is presented at that location and the subject does not record the stimulus as seen.
Visual fields which have a large proportion of fixation losses, false negatives, or especially false positives are likely to be unreliable. The Humphrey Field Analyzer displays a 'low patient reliability' message to alert the examiner to this. These messages should be interpreted with caution, however, as there is no proven association between machine-generated reliability indices and actual patient reliability. If the machine fails to determine the location of the physiological blind spot correctly, or if the patient's head tilts or moves out of position during the test, the Heijl-Krakau technique will record a large number of fixation losses even though the subject's fixation is steady.[144] Furthermore, early glaucoma may be associated with an increased proportion of false negatives in some patients: in these cases false negatives are an index of disease rather than poor reliability.[144]

1.3.2.7 Dynamic range

All perimeters are restricted in the maximum brightness of stimulus possible. Once a test location deteriorates towards a level of sensitivity at which even the brightest stimulus produced by the machine elicits no response, further deterioration cannot be measured as the sensitivity of the location is beyond the dynamic range of the instrument. If this process occurs during follow-up, then only the measurable portion of the location's behaviour is available for analysis: the data is effectively censored. This may explain the finding that locations which have a low initial sensitivity are rarely detected as deteriorating by glaucoma change algorithms.[114]

1.3.3 Results of treatment

Glaucoma treatment is, at present, aimed almost exclusively at lowering intraocular pressure. This rationale is based on the observation that higher intraocular pressure tends to be associated with more severe disease. Cross-sectional studies indicate that an intraocular pressure of greater than 16 mmHg is associated with a higher rate of visual field progression in POAG.[145] For this reason 16 mmHg is often

specified as a desirable target pressure in POAG. A large prospective trial (the Early Manifest Glaucoma trial in Malmö) is currently in progress to evaluate the effects of treatment on visual field progression in POAG, but results are not yet available. Similarly, the ongoing Ocular Hypertension Treatment study has not yet reported on whether topical ocular hypotensive medications prevent or delay the onset of glaucoma in ocular hypertensive patients, though it has stressed the importance of confirmatory retesting when using an event analysis: 88% of follow-up visual fields showing abnormalities based on the Glaucoma Hemifield Test or Corrected Pattern Standard Deviation were not confirmed as abnormal on subsequent retesting.[146] However, longitudinal treatment studies including visual field analysis have been conducted in normal tension glaucoma patients. Although some of these studies use event analysis and some use trend analysis, they consistently indicate that effective lowering of intraocular pressure retards progressive visual field damage.[116 147-150]

Some of the newer therapeutic agents in glaucoma, such as brimonidine, have been postulated to exert a neuroprotective effect on the optic nerve. So far, these theories have been validated on an anatomical basis by means of optic nerve crush experiments in animals.[151] It remains to be seen whether these findings will translate into a practical benefit for the glaucoma patient. The measure of this will be by the reliable detection of visual field improvement: this will depend on methods such as those described above.

CHAPTER 2. FREQUENCY OF VISUAL FIELD TESTING

2.1 Abstract

Background: This study was undertaken to determine whether the interval between visual field tests affects the ability to detect progressive glaucomatous field loss.

Methods: 119 retinal locations which were deteriorating significantly by ≥1dB/year (untreated normal tension glaucoma patients: 6 eyes) were studied. Analysis was repeated using 'thinned' fields: 1 field per year instead of the complete 3 per year over 4 years.

Results: The 'thinned' fields only identified 45.4% of the deteriorating points over the 4 year period. Furthermore, there was a mean delay of 1.10 years in detection ($p < 0.01$).

Conclusions: Less frequent visual fields detect fewer progressing locations and detect them later.

(Viswanathan AC, Hitchings RA, Fitzke FW. How often do patients need visual field tests? *Graefes Arch Clin Exp Ophthalmol* 1997;235:563-568.)

2.2 Introduction

Although clinical decisions about how often to measure the visual field of a glaucoma patient are influenced by many factors (as described in 1.3.2.4), one of the commonest and most important reasons for repeatedly measuring visual fields is in order to detect progressive damage as early as possible. It is to this aim that the study described in this chapter is directed.

With respect to point-by-point analysis of sequential fields, it has been demonstrated that the most reliable estimates of progressive glaucomatous visual field decay are obtained by using a linear fit, i.e. postulating a constant reduction of sensitivity over time for each retinal location tested.[120] Thus it would seem self-evident that increasing the frequency of visual field tests would provide a reliable estimate of the rate of decay more quickly and therefore lead to the detection of progressing retinal locations sooner. This has been assumed[71 152] but not directly addressed.

However, a significant degree of variation exists both within a visual field test (short-term fluctuation, SF)[101] and between tests (long-term fluctuation, LF).[153] These contribute 'noise' to the underlying 'signal': both SF and LF are greater in patients with glaucoma than in normals.[128] When attempting to estimate the underlying behaviour of a signal in a noisy environment from a series of samples, the accuracy of the estimate increases as the interval between samples falls and the number of samples grows. However, it is possible that if the frequency of visual field testing rises above a certain level the estimate of visual field decay will cease to improve significantly since noise rather than signal will predominate. If this effect occurred at the frequencies of visual field testing commonly used in clinical practice for the follow-up of glaucoma patients (i.e. 4 months to 1 year) it would have important implications for cost-benefit analysis. The aim of this study was to ascertain whether increasing the frequency of visual field testing from 1 per year to 3 per year provides a useful improvement in the detection of progressing points.

2.3 Materials and methods

2.3.1 Subjects

The Moorfields Eye Hospital visual field database contains 64,949 automated visual field records from 9,482 patients. Records were selected for study on the basis of the following criteria:

1. Untreated normal tension glaucoma patients were chosen in order to analyse the natural history of glaucomatous visual damage in the absence of therapy: current clinical practice dictates that IOP should be lowered in confirmed early POAG, whereas no consensus exists regarding NTG. Following the demonstration that lowering IOP by 25% or more significantly affects the rate of progression,[147] this option is offered to all patients. A cohort of 220 patients who were untreated at the time of the study and whose diagnosis had been confirmed on phasing (IOP $\leq$ 21 mmHg) was identified.
2. Their visual fields appeared to be deteriorating progressively in a typically glaucomatous manner based on the subjective clinical assessment of an experienced observer: serial Humphrey 30-2 printouts were examined and field series demonstrating unequivocal glaucomatous change based on global indices, Total Deviation, Pattern Deviation or grayscales were selected.
3. They had had visual field tests at fairly regularly spaced intervals (Table 2.1). This was necessary in order to simulate less frequent tests by 'thinning' the fields (see 2.3.5).
4. All subjects were experienced in Humphrey 30-2 tests and able to produce reliable computerized visual fields (less than 30% fixation losses and false negatives and less than 15% false positives). Each had had at least 2 tests over 4 months prior to the observation period: this is sufficient to obviate any learning effects[134 154] which may delay the diagnosis of progression.
5. All subjects had visual acuity of 6/12 or better. None had ocular pathology apart from normal tension glaucoma.

	PATIENT 1	PATIENT 2	PATIENT 3	PATIENT 4	PATIENT 5	PATIENT 6
Time (test 1)/yrs	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
Time (test 2)/yrs	0.499	0.057	0.556	**1.285**	0.269	**1.018**
Time (test 3)/yrs	0.690	0.574	0.824	1.440	0.538	1.286
Time (test 4)/yrs	**1.036**	**0.863**	**1.110**	1.553	**0.898**	1.536
Time (test 5)/yrs	1.611	1.247	1.321	1.819	1.244	1.782
Time (test 6)/yrs	1.858	1.515	1.781	**2.164**	1.513	**2.204**
Time (test 7)/yrs	**2.068**	**1.995**	**2.011**	2.452	**1.986**	2.511
Time (test 8)/yrs	2.509	2.394	2.299	**2.912**	2.337	**2.818**
Time (test 9)/yrs	**3.022**	2.739	2.701	3.238	2.647	3.335
Time (test 10)/yrs	3.487	**3.066**	**2.951**	3.545	**2.973**	3.683
Time (test 11)/yrs	3.718	3.871	3.643	3.894	3.260	4.009
Time (test 12)/yrs	**3.970**	**4.258**	**3.816**	**4.047**	**3.644**	**4.373**

Table 2.1 *Times of visual field tests in years from baseline.*

Times included in the 'thinned' fields set are shown in ***bold.***
Mean interval between tests: *4.60 ± 2.20 months (complete fields)*
12.1 ± 2.4 months ('thinned' fields)
Mean period of observation: *4.02 ± 0.27 years*

On the basis of the foregoing criteria, 6 eyes from 6 subjects were selected.

The subjects had an age range of 44 to 67 years (58.0 yrs ± 10.33 yrs) at the start of the observation period.

Four of the subjects were female. Two were male.

2.3.2 Testing strategy

All tests were performed on a standard Humphrey Field Analyzer model 630 automated perimeter. The 30-2 full threshold program with standard 4-2 dB double reversal strategy, as described in 1.2.3.2 (i),was used throughout.

2.3.3 Progression criteria

In order to identify whether a given retinal location was progressively deteriorating, linear regression of retinal sensitivity on time was performed. A negative slope of 1 dB per year or worse associated with $p < 0.1$ for a two tailed t-test of the slope against zero (i.e. the null hypothesis of no deterioration) was used to diagnose progression for a given location. These criteria have been demonstrated to compare closely with the Humphrey Statpac 2 Glaucoma Change Probability analysis.[119]

Pointwise linear regression was preferred to Statpac 2 because this study demanded a reliable estimate of progression for each individual field test. Statpac 2 cannot provide this: it is more effective at diagnosing progression when a location is repeatedly ascribed a high probability of change over a series of consecutive fields.[118]

2.3.4 Detection of progression

Test locations satisfying progression criteria (see 2.3.3) when pointwise linear regression was performed using all the field tests in the 4 year (approx.) observation period for a given eye were defined as 'gold standard' progressing points. For each of these points, the earliest test in which the progression criteria were satisfied was defined as the time of detection of progression. The first test in the observation period was taken as baseline (time = 0).

2.3.5 'Thinning' the fields

Initially, detection times were calculated for all progressing points. Next, in order to simulate a decrease in frequency of testing, the analysis was repeated using only 1 test per year for each eye instead of the complete 3 per year. For example, if tests were performed at times 0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44 and 48 months the 'thinned' fields would only consider the tests done at times 0, 12, 24, 36, and 48 months. Table 2.1 shows the actual times for both complete and 'thinned' fields for each patient.

2.3.6 Statistical analysis

Progressing points which were detected using the 'thinned' fields were identified. Since, by definition, all progressing points were detected using the complete fields, the sensitivity and specificity of the 'thinned' fields could be calculated relative to the gold standard of the complete fields.

For those points which were detected by the 'thinned' fields, detection times were compared with their correlates in the complete fields using a non-parametric test for

paired data from two related samples (Wilcoxon Signed Rank Z test). The statistical software used was SPSS for Windows, version 6.0 (SPSS Inc., Chicago, Ill., USA).

2.4 Results

2.4.1 Ability to detect progression

119 locations were diagnosed as gold standard progressing points by the complete fields. Of these, only 54 were also identified by the 'thinned' fields. Thus the sensitivity of the 'thinned' fields is $54 \div 119 \times 100 = 45.4\%$.

Of the 337 points diagnosed as not progressing using the complete fields, 321 were also identified as not progressing by the 'thinned' fields. The specificity of the 'thinned' fields is thus $321 \div 337 \times 100 = 95.3\%$.

The likelihood ratio(the ratio of the probability that a point is progressing, given that it is identified by the 'thinned' fields as such, to the corresponding probability if it is not progressing) is 9.55.

The 'thinned' fields gave rise to 16 false positives and 65 false negatives (as measured against the gold standard of the complete fields).

These findings are summarised as a contingency table (Table 2.2.)

		State according to complete field series		
State according to 'thinned' field series		Progressing	Not progressing	**TOTAL**
	Progressing	54	16	**70**
	Not progressing	65	321	**386**
	TOTAL	**119**	**337**	**$76 \times 6 = 456$**

Table 2.2 *Contingency table for the performance of the 'thinned' fields*

2.4.2 Detection times

The complete fields gave a mean detection time of 2.52 years with a standard deviation of 1.10 years. The 'thinned' fields gave a mean detection time of 3.46 years with a standard deviation of 0.78 years. These findings are displayed as a

drop-line graph in Figure 2.1. The 'thinned' fields gave consistently later detection times than the full fields: there are only 3 points for which the detection time for the complete fields (represented by a solid circle) is greater than that for the 'thinned' fields (represented by a hollow circle). These points were amongst the latest detection times for the complete fields.

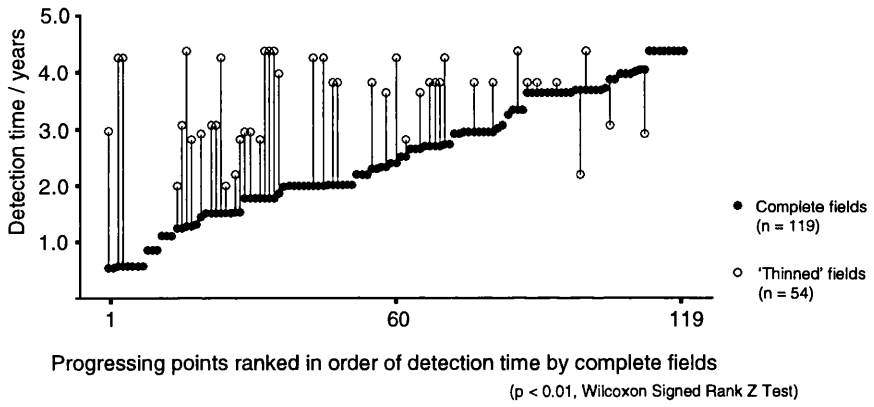


Figure 2.1 *Detection times for complete and 'thinned' visual fields.*

For those progressing points which were detected by the 'thinned' fields, delay was calculated as the difference between the detection time for the complete fields and that from the 'thinned' fields. The mean delay in detection associated with the 'thinned' fields was 1.10 years and the standard deviation was 1.12 years. As can be seen in Figure 2.2, the great majority of values of delay are greater than zero. In other words, the complete fields detect progression sooner than the 'thinned' fields. This is borne out by the results of the Wilcoxon Signed Rank Z test ($p < 0.01$, Table 2.3.)

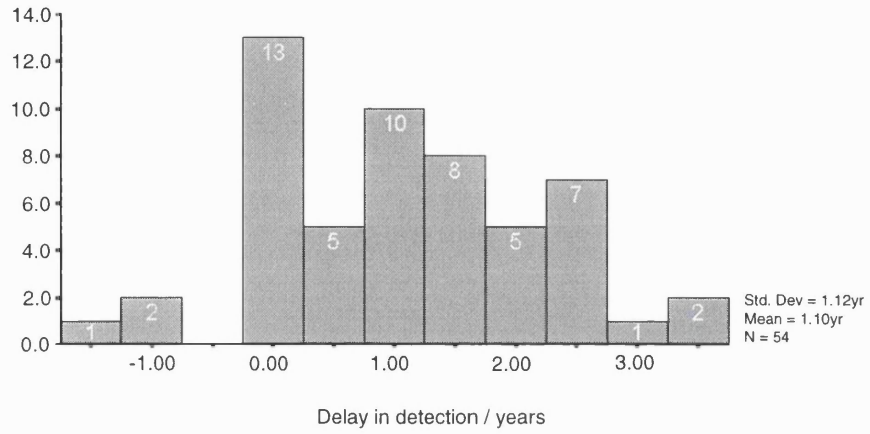


Figure 2.2 *Histogram of delay in detection of progressing points using 'thinned' visual field tests.*

	Cases	Mean Rank
Negative ranks	3	17.00
Positive ranks	41	22.90
Ties	10	
TOTAL	54	

Table 2.3 *Comparison of detection times between complete and 'thinned' fields* (Wilcoxon Signed Rank Z test)

$Z = -5.1816, p < 0.01$

Negative ranks occur when the 'thinned' fields detect progression at a given location earlier than the complete fields.

Positive ranks occur when the complete fields detect progression at a given location earlier than the 'thinned' fields.

Ties occur when both sets of fields detect progression at a given location at the same time.

For each patient, the earliest detection time for any progressing point in the whole visual field is the time at which that patient might first be said to have progressive disease. As Figure 2.3 demonstrates, these times are consistently lower for the complete fields than for the 'thinned' fields ($p < 0.05$). The mean value of the delay associated with the 'thinned' fields for these earliest whole-field detection times is 1.54 years with a standard deviation of 0.54 years.

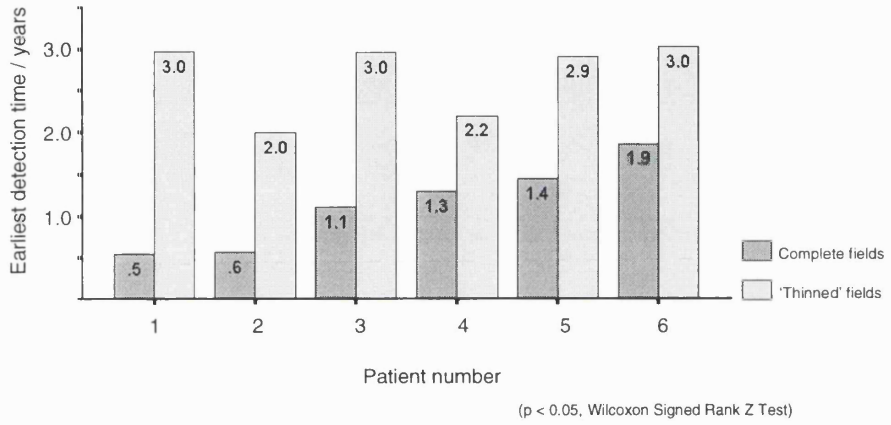


Figure 2.3 *Earliest detection times for each patient.*

2.5 Discussion

One of the main aims of administering regular visual field tests to glaucoma patients is in order to detect progressive disease so that appropriate therapeutic measures may be instituted.

This study demonstrates that an increase in the frequency of visual fields used from 1 field per year to 3 per year provides more rapid, more sensitive diagnosis of progressive glaucoma. It is possible that these effects have been underestimated owing to the technique of 'thinning' the fields. In reality, patients who have undergone 1 field test per year over a given period will be less experienced subjects than those performing 3 tests per year over the same period. For this reason the former group may take longer to overcome learning effects[134 154] and to produce reliable fields. This effect has not been accounted for in the present study since patients are effectively acting as internal controls: the 'thinned' fields are a subset of the complete fields.

In addition to the generalised delay of detection of progression associated with a simulated lower frequency of field testing (Figure 2.2 and Table 2.3), the delay for the earliest whole-field detection times (Figure 2.3) is of particular interest. These times correspond to the times at which progressive disease in at least one retinal location would be detected and decisions concerning the clinical management of the patient would be prompted. In this simulation of less frequent visual fields this delay is marked: 1.54 years over an observation period of 4.02 years.

The sample size in this study was necessarily quite small (119 retinal locations, 6 eyes) since patients were chosen, albeit from a very large database, according to strict criteria. These criteria were chosen to isolate cases worthy of initial study. The high statistical significance of the results, notwithstanding the small sample size, appears to validate the criteria. Further work is required in order to determine whether these results may be generalised to patients with other forms of glaucoma and to those who have received or are receiving medical treatment.

Leaving aside the benefits which earlier diagnosis and treatment of progressive glaucoma may confer upon the individual patient, in pure cost terms the increased use of resources implicit in performing visual fields 3 times per year instead of

once per year must be set against the wider cost of allowing a potentially blinding disease to progress undetected.

CHAPTER 3. PROGRESSOR FOR WINDOWS

PROGRESSOR is a software package which performs pointwise linear regression of sensitivity on time upon visual field series obtained from the Humphrey Field Analyzer. The basic features of PROGRESSOR have been described in 1.3.1.4 (ii).

PROGRESSOR was originally written by Professor F. W. Fitzke at the Institute of Ophthalmology in 1989. This original version of the program was designed to run on the Microsoft Disk Operating System (MSDOS) and has been described in the ophthalmic literature.[119]

In order to increase the speed of analysis, to benefit from the increased flexibility and ease of use which the Windows environment offers and to incorporate new methods of analysis, a new Windows based version of PROGRESSOR was written. It is this latter version which was used in the research in the following chapters of this thesis.

This chapter provides a description of the new features available with the Windows version of PROGRESSOR. The source code for the program is listed in Appendix A. A demonstration version of the program is stored on the CDROM submitted together with this thesis.

3.1 Bilateral display

The MSDOS version of PROGRESSOR analysed each eye of a subject separately and a full screen was necessary to display the results from each eye. The present version of the program displays the results from both eyes within the same window separated by a splitter bar (Figure 3.1). This bar may be moved to the edge of the window if a view of the results from only one eye is desired.

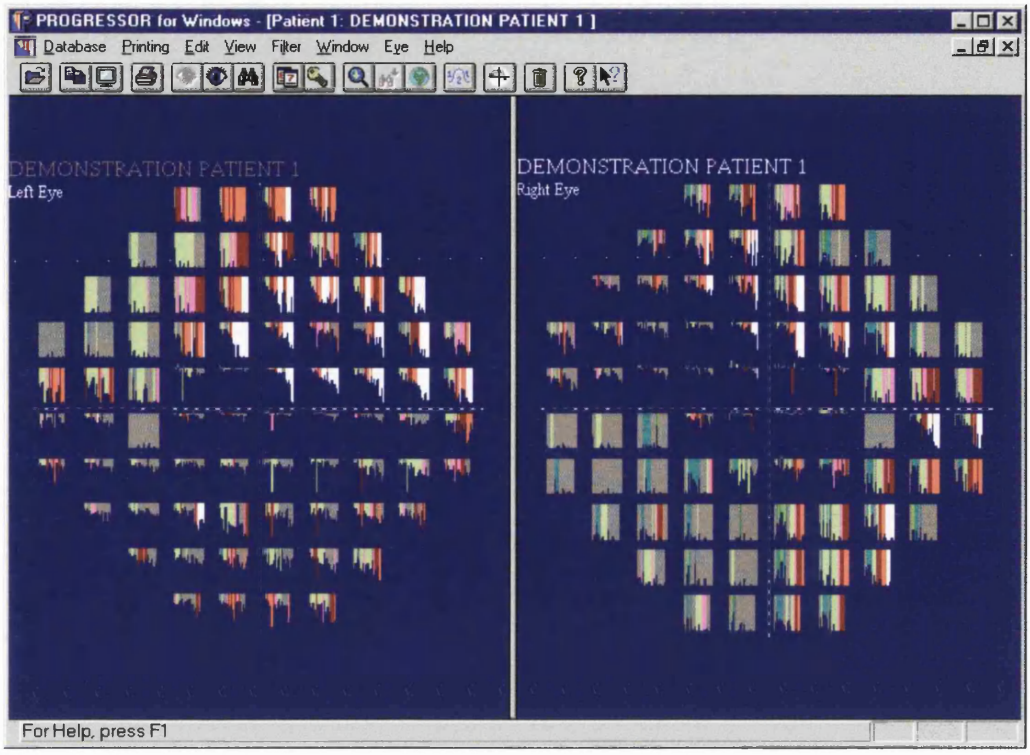


Figure 3.1 *Simultaneous display of the results of analysis from each eye of a subject.*

3.2 Status bar information

As the mouse cursor is moved over the cumulative graphical display, the information shown in the status bar changes from the default "for Help, press F1" to show information about the test location to which the cursor is moved. Sensitivity in decibels, rate of change of sensitivity (slope of the regression line) and the p-value of the regression line are shown. These data relate to the most recent test for the location under consideration unless a subset of dates are being analysed (see 3.4).

3.3 Magnification

A magnified, detailed image of the cumulative bar graph at any test location may be viewed using the 'click and drag' technique familiar to Windows users. An example of the results of this process is shown in Figure 3.2.

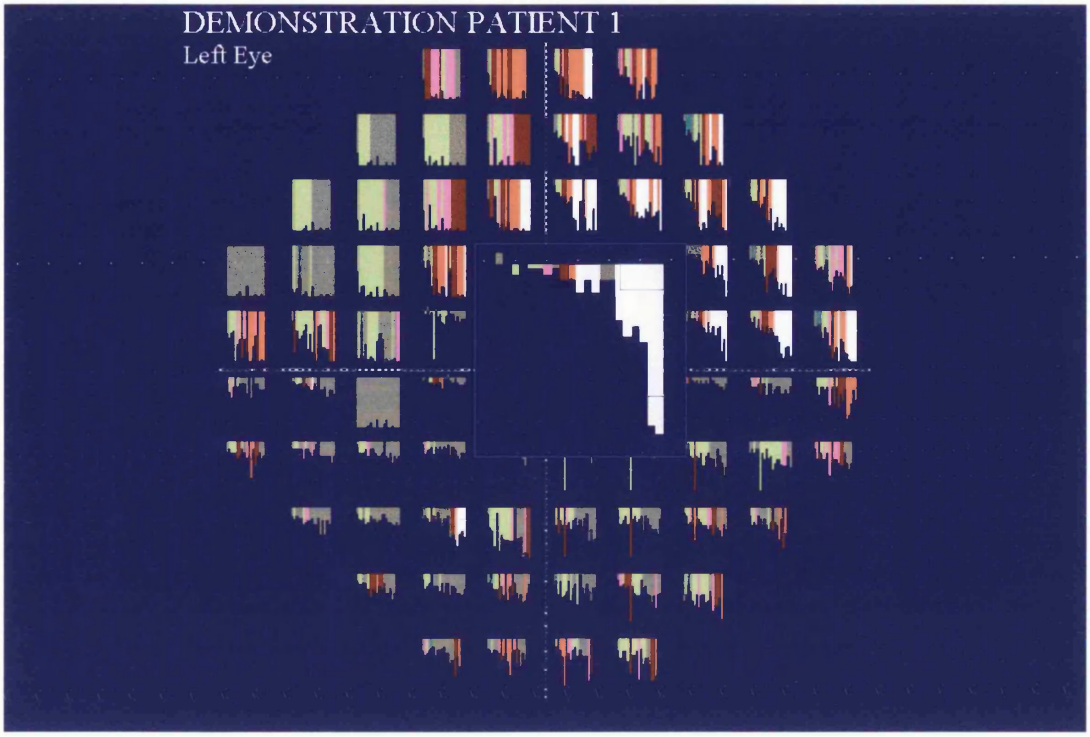


Figure 3.2 *Cumulative graphical display after magnification of a central test location.*

3.4 Progressing points

As already shown in Figure 1.8a, test locations which satisfy progression criteria may be highlighted in red. It is possible to specify progression criteria within a wide range of values for both regression slope and p-value for both inner (non-edge) and edge points. The dialog box used to control progression criteria is shown in Figure 3.3.

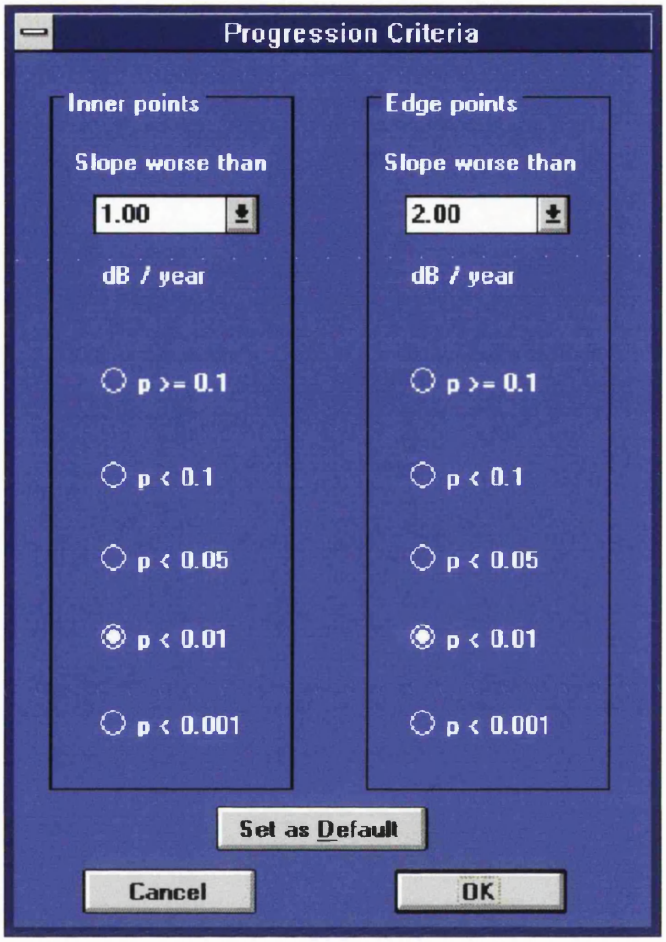


Figure 3.3 *Progression criteria dialog box.*

3.5 Subset analysis

By default, all the visual field test results from both eyes of a subject are analysed by PROGRESSOR. However, any subset of tests may be readily analysed by means of the dialog box shown in Figure 3.4. This may be useful for re-analysing results when other clinical parameters change, for example before and after surgery. It may also be used to exclude test results which are regarded as outliers.

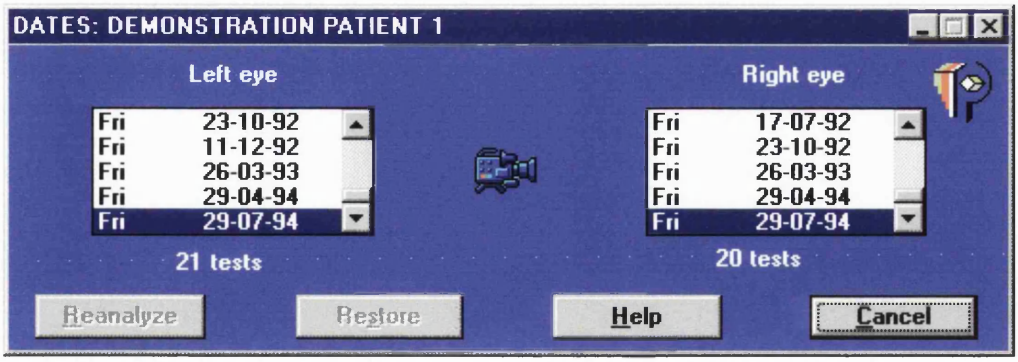


Figure 3.4 *Date selection dialog box.*

The dates of tests from each eye required for inclusion in the sub-analysis are selected and then the Reanalyze button is clicked. To return the program to a full analysis of all field tests from both eyes, the Restore button is clicked.

3.6 Progression indices

For any selected tests, summary measures of progression status may be displayed as shown in Figure 3.5. These consist of the number of progressing points which satisfy progression criteria, the mean value of the regression slopes for all test locations and the mean value of the regression slopes for progressing points only.

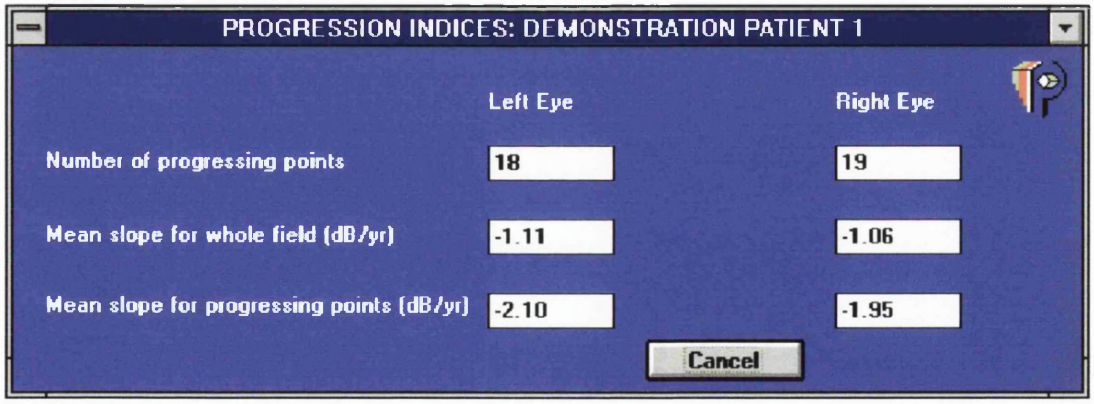


Figure 3.5 *Progression indices dialog box.*

The mean slope value has been used to investigate the effect of surgery on visual field progression in normal tension glaucoma, as mentioned in 1.3.3.[147]

3.7 Gaussian filtering

This technique has been mentioned in 1.3.2.1 and has been investigated further in Chapter 6. PROGRESSOR is able rapidly to apply the standard 3 x 3 Gaussian filter to series of visual field tests. Figure 3.6 shows the results of Gaussian filtering on an example cumulative graphical display.

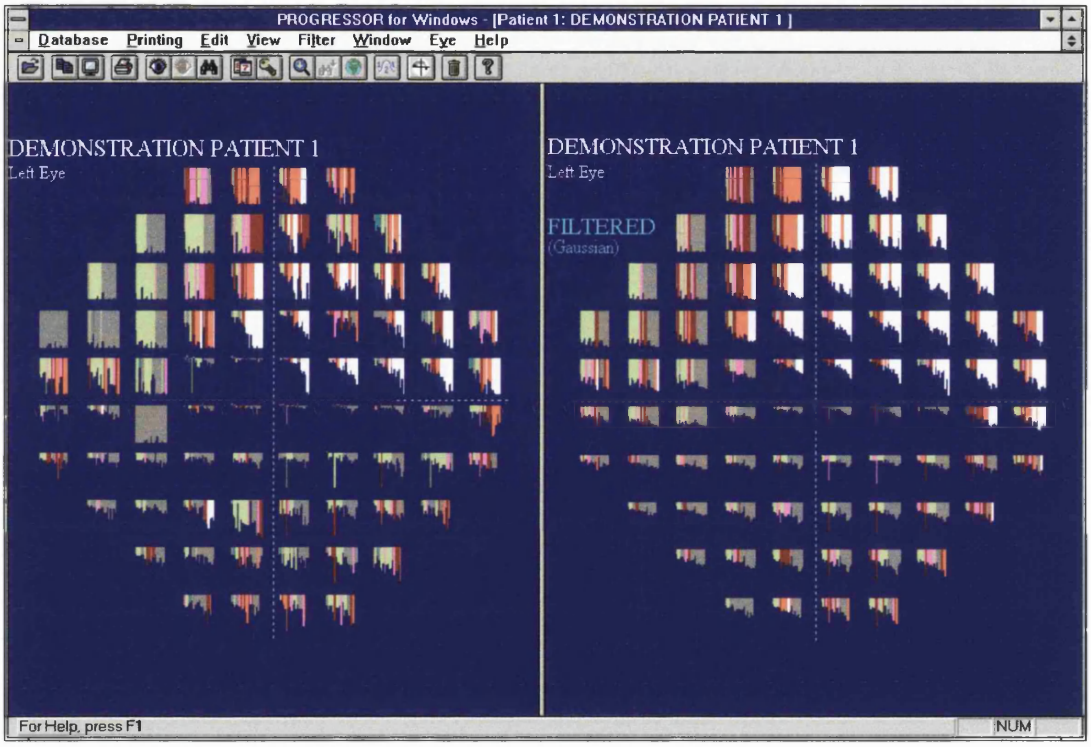


Figure 3.6 *An example of Gaussian filtering. The left pane shows the unfiltered PROGRESSOR analysis. The right pane shows the same visual field series after the filter has been applied. The progressing locations in the superonasal area of the field appear to decay more regularly and to reach high statistical significance (as shown by white bars, $p < 0.001$) sooner after filtering has been performed.*

3.8 Binocular simulation

The results of bilateral monocular visual field tests may be merged to give a representation of the central binocular visual field. In order to achieve this, the sensitivity value of any given test location in the left eye is compared with the sensitivity value of the corresponding test location in the right eye and the higher of the two sensitivity values is taken as the value to be displayed in the binocular

simulation. The result is displayed in a gray scale format similar to the gray scale produced by the Humphrey Field Analyzer (Figure 3.7).

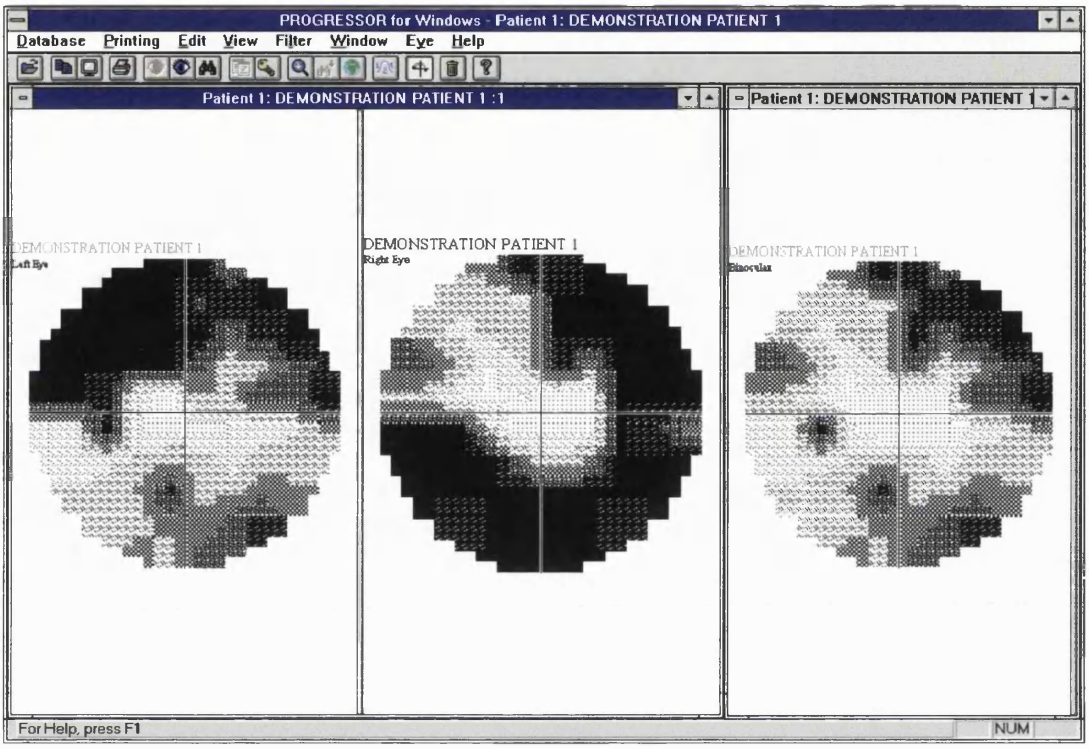


Figure 3.7 *Binocular simulation display. The left and middle panes show grayscales for left and right eyes respectively. The right pane shows the results of binocular simulation.*

In addition, test locations in the binocular simulation with a sensitivity of less than 10 decibels may be highlighted, along with a ring which corresponds to the boundary of the central 20 degrees of the visual field (Figure 3.8). Locations which have a binocular sensitivity of less than 10 decibels and lie within the central 20 degrees of the visual field have an important bearing on the subject's legal fitness to drive in the United Kingdom.(155)

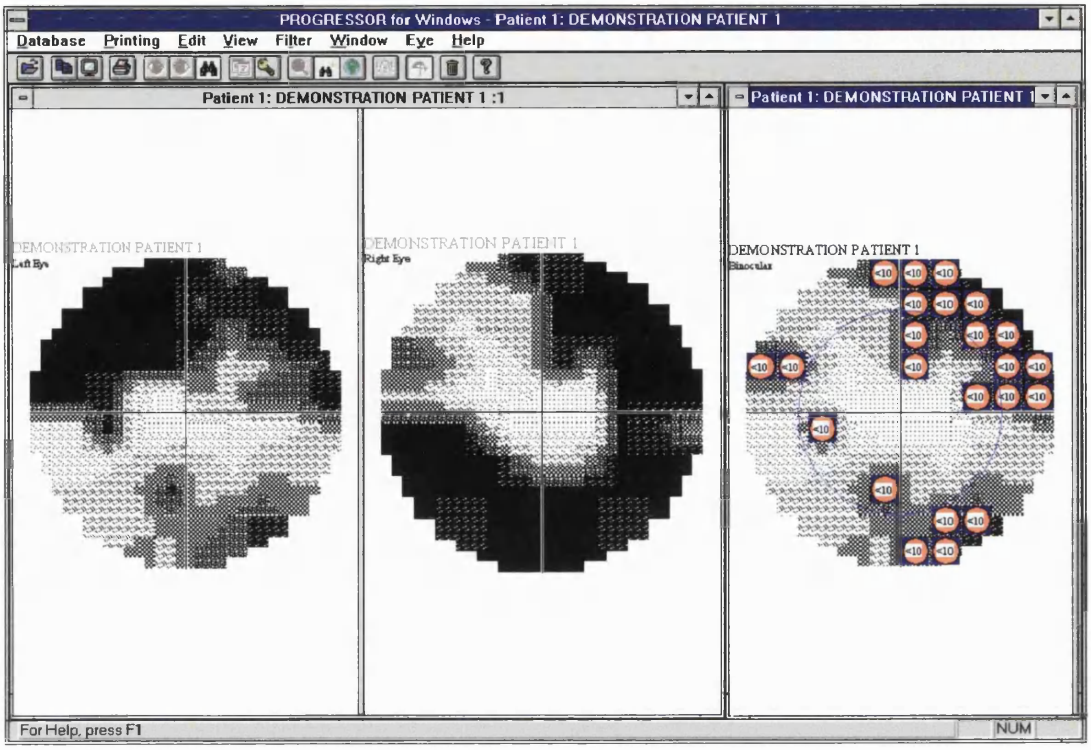


Figure 3.8 *Binocular simulation display identical to that shown in Figure 3.6 except that points in the binocular simulation with a sensitivity of less than 10 dB and a blue ring corresponding to the central 20 degrees of the visual field are shown.*

The binocular simulation analysis performed by PROGRESSOR has been formally validated and has been shown to be in good agreement with the results of true binocular testing.(156)

3.9 Help system

A comprehensive context-sensitive Help system has been built into the program. Users who are familiar with other Windows programs are able to use the software without a lengthy training period. The Help system is included with the demonstration version of the program on the CDROM which is submitted together with this thesis.

CHAPTER 4. VALIDATION OF PROGRESSOR

4.1 PROGRESSOR compared with Statpac 2

4.1.1 Abstract

Purpose: To compare the performance of PROGRESSOR (pointwise linear regression) and Statpac 2 (comparison with baseline values) in detecting early deterioration in the visual fields of glaucoma patients.

Methods: Visual field series from 19 untreated normal tension glaucoma eyes which were deteriorating on clinical grounds were analysed by PROGRESSOR and Statpac 2. Progression criteria for PROGRESSOR were 1) inner points: slope < -1 dB / year, $p < 0.05$ and 2) edge points: slope < -2 dB / year, $p < 0.05$. Criteria for Statpac 2 were $p < 0.05$ change probability for any point on 3 consecutive fields. Detection time was defined as the time interval between the initial field and the first field in which at least one progressing point was identified. Detection times produced by the two techniques were compared.

Results: PROGRESSOR and Statpac 2 agreed on progression in all 19 eyes. Mean detection time for PROGRESSOR was 1.077 years (SD 0.985 years) and for Statpac 2 was 2.161 years (SD 1.357 years). PROGRESSOR detected progression sooner than Statpac 2 in 18 eyes ($p < 0.01$, Wilcoxon Matched-Pairs Signed-Ranks Test). PROGRESSOR detected progression earlier by a mean of 1.085 years (SD 0.936 years).

Conclusions: PROGRESSOR consistently detected progression earlier than Statpac 2. The PROGRESSOR software is a useful tool for the early detection of visual field deterioration in glaucoma.

(Viswanathan AC, Fitzke FW, Hitchings RA. Early Detection of Visual Field Progression in Glaucoma: A Comparison of PROGRESSOR and Statpac 2. *Br J Ophthalmol* 1997;81(12):1037-1042.)

4.1.2 Introduction

PROGRESSOR and Statpac 2 are methods of pointwise analysis of glaucomatous visual field series: they are described in 1.3.1.4.

PROGRESSOR has been found to emulate Statpac 2 closely in diagnosing individual locations in a single field of a series as either stable or progressing when the appropriate progression criteria are used.[124] However, the relative ability of the two algorithms to detect the first sign of deterioration has not been investigated directly. This is an important aspect of the techniques' behaviour to compare, since the timely and reliable detection of glaucomatous visual field progression is of paramount clinical importance: in the presence of adequate intraocular pressure control, questions concerning the commencement or alteration of therapy are prompted only when definite visual deterioration is seen.

4.1.3 Methods

4.1.3.1 Subjects

The Moorfields Eye Hospital visual field database currently contains 64,949 automated visual field records from 9,482 patients. Records were selected for study on the basis of the following criteria:

1. Untreated normal tension glaucoma patients were chosen in order to analyse the natural history of glaucomatous visual damage in the absence of the potentially confounding effects of medical therapy. A cohort of 220 such patients, whose diagnosis had been confirmed on phasing, was identified.
2. Their visual fields showed progressive deterioration of a typically glaucomatous nature. This was done by inspection of the Statpac overview analysis of serial Humphrey visual fields by an experienced observer.
3. All subjects were experienced in Humphrey 30-2 tests and able to produce reliable computerized visual fields (less than 30% fixation losses and false negatives and less than 15% false positives). Each had had at least 2 tests over 4 months prior to the observation period: this is sufficient to obviate any learning effects[134 154] which may delay the diagnosis of progression.
4. All subjects had visual acuity of 6/12 or better. None had ocular pathology apart from normal tension glaucoma.

On the basis of the foregoing criteria, 19 eyes from 13 subjects were selected.
An indication of the degree of glaucomatous damage in the selected group is given by the following summary measures of the Mean Deviation (MD) of the initial field in each test series: the mean of the MDs was -6.81 dB (SD 6.01 dB), the median was -5.43 dB and the range was -22.40 dB to +1.07 dB.

4.1.3.2 Testing strategy

All tests were performed on a standard Humphrey automated perimeter. The full threshold 30-2 program with standard 4-2 dB staircase strategy was used throughout. Tests were performed at intervals of approximately four months.

4.1.3.3 Progression criteria

4.1.3.3(i) Glaucoma Change Probability: Statpac 2

A field series was regarded as progressing if the Glaucoma Change Probability analysis of Statpac 2 marked any test location as showing significant deterioration ($p < 0.05$) from the baseline relative to the Humphrey normal database on three consecutive occasions. The requirement for a location to be repeatedly ascribed a high probability of change over a series of consecutive fields has been recommended by the originators of Statpac 2[118] and has been used in previous studies.[119 124] The specific criterion of three consecutive fields is currently in use in the Early Manifest Glaucoma Trial in Malmö, and the importance of confirmatory testing to establish progression relative to a baseline has been upheld in the ongoing Collaborative Normal Tension Glaucoma study.[115]

4.1.3.3(ii) PROGRESSOR

A field series was regarded as progressing if PROGRESSOR identified at least one test location with a negative slope of 1 dB per year or worse associated with $p < 0.05$ for a two tailed t-test of the slope against zero (i.e. the null hypothesis of no deterioration). The slope criterion of 1 dB per year represents a rate of sensitivity loss approximately ten times greater than the normal age-related decline.[97] Edge points are known to be more subject to fluctuation[97] so a stricter slope criterion of 2 dB per year (also with $p < 0.05$) was introduced for them. These slope criteria, in combination with a less stringent slope significance criterion of $p < 0.1$, have been demonstrated to compare closely with the Humphrey Statpac 2 Glaucoma Change Probability analysis.[119 124]

4.1.3.4 Detection time

The detection time for a given field series for a given algorithm was defined as the time interval between the initial field in the series and the field when the progression criteria for that algorithm (see 4.1.3.3) were first satisfied.

4.1.3.5 Reliability of early detection

There is at present no 'gold standard' for the identification of visual field progression in glaucoma. Thus it is very difficult to assess whether a given technique is detecting 'true' progression. Some authors have used clinical impression against which to compare the performance of various algorithms,[111] but this has been shown to be a largely subjective measure.[104] Others have avoided the problem entirely by not attempting to estimate the reliability of their techniques.[106]

In the field series selected for this study, all three methods of assessing change (clinical expertise, PROGRESSOR and Statpac 2) agreed that all of the series showed progression. However, this study was designed to examine early detection of progression by the algorithms, so it was important to assess whether progression detected at a given field in a series was sustained in the rest of the series: it is likely that, if progression is diagnosed in one field but not in subsequent fields, the diagnosis is incorrect.

The reliability of early detection of progression by PROGRESSOR and Statpac 2 was assessed by examining whether at least one of the locations initially detected as progressing was still progressing in the final field of the series. For Statpac 2, the criterion of repeatability over three fields was not applied for the final field of the series in this analysis: Statpac 2 was judged as reliable if at least one location initially diagnosed as progressing was labelled with a black triangle in the single final field.

4.1.3.6 Statistical analysis

For each field series, detection time (see 4.1.3.4) was calculated for each algorithm. Detection times for PROGRESSOR were compared with their correlates for Statpac 2 using a non-parametric test for paired data from two related samples (Wilcoxon Signed Rank Z test).[157]

Statistical analysis was performed using the software package SPSS for Windows, version 6.0 (SPSS Inc., Chicago, Ill., USA).

4.1.4 Results

4.1.4.1 Detection times

All 19 field series satisfied the progression criteria for both PROGRESSOR and Statpac 2. The mean detection time for PROGRESSOR was 1.077 years with a standard deviation of 0.985 years. Statpac 2 gave a mean detection time of 2.161 years with a standard deviation of 1.357 years. These findings are displayed in Table 4.1.1, and the individual differences between the algorithms for each field series are shown as a drop-line graph in Figure 4.1.1. Statpac 2 has consistently later detection times than PROGRESSOR: there is only one field series in which the detection time for PROGRESSOR is greater than that for Statpac 2.

	Detection time (years)	
	Mean	**SD**
PROGRESSOR	1.077	0.985
Statpac 2	2.161	1.357

Table 4.1.1 *Detection times for PROGRESSOR and Statpac 2*

($p < 0.01$, Wilcoxon Signed Rank Z test)

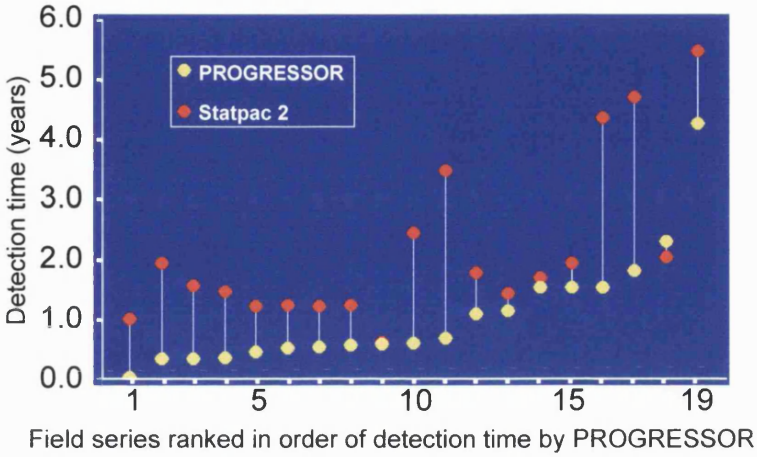


Figure 4.1.1 *Drop-line graph of detection times for each field series for PROGRESSOR and Statpac 2. The field series are ranked in order of detection time by PROGRESSOR.*

4.1.4.2 Delay

Delay in detection was calculated as the difference between the detection time for Statpac 2 and that for PROGRESSOR. The mean delay in detection associated with Statpac 2 was 1.085 years and the standard deviation was 0.936 years. As can be seen in Figure 4.1.2, the great majority of values of delay are greater than zero. In other words, PROGRESSOR detects progression sooner than Statpac 2. This is borne out by the results of the Wilcoxon Signed Rank Z test ($p < 0.01$).

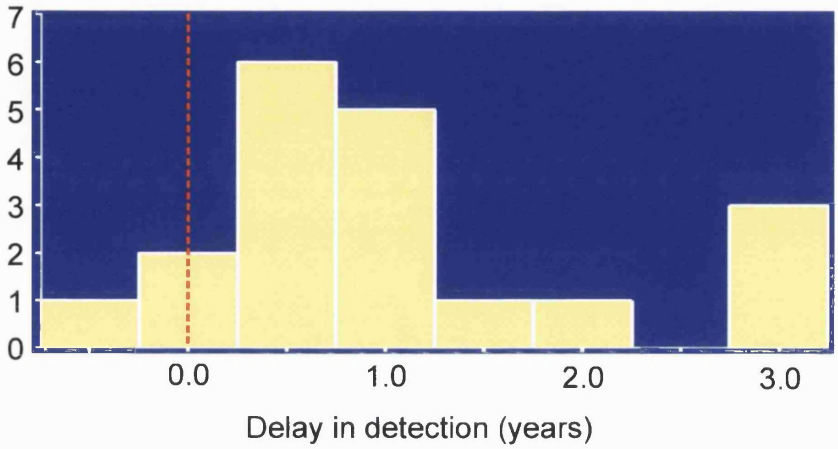


Figure 4.1.2 *Histogram of delay in detection associated with Statpac 2. The mean delay is 1.085 years (SD 0.936 years).*

4.1.4.3 Reliability

For PROGRESSOR, at least one of the locations initially detected as progressing was still progressing in the final field of the series in 78.9% (15 out of 19) of the field series. For Statpac 2, the corresponding figure was 68.4% (13 out of 19).

4.1.5 Discussion

Previous work has shown that PROGRESSOR agrees closely with Statpac 2 in terms of which test locations are classified as progressing and which are classified as stable.[119 124] Thus it seems paradoxical that PROGRESSOR should be found to be superior to Statpac 2 in the detection of progression. This apparent contradiction may be explained by considering the differences in methodology between this and the previous studies.

In the absence of a gold standard for glaucomatous visual field progression, the previous authors chose Statpac 2 as an arbitrary gold standard against which to measure PROGRESSOR. Statpac 2 was used to identify test locations which had 'unequivocally deteriorated' in the last three fields of series consisting of sixteen fields each.[124] The ability of PROGRESSOR to discriminate between these locations and the other locations (defined as stable by Statpac 2) was then examined using the kappa statistic[158] to assess the level of agreement with Statpac 2. A kappa coefficient of $\kappa = 0.62$ (SE 0.04) was obtained, which represents good agreement for this length of follow up.[159] However, as the authors note,[119 124] this analysis is inherently unable to detect any potential superiority of PROGRESSOR in detecting progression: it would be interpreted as a lack of specificity.

There are also theoretical explanations for the finding that PROGRESSOR detects progression sooner than Statpac 2. The two algorithms attempt to diagnose progression in very different ways. Statpac 2 compares each field under analysis with the baseline: information in fields after the baseline but before the field under analysis is ignored. Statpac 2 is thus an *event* type of analysis and would be particularly sensitive to catastrophic, stepwise change. In contrast, PROGRESSOR uses all the fields up to and including the field under analysis: it is thus a *trend* type of analysis and would be more sensitive to gradual, sustained change. The pattern of pointwise change in progressive glaucoma has been investigated: although sudden, stepwise change does occur[160] the mode of progression of the visual field of untreated normal tension glaucoma patients is best described by a pointwise linear analysis.[120] Furthermore, Statpac 2 is only able to classify

locations if their level of loss can be determined relative to a normal database: previous work comparing PROGRESSOR with Statpac 2 has excluded 26% of test locations from analysis because they cannot be classified by Statpac 2.[124]

In an early report, the Normal Tension Glaucoma Study Group described an event type of analysis which can be used to diagnose visual field progression in a timely, sensitive and specific way.[115] In this analysis, patients are followed every three months with visual field tests. If the criteria for suspected progression are met, the patient returns within one to four weeks for one or two confirmatory tests. If progression is confirmed in this way on two consecutive series of visits three months apart, true progression is diagnosed. Thus the minimum time to diagnosis is three months and one week, which is earlier than for either algorithm in this study. The application of this testing protocol would enable earlier diagnosis of progression with Statpac 2. However, PROGRESSOR would also be likely to detect change earlier in this case, since more frequent testing results in the earlier achievement of a significant slope: formal investigation would be required to determine whether PROGRESSOR or Statpac 2 would benefit more from this strategy of confirmatory testing. Furthermore, this study protocol entails a high frequency of testing which would have profound resource implications if it were implemented on a routine basis, since it may involve three tests every three months.

In summary, this study was not designed to assess the sensitivity or specificity of PROGRESSOR or Statpac 2 relative to an arbitrary gold standard (since this has been examined elsewhere[111 124]), but rather to investigate the performance of the two algorithms in reliably detecting early change in a group of patients known to be deteriorating unequivocally on clinical grounds. The fact that PROGRESSOR consistently detected progression earlier than Statpac 2, and detected it more reliably, suggests that PROGRESSOR is a useful software tool for the analysis of visual field progression in glaucoma.

4.2 PROGRESSOR compared with standard clinical methods

4.2.1 Abstract

Purpose: To examine the agreement between clinicians in assessing visual field progression using two different methods of analysis.

Methods: Twenty-seven Humphrey visual field series from 27 patients were selected from a database of 64,949 field tests. Each series satisfied the following criteria: more than 19 fields (all with adequate Humphrey reliability indices), patient age over 40 years, macular threshold at least 30 dB. The first 3 fields in each series were excluded to minimise learning effects: the following 16 were studied. Five clinicians assessed the progression status of each field series using both standard Humphrey printouts and using pointwise linear regression (PROGRESSOR). The level of agreement between each possible pair of clinicians was evaluated using a weighted kappa statistic.

Results: The agreement on progression status between all pairs of clinicians using PROGRESSOR (median kappa = 0.63) was always superior to using Humphrey printouts (median kappa = 0.28).

Conclusions: Agreement between expert clinicians about visual field progression status is poor when standard Humphrey printouts are used, even when the field series studied are long and consist solely of reliable fields.

Clinicians agree more closely about patients' visual field progression status when using PROGRESSOR than when inspecting series of Humphrey printouts.

(Viswanathan AC, Crabb DP, McNaught AI, Westcott MC, Kamal D, Garway-Heath DF, Fitzke FW, Hitchings RA. Inter-Observer Agreement on Visual Field Progression in Glaucoma: a Comparison of Methods. *Invest Ophthalmol Vis Sci* 1999, under review.)

4.2.2 Introduction

"At present, there is no generally accepted technique for detecting change in the visual field over time using automated perimetry...most clinicians who use automated perimetry are probably employing simple visual inspection of the perimetric data to diagnose progressive visual field loss in patients with glaucoma."[104] Although these statements were made a decade ago they are still true today, even though the knowledge of whether a patient's glaucoma is progressive or stable remains central to the management of the condition. Automated perimetry, as opposed to manual perimetry, provides an objective reproducible strategy for testing the visual field. However, even expert observers show considerable disagreement about whether a given visual field series signifies progression or stability.[104] One possible reason for this is that the standard output of most automated perimeters provides inadequate information relating to progression or stability. Thus, when the clinician is attempting to decide about whether or not a given series of outputs constitutes progressive disease, the task involves manually comparing the decibel sensitivity values (or processed versions thereof) or graphical plots for all fields in the series. This task is further complicated by the contributions of within-test variability (short-term fluctuation)[101] and between-test variability (long-term fluctuation),[153] both of which are known to be increased above normal in glaucoma.[128] For these reasons, and because the grids of numbers produced by automated perimeters are easily amenable to numerical analysis, a great variety of software and statistical approaches have been taken to determine visual field progression in glaucoma.

The PROGRESSOR software for the analysis of serial visual fields to detect glaucomatous change has been described in 1.3.1.4(ii). Since there is no general consensus on what pointwise value of regression slope and p-value constitutes progression, or whether there should be a requirement for contiguous points to show this behaviour and whether it should be maintained in subsequent fields, at present PROGRESSOR remains a subjective analysis. For this reason, and because there is no universally accepted gold standard for visual field progression, this study investigates the usefulness of PROGRESSOR by determining the level

of agreement between expert observers using both PROGRESSOR and standard clinical techniques (manually comparing serial printouts from an automated perimeter).

4.2.3 Methods

4.2.3.1 Selection of visual field data

The visual field series presented to the five observers in the study for evaluation were drawn from the clinical visual field database of the Glaucoma Service at Moorfields Eye Hospital, a tertiary referral centre also serving the local community. At the time of the study this database contained 64,949 automated visual field records from 9,482 patients. All data was the result of standard clinical testing on Humphrey model 630 perimeters. Field series were selected for study on the basis of the following criteria:

- Patient age greater than 40 years.
- Each field series consisted solely of tests using the Humphrey 24-2 and 30-2 test grid patterns with standard 4-2 dB staircase thresholding strategy.
- A white stimulus of Goldmann size III was used throughout.
- Each field test was required to meet reliability criteria as set by the Humphrey perimeter: fewer than 20% fixation losses, fewer than 33% false positive and fewer than 33% false negative responses.
- The macular threshold of each test was required to be at least 30 decibels. This criterion was introduced so as to rule out patients with significant media opacities or macular disease.
- Each series included in the study contained at least 19 visual fields satisfying the foregoing criteria. The first three fields in each series were ignored to obviate learning effects[134]: the following 16 fields were presented to the observers for analysis.
- If both eyes of a particular patient fulfilled the inclusion criteria, one eye was selected at random.

After the above criteria had been applied to the database, 27 visual field series from 27 patients were included in the study.

4.2.3.2 Analysis of visual field data

Each of the visual field series was assessed by all five observers. The observers were experienced in the assessment of both Humphrey printouts and PROGRESSOR datasets. Initially, the observers were asked to examine the visual field series presented as standard Humphrey printouts. Although each patient's visual fields were, of course, presented in chronological order within a visual field series, the visual field series themselves were presented in a random order determined by a software random number generator. Observers were asked to use their clinical expertise to assess each visual field series (which was presented as raw sensitivity values, the grey scale plot, total deviation values and plot, pattern deviation values and plot, global indices and Glaucoma Hemifield Test in the standard format) and to assign it to one of four categories: definitely stable, probably stable, probably progressing, or definitely progressing. Then, the observers were asked to rate the field series using PROGRESSOR. For this analysis the visual field series were presented in a different random order from that used for the Humphrey printouts. (Naturally, each patient's visual fields remained in chronological order within a given visual field series).

The observers were allowed to use any of the options within PROGRESSOR which they considered were necessary for an accurate analysis. These included the cumulative graphical output, a choice of progression criteria for slope and p-value, grey scale plots, animation analysis and Gaussian filtering. The latter has been shown to reduce long-term fluctuation[129] without delay in the detection of visual field deterioration.[161] Gaussian filtering is described in detail in 6.3.5. The observers were asked to categorise the visual field series using PROGRESSOR into the same four categories previously described for use with the Humphrey printouts.

In order to measure the level of intra-observer agreement two of the observers were asked to re-examine all the visual field series using both Humphrey printouts and PROGRESSOR three months after their original analysis. These observers were given no prior warning that intra-observer variation was to be measured as part of the study.

The level of inter-observer agreement was measured using the weighted kappa statistic[158] for all pairs of observers. Weighted kappa is an appropriate chance-adjusted measure of agreement when there are more than 2 ordered categories of classification. The statistic, which ranges from 0 (agreement no better than chance) to 1 (perfect agreement), gives partial credit for partial agreement in accordance to a linear weighting scheme. Additionally, the level of the observer concordance was assessed: the proportion of the observers agreeing about the category of a particular field series was measured. Similar statistics were used to measure the level of intra-observer agreement.

4.2.3.3 Statistical analysis

All statistical analysis was performed using the software package S-PLUS 3.2 for Windows™ (StatSci Europe, MathSoft Inc., Oxford, UK).

4.2.4 Results

The median age of the patients used in this study at the first visual field in the series analysed was 61 years (range 44 to 72 years). The median Humphrey Mean Deviation (MD) of the first visual field in the series analyzed was -7.7 dB (range -0.1 to -14.8 dB). The median length of follow up for the visual field series was 5.7 years (range 3.3 to 7.7 years).

Tables of agreement were produced for all pairs of the observers. For example, Table 4.2.1 shows the classification of the 27 field series by one pair of the observers (A and B) when using the Humphrey printouts.

		Observer B		
Observer A	Definitely Stable	Probably Stable	Probably Progressing	Definitely Progressing
Definitely Stable	**5**	4	1	0
Probably Stable	4	**1**	2	1
Probably Progressing	0	3	**1**	3
Definitely Progressing	0	1	0	**1**

Table 4.2.1 *Classification of field series using Humphrey Printouts.*

Classification of the 27 visual field series by Observer A and Observer B when using Humphrey printouts. These two observers agreed exactly on the assessment of 8 series (along the diagonal of the table). Note that one field series was assessed as being 'probably stable' by observer A but 'definitely progressing' by observer B. This level of disparity in assessment occurred with three series in total. These 'serious disagreements', which are accounted for in the calculation of the weighted kappa value, were a feature of the results from other pairs of observers when Humphrey printouts were the method of assessment.

The weighted kappa value for the agreement exhibited in Table 4.2.1 is 0.28 (SE 0.08). Table 4.2.2 shows the results for the same pair of observers when using PROGRESSOR analysis.

		Observer B		
Observer A	Definitely Stable	Probably Stable	Probably Progressing	Definitely Progressing
Definitely Stable	**7**	0	0	0
Probably Stable	0	**0**	1	0
Probably Progressing	0	4	**2**	4
Definitely Progressing	0	1	2	**6**

Table 4.2.2 *Classification of field series using PROGRESSOR.*

Classification of the 27 visual field series by Observer A and Observer B when using PROGRESSOR. These observers now agreed exactly on the assessment of 15 series (along the diagonal of the table). 'Serious disagreements' have been reduced and the table suggests closer conformity between the observers when they assessed the series with PROGRESSOR.

The weighted kappa value for the agreement between the two observers has increased substantially to 0.63 (SE 0.13).

The weighted kappa values for all the pairs of observers using the different methods of analysis are shown graphically in Figure 4.2.1.

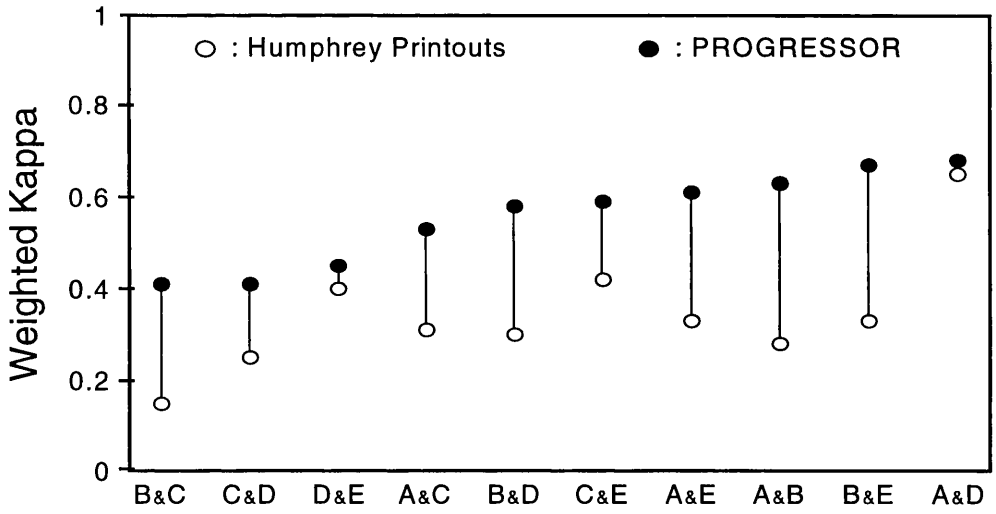


Figure 4.2.1 *Weighted kappa values of agreement for all ten pairs of observers. Each pair is represented by an open symbol representing the weighted kappa value when using Humphrey printouts and a closed symbol when using PROGRESSOR. The pairs are ranked along the x-axis by magnitude of the weighted kappa value. In all cases the weighted kappa value of agreement was higher when observers used PROGRESSOR compared to when using Humphrey printouts. The range of standard error for all the weighted kappa values was 0.08 to 0.14.*

All pairs of observers demonstrated greater weighted kappa values for agreement when PROGRESSOR was used compared to Humphrey printouts. The median weighted kappa value for all pairs of observers using PROGRESSOR was 0.59 compared to a median of 0.32 when using Humphrey printouts. Qualitative interpretation of kappa levels of agreement are imprecise but guidelines have been published.[159] Generally a value less than 0.40 indicates only 'slight' agreement and a value above 0.60 indicates 'substantial' agreement.

Tables 4.2.3a and 4.2.3b show the concordance between the observers in classifying the visual field series using the different methods.

Concordance among observers	Series classified as progressing	Series classified as stable	**Total**
100 %	5	5	**10 (37%)**
80 %	2	6	**8 (30%)**
60 %	6	3	**9 (33%)**
Total	**13**	**14**	**27 (100%)**

Table 4.2.3a *Concordance between observers when using Humphrey printouts.*

Concordance among observers	Series classified as progressing	Series classified as stable	**Total**
100 %	13	7	**20 (74%)**
80 %	3	0	**3 (11%)**
60 %	3	1	**4 (15%)**
Total	**19**	**8**	**27 (100%)**

Table 4.2.3b *Concordance between observers when using PROGRESSOR.*

Concordance between observers when using Humphrey printouts (a) and PROGRESSOR (b). Categories were collapsed into progressing (probably and definitely) and stable (probably and definitely). One hundred percent concordance represents exact agreement among all 5 observers, 80% concordance represents exact agreement among four observers, and 60% concordance represents exact agreement among 3 observers.

One hundred percent total concordance was defined as all 5 observers either agreeing a series was progressing (either probably or definitely) or all 5 observers agreeing a series was stable (either probably or definitely). This rate of

concordance was achieved in only 10 of the 27 series (37%) when the observers used Humphrey printouts. In comparison the 100% rate of concordance was doubled when the observers used PROGRESSOR: all 5 observers had identical opinions on 20 of the 27 series (74%).

Tables 4.2.3a and 4.2.3b also show that 13 of the 27 series (48%) were classified as progressing by the majority (at least 3 or 60%) of the observers using Humphrey printouts. In contrast 19 of the 27 series (70%) were classified as progressing by the majority of the observers when using PROGRESSOR.

Intra-observer agreement was assessed in two of the observers as described in the Methods section. The weighted kappa statistic for intra-observer agreement for observer A was 0.43 (SE: 0.11) using Humphrey printouts and 0.71 (SE: 0.13) when using PROGRESSOR. The weighted kappa statistic for intra-observer agreement for observer B was 0.60 (SE: 0.12) using Humphrey printouts and 0.83 (SE: 0.09) when using PROGRESSOR.

4.2.5 Discussion

It is not particularly surprising that only slight inter-observer agreement was found when standard Humphrey printouts were used as a basis for a decision about glaucomatous visual field progression (median weighted kappa value 0.32). In a previous study, Werner and colleagues measured the level of agreement between six experienced clinicians in rating the progression status of automated visual field series from 30 glaucoma patients.[104] Although the weighted kappa value was not reported in that paper, it has been calculated subsequently as 0.402.[105] This is comparable to our findings. It is interesting that Werner and colleagues found a slightly higher level of agreement than ours, even though in their study the mean number of visual fields in each series was 6.3 whereas in the present study it was fixed at 16. This suggests that the observers in the present study did not benefit from being asked to analyse relatively more visual field data per subject. In fact, longer periods of visual field follow-up may actually make the task of deciding about progression status more difficult and complex, when this task is based on standard automated visual field printouts. Werner and colleagues found that all the observers agreed about progression status in 11 of their 30 subjects. The present study found that there was complete agreement about 10 of the 27 subjects. Although Werner and colleagues used six observers whereas the present study used five these results appear comparable. Once again, the level of agreement between observers in rating visual field progression based on standard automated perimeter output does not seem to have been influenced by the relatively longer follow-up in the present study.

A consistently higher level of inter- and intra-observer agreement was found when PROGRESSOR was used to rate progression status rather than standard perimeter output. This suggests that clinicians are better able to make meaningful, systematic decisions about visual field progression status when using PROGRESSOR rather than standard automated perimeter output. Chauhan and colleagues conducted a similar study to the present one in which five observers rated the progression status of 32 visual field series using a computer animated graphics technique which corrected for test-retest variability in order to aid the recognition of

progression.[105] They found a weighted kappa value of 0.572 which is very similar to the figure of 0.59 obtained in the present study. Chauhan and colleagues found that at least four out of five observers (80% or greater concordance) agreed on progression status in 27 out of 32 visual field series (84%). This is remarkably similar to our findings of 80% or better concordance on 23 out of 27 field series (85%). The level of complete agreement between all five observers (100% concordance) is, however, greater in the present study: 74% as compared to Chauhan and colleagues' figure of 56%. Both these studies suggest that the level of agreement about progression status rises when clinicians analyse displays of visual field series which highlight information pertaining to progression which is not obvious in the standard perimeter outputs.

The observers in the present study were given no explicit guidelines upon which to base the diagnosis of stability or progression for each visual field series. They were merely asked to use their clinical expertise. Thus it is possible that a different group of observers might have used different personal progression criteria which would have led to different results. In the absence of a gold standard for the diagnosis of glaucomatous visual field progression, however, the analytical task presented to the observers in the present study resembles the conditions encountered in clinical practice more closely than if predetermined progression criteria were imposed. The comparability of our results with those of previous workers suggests that different groups of expert observers may use similar personal progression criteria when analysing visual field series consisting of standard perimeter outputs. Although the level of agreement between the observers was similar for PROGRESSOR and for the glaucoma change analysis described by Chauhan and colleagues, no comparison can be made between the personal progression criteria used by the observers in the two studies, since the analytical task of diagnosing progression or stability using PROGRESSOR is quite different from that using the technique described by Chauhan and colleagues.

The use of PROGRESSOR led to the observers classifying a higher proportion of the field series as progressing than when the standard printouts were used. In the absence of an external gold standard for visual field progression, this may indicate either an increased sensitivity in the detection of progression or alternatively a reduction in specificity. There is some evidence that the reliability of

PROGRESSOR in detecting visual field deterioration in glaucoma compares favourably with other automated analyses such as Statpac 2.[162] Furthermore, a greater number of visual field series were classified as stable with 100% concordance using PROGRESSOR than using standard Humphrey printouts: in only one of the eight field series classified as stable using PROGRESSOR was there less than 100% concordance.

It is likely that new developments and refinements in the area of computer-assisted diagnosis of visual field progression in glaucoma will yield higher levels of agreement between clinicians in the future. However, the interpretation of any algorithm and its application to patient management within the clinical context will remain a subjective matter of clinical expertise.

CHAPTER 5. PATIENT PERCEPTION OF VISUAL FIELD STATUS

5.1 Abstract

Objectives: To elucidate the relationship between glaucoma patients' subjective assessment of 1) the severity of their visual loss and 2) any deterioration in their visual function, and their objective visual fields as measured by computerized perimetry.

Design: 1) Patients completed a questionnaire relating to perceived visual disability and underwent binocular visual field testing. 2) A separate group of patients answered a question about perceived visual deterioration: their monocular field tests were analysed retrospectively by pointwise linear regression to establish stability or deterioration. *Setting*: The Glaucoma Service of a specialist eye hospital which is a tertiary referral centre and serves the local community. *Subjects*: 123 glaucoma patients comprising 62 for the 'severity' arm of the study and 61 for the 'progression' arm. *Main outcome measures*: Questionnaire responses. Binocular Esterman Disability Score. Objective visual field deterioration.

Results: Questions strongly associated with Binocular Esterman Disability Scores related to bumping into things, problems with stairs, and finding things which have been dropped. There was a strong association between perceived visual deterioration and measured bilateral progression ($p < 0.01$).

Conclusions: There is a strong association between some types of perceived visual disability and the severity of binocular field loss. A patient who notices gradual visual deterioration is twice as likely to have bilateral field progression as not. These findings, in this sample of patients with mild to moderate glaucoma, challenge the belief that glaucoma is an insidious process whose symptoms do not appear until the end stage of the disease.

(Viswanathan, A.C., A. I. McNaught, D. Poinoosawmy, L. Fontana, D. P. Crabb, F. W. Fitzke, R. A. Hitchings. Severity and Stability of Glaucoma: Patient Perception compared with Objective Measurement. *Archives of Ophthalmology* 1999, **117**: 450-454.)

5.2 Introduction

Two of the most important facets of a glaucoma patient's disease status are the extent of established optic nerve damage and the rate of progression of this damage. The combination of these static and dynamic components has a critical bearing on the patient's present and future visual function, and on the clinical management employed. This study was designed to investigate the relationship between objective measurements of these two components and subjective patient perception of them.

5.2.1 Severity of visual field damage

When automated perimetry is used to measure visual field damage in glaucoma, monocular testing is usually employed in order to be able to tailor therapy to each eye. However, the patient's visual disability through field damage is better assessed using binocular testing: binocular visual field tests are routinely used in order to measure whether a patient's visual field is legally adequate for driving.[155] The most widely used strategies for testing and scoring the binocular visual field are based on the system devised by Esterman (see 1.2.4.2). This system was originally for use with monocular fields[163] and was then adapted for use with binocular fields.[164] Test points are more closely spaced in areas of the field considered more functionally important (Figure 5.1). This gives additional weight to these areas when the Esterman Disability Score (the proportion of points missed) is calculated.

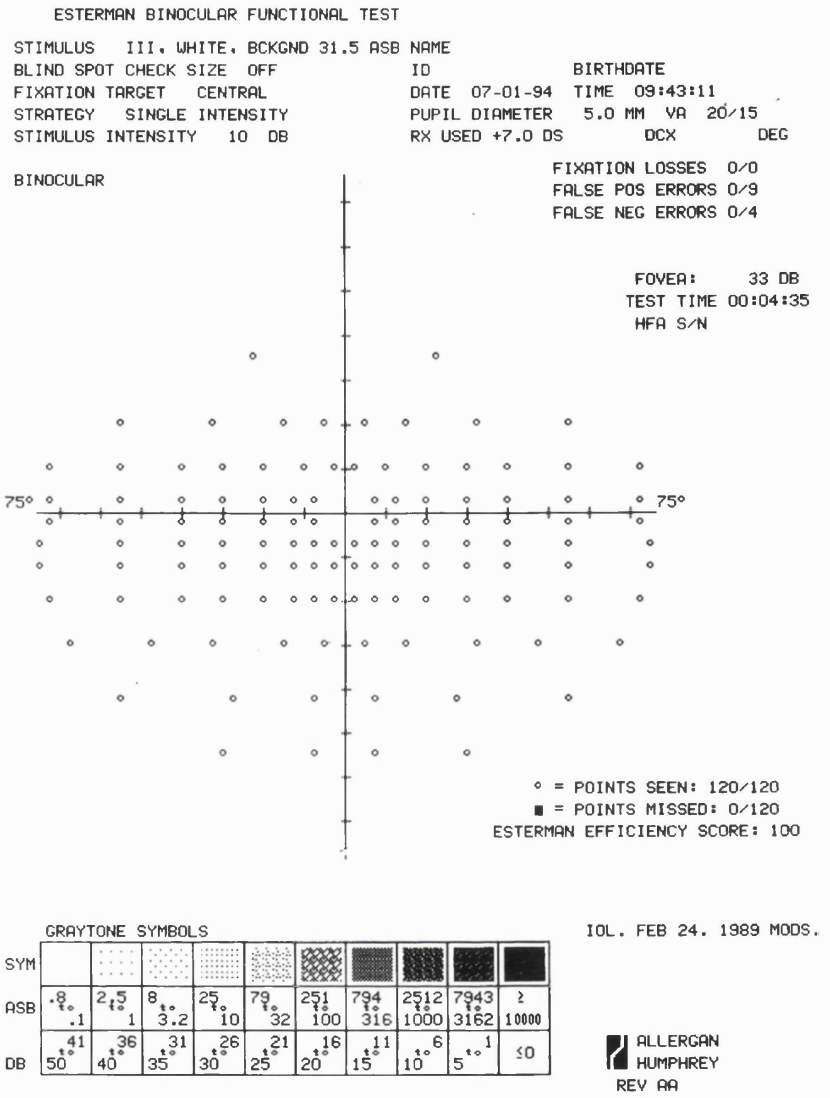


Figure 5.1 *Example of a printout of a Humphrey Field Analyzer Esterman strategy test.*

In a study of patients with severe glaucomatous field loss a moderate association was found between some questions about visual disability and the degree of binocular visual field damage as measured by Esterman Disability Score.(165) The modest ability of the questions as predictors of the Esterman Disability Score was ascribed to denial of visual disability and the use of adaptive strategies. One of the aims of the present study was to examine the strength of association between subjective patient perception of visual disability (using questions similar to those

used by previous workers[165] to enable comparison) and objective measurements of functional vision (Esterman Disability Score).

5.2.2 Rate of visual field progression

The PROGRESSOR software for the analysis of serial visual fields to detect glaucomatous change has been described in 1.3.1.4(ii). The pointwise linear model has been demonstrated to provide a valid framework for detecting and forecasting glaucomatous loss[120] and has been found to compare favourably with other pointwise methods of analysis such as the Statpac 2 Glaucoma Change Probability analysis.[124]

This study investigates the relationship between perceived visual field deterioration (assessed by questionnaire) and objectively measured field progression (using PROGRESSOR).

5.3 Methods

5.3.1 Severity of visual field damage

In order to compare severity of visual field damage with perceived visual disability, patients consecutively attending the Glaucoma Service at Moorfields Eye Hospital for visual field testing were studied. All patients gave informed consent for the study: there were no refusals to participate. Visual acuity in each eye was required to be better than 6/12 and eligible patients had no ocular pathology apart from primary open angle glaucoma. 62 patients met these inclusion criteria. One of the investigators (DP) administered a brief binary forced-choice questionnaire to each patient (Table 5.1). The questionnaire consisted of 10 questions related to visual disability derived from those validated in previous published research in this area.[165]

1	Do you ever notice that parts of your field of vision are missing?
2	Have you noticed any deterioration in your sight over the last few years?
3	Do you ever have trouble following a line of print or finding the next line when reading?
4	Do you notice variation in colour intensity?
5	Do you bump into things sometimes?
6	Do you trip on things or have difficulty with stairs?
7	Have you had to give up activities because of your sight?
8	Do you have difficulty finding things that you have dropped?
9	Are you troubled by glare or 'dazzled' on sunny days or in bright lighting?
10	Do you have particular difficulty seeing after moving from a light to a dark room?

Table 5.1 *Visual disability questionnaire*

Patients then underwent a binocular visual field test on a Humphrey model 630 perimeter using the 120 point Esterman strategy program. The Esterman Disability Score represents the proportion of points missed during the test. Thus as the Esterman Disability Score increases, disability increases. The strength of association between responses to the questionnaire and Esterman Disability Scores was investigated using stepwise multiple regression and correlation analysis. In

these analyses, an attempt is made to ascertain which of the questions are good predictors of the Esterman Disability Score. However it is also possible to regard each of the questions as a possible outcome of the Esterman Disability Score, since poor binocular field may lead to problems with the performance of visual tasks. For this reason, the data were also analysed using logistic regression: in this analysis the Esterman Disability Score is regarded as the independent variable and each question as a separate dependent variable.

5.3.2 Rate of visual field progression

61 consecutive untreated patients attending the normal tension glaucoma clinic at Moorfields Eye Hospital were studied. All patients gave informed consent for the study: there were no refusals to participate. Untreated patients were chosen as it was felt to be important to analyse the natural history of glaucomatous visual damage and the concomitant perception of that damage in the absence of the potentially confounding effects of therapy. Patients were untreated if no evidence of progression using global indices could be detected in the presence of open-angle glaucoma with normal intraocular pressure. Subsequent detection of progression using pointwise analysis, combined with the demonstration that a 25-30% lowering of intraocular pressure reduces the rate of progression,[147 150] led to treatment being recommended. Patients had no ocular pathology apart from normal tension glaucoma. All patients had a series of at least 5 visual fields over 16 months in each eye. The maximum number of visual fields performed during follow-up was 23 and the maximum length of follow-up was 9.6 years.

Patients were asked the single question "Have you noticed any deterioration in your sight over the last few years?" and were allowed a choice of yes or no. The visual field series for each eye was then assessed for progression or stability using PROGRESSOR. Patients were unaware of the results of the PROGRESSOR analysis at the time that the question was asked. A visual field series was regarded as progressing if PROGRESSOR identified at least one test location with a negative slope of 1 dB per year or worse associated with $p < 0.01$ for a two tailed t-test of the slope against zero (i.e. the null hypothesis of no deterioration). The slope criterion of 1 dB per year represents a rate of sensitivity loss approximately

ten times greater than the normal age-related decline.[97] Edge points are known to be more subject to fluctuation[97] so a stricter slope criterion of 2 dB per year (also with $p < 0.01$) was introduced for them. These slope criteria, in combination with a less stringent slope significance criterion of $p < 0.1$, have been demonstrated to compare closely with the Humphrey Statpac 2 Glaucoma Change Probability analysis.[119 124] The association between monocular or binocular progression and subjective perception of perception was examined with the χ^2 statistic.

5.3.3 Statistical analysis

Statistical analysis was performed using the software package SPSS for Windows, version 6.0 (SPSS Inc., Chicago, Ill., USA).

5.4 Results

5.4.1 Severity of visual field damage

Binocular Esterman Disability Scores ranged from 0 to 91.7. The mean was 12.85 (SD 18.12) and the median was 7.08. The distribution of Binocular Esterman Disability Scores is shown as a histogram in Figure 5.2.

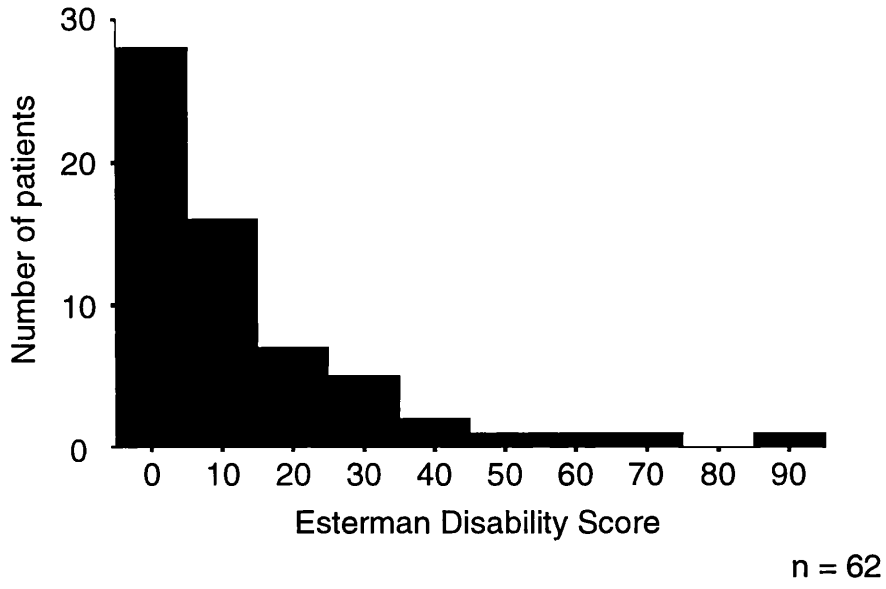


Figure 5.2 *Histogram of Esterman Disability Scores.*

Stepwise multiple regression identified 3 questions as predictors of the Binocular Esterman Disability Scores: question 5 (Do you bump into things sometimes?), question 6 (Do you trip on things or have difficulty with stairs?) and question 8 (Do you have difficulty finding things that you have dropped?). The adjusted R^2 resulting from the analysis was 0.34.

For comparison with previous studies[165] a correlation coefficient value of 0.4 or greater was taken to indicate a noteworthy association between the question concerned and the Binocular Esterman Disability Score. Questions satisfying this criterion were the same as those identified by the multiple regression analysis

(questions 5, 6 and 8, *vide supra*) with the addition of question 4 (Do you notice variation in colour intensity?).

Logistic regression identified the same questions as correlation analysis (questions 4, 5, 6 and 8). It also identified question 1 (Do you ever notice that parts of your field of vision are missing?), question 3 (Do you ever have trouble following a line of print or finding the next line when reading?) and question 10 (Do you have particular difficulty seeing after moving from a light to a dark room?)

These results are summarised in Table 5.2.

Question number	Multiple regression T-ratio	Correlation coefficient	Logistic regression p-value
1) Do you ever notice that parts of your field of vision are missing?			0.0096
3) Do you ever have trouble following a line of print or finding the next line when reading?			0.0146
4) Do you notice variation in colour intensity?		0.405	0.0088
5) Do you bump into things sometimes?	2.10	0.514	0.0022*
6) Do you trip on things or have difficulty with stairs?	2.02	0.509	0.0024*
8) Do you have difficulty finding things that you have dropped?	1.93	0.546	0.0012*
10) Do you have particular difficulty seeing after moving from a light to a dark room?			0.0097

Table 5.2 *Questions associated with Esterman Disability Score.*

**These questions are significant at the 5% level after Bonferroni correction[166] for multiple comparisons.*

5.4.2 Rate of visual field progression

Of the 61 patients questioned, 29 reported a subjective perception of gradual deterioration of vision and 32 reported no such perception. 22 of the 29 patients reporting visual deterioration were demonstrated by PROGRESSOR to have field progression in at least one eye: 16 of these patients had progression in both eyes. Of the 32 patients reporting no deterioration in vision, 19 were found by

PROGRESSOR to be progressing in at least one eye and 5 had field progression in both eyes. These findings are summarised in Tables 5.3 and 5.4. The χ^2 statistic indicates a significant association ($p < 0.01$) between subjectively perceived visual deterioration and measured binocular progression. If a subjective report of progression is regarded as a test for binocular progression, its sensitivity is 16/21 = 76.2% and specificity is 27/40 = 67.5%.The corresponding figures for monocular progression in either eye are a sensitivity of 22/41 = 53.7% and a specificity of 13/20 = 65%. The likelihood ratio (the odds that a patient reporting visual deterioration has measurable binocular progression) is 76.2/(100-67.5) = 2.34. Of the 16 binocularly progressing patients who were aware of visual field deterioration, 15 showed either progression at the same location in each eye or progression in one eye corresponding to a previously established scotoma in the other eye. Of the five binocularly progressing patients who were unaware of visual field deterioration, 3 showed this type of behaviour but 2 did not.

		Perceived progression		
		No	Yes	**Total**
Measured	No	13	7	**20**
progression	Yes	19	22	**41**
(either eye)	**Total**	**32**	**29**	**61**

Table 5.3 *Association between perceived progression and measured progression in either eye.*

$p = 0.170$ (χ^2)

		Perceived progression		
		No	Yes	**Total**
Measured	No	27	13	**40**
progression	Yes	5	16	**21**
(binocular)	**Total**	**32**	**29**	**61**

Table 5.4 *Association between perceived progression and measured binocular progression.*

$p < 0.01$ (χ^2)

5.5 Discussion

As mentioned previously, the present study was partly inspired by work carried out by Mills and Drance in 1986.[165] They selected 42 glaucoma patients with severe visual loss and examined the association between questionnaire responses and Esterman Disability Scores. Although the methods used in the present study and those of Mills and Drance are similar, the aims and patient selection criteria of the studies are quite different: Mills and Drance selected patients on the basis of severe visual loss (in 20 of their 42 patients, glaucoma was not the primary reason for impaired vision) whereas the present study concentrated on patients with good visual acuity in whom glaucoma was the primary reason for visual loss. Given these differences, it is interesting that three out of the five questions which Mills and Drance found to have a correlation coefficient of greater than 0.4 were also identified as such in the present study: these were questions related to colour intensity (q4), bumping into things (q5) and tripping (q6). Mills and Drance described four questions as important according to both stepwise regression and correlation analysis:

- Do you have trouble following a line of print or finding the next line?
- Do you bump into things?
- Have you had to give up any activities because of your vision?
- Do you notice any variation in colour richness from time to time?

The present study found that three questions were consistently identified as strongly associated with Esterman Disability Scores by stepwise regression, correlation analysis and logistic regression:

- Do you bump into things sometimes?
- Do you trip on things or have difficulty with stairs?
- Do you have difficulty finding things that you have dropped?

These findings suggest that trouble following print and having to give up activities are a feature of the severe visual disability more prevalent in the patient group studied by Mills and Drance, whereas bumping into things is related to the level of visual disability for glaucoma patients with mild to moderate field damage as well

as those whose vision is more severely impaired. The question about tripping was excluded from the multiple regression analysis in Mills and Drance's study as it was found to be closely related to the question about bumping into things. This was not the case in the present study, perhaps because the addition of the phrase '...or have difficulty with stairs?' was sufficient to differentiate it from the question about bumping into things. The question 'Do you have difficulty finding things that you have dropped?' which the present study found to have an important association with binocular visual disability was derived from the question 'Do you have trouble locating things?' which Mills and Drance did not find to be associated with Esterman Disability Score. This difference may be a result of the difference between the patient groups in the two studies or it may be because the former question is related to a specific visual task: patients may find it easier to remember difficulty, or the lack of it, in this area. The fact that the three questions identified by this study as strongly associated with Esterman Disability Score all relate to visual tasks primarily involving the inferior visual field may reflect the fact that the Esterman Test grid is biased towards the inferior visual field.

Although the three questions (questions 5, 6 and 8, Table 5.2) mentioned earlier were found to have a strong association with Esterman Disability Score by all three methods of analysis (stepwise regression, correlation analysis and logistic regression) the adjusted R^2 of the multiple regression analysis is 0.34: the questions only 'explain' 34% of the variance of the Esterman Disability Score, so they are not viable as predictors of it. (Mills and Drance found an adjusted R^2 of 48% for their 'explanatory' questions.)

The findings of an association between field progression in both eyes and the subjective perception of visual deterioration, but no association between field progression in at least one eye and subjective deterioration, suggests that a stable visual field in one eye tends to 'mask' the perception of progression in the other. It is unlikely that the degree of initial visual field damage is a confounding variable for the perception of visual deterioration: if this were the case, question 2 of the questionnaire ('Have you noticed any deterioration in your sight over the last few years?') would have been found to have an association with Binocular Esterman Disability Score. It is noteworthy that, of the 21 patients with measured binocular progression, 5 were unaware of any visual deterioration. This is reflected in the

poor performance of perceived progression as a marker of objective binocular progression (sensitivity 76.2%, specificity 67.5%) and the modest likelihood ratio: a patient reporting visual deterioration was 2.34 times more likely to have objective binocular progression than not.

Patients' knowledge of their diagnosis and information contained about the status of their eye health during previous visits would be likely to influence their answers to questions relating to visual function. For this reason the findings of this study cannot be generalised to newly diagnosed or undiagnosed glaucoma patients.

Previous studies have demonstrated moderate correlation between the perceived visual disability and visual field damage as measured by automated perimetry in severe glaucoma[167] and moderate/severe glaucoma.[168 169] Taken together, the results of this study suggest that, in mild to moderate glaucoma, both the existing severity of visual field damage in glaucoma and its rate of progression are associated with corresponding subjective perceptions of visual disability and deterioration. This challenges the belief that glaucoma is an insidious process whose symptoms do not appear until the end stage of the disease. However, patient perception is a poor predictor of objective measurement of the visual field: the latter will remain paramount as means of measuring the visual consequences of the disease, or of any therapy.

CHAPTER 6. SPATIAL FILTERING: IMPROVING REPRODUCIBILITY

6.1 Abstract

Background: PROGRESSOR for Windows is a computerized system for the analysis of glaucomatous field progression incorporating a graphical user interface. The software package includes spatial filtering, which has been shown to reduce long-term fluctuation.[153] However, some spatial processing, such as Gaussian filtering, may 'blur' early focal defects.

Purpose: To determine the role of Gaussian filtering in the early detection of glaucomatous loss.

Methods: 19 field series from untreated normal tension glaucoma patients, which an experienced observer judged as deteriorating, were studied. The time taken from the start of each series until progression criteria (slope worse than -1 dB/year for inner points, -2 dB/year for edge points, $p < 0.05$) were satisfied by at least one retinal location was calculated with and without Gaussian filtering.

Results: The unfiltered fields detected progression earlier than the filtered fields in 3 of the 19 field series (mean 1.18 years, SD 0.30 years). The filtered fields detected progression earlier in 5 series (mean 1.04 years, SD 1.48 years). Both filtered and unfiltered fields detected progression at the same test in 11 field series. There was no predominance of focal defects in the series where progression was detected earlier by the unfiltered fields.

Conclusions: PROGRESSOR for Windows is a convenient software tool for analysing glaucomatous field decay. Gaussian filtering reduces long-term fluctuation without delaying the detection of early loss in normal tension glaucoma.

(Viswanathan AC, Hitchings RA, Fitzke FW. Spatial Filtering of Glaucomatous Visual Fields using PROGRESSOR for Windows. In: Wall M, Heijl A, editors.

Perimetry Update. Amsterdam / New York: Kugler Publications, 1996/1997:311-319.)

6.2 Introduction

Visual field series obtained by automated perimetry provide both spatial and temporal information about disease progression in glaucoma patients. It is crucial that the rate of progression is accurately quantified since this provides a measure of the need for, and the efficacy of, treatment. This task is hampered by the inherent variability between field tests (long-term fluctuation[153]) which is known to be greater in glaucoma.[170] The benefits of using spatial image processing techniques, such as Gaussian filtering, in order to reduce the effects of long-term fluctuation have been described in 1.3.2.1. However, it is possible that spatial filtering will smooth areas of the field representative of early focal decay, rather than simply noise, and valuable information will be lost.

As described in 3.6, the PROGRESSOR software incorporates a Gaussian filtering process which may be applied to any visual field series. This study was designed to investigate, using PROGRESSOR, whether spatial filtering leads to a clinically significant delay in the detection of glaucomatous field progression.

6.3 Methods

6.3.1 Subjects

Patients with untreated normal tension glaucoma which had been confirmed by phasing were chosen for study as it was felt to be important to analyse the natural history of glaucomatous visual damage in the absence of the potentially confounding effects of therapy. All subjects had visual acuity of 6/12 or better. None had ocular pathology apart from normal tension glaucoma. All subjects were experienced in Humphrey 30-2 tests and able to produce reliable computerized visual fields (less than 30% fixation losses and false negatives and less than 15% false positives). Each had had at least 2 tests over 4 months prior to the observation period: this is sufficient to obviate any learning effects[134 154] which may delay the diagnosis of progression.

Patients were chosen whose visual fields appeared to be progressively deteriorating in a typically glaucomatous manner, since it is in these patients that the 'blurring' associated with spatial filtering is most likely to cause delay in the detection of progression. This was done by inspection of Statpac overview printouts by an experienced observer.

On the basis of the foregoing criteria, 19 eyes from 13 subjects were selected. An indication of the degree of glaucomatous damage in the selected group is given by the following summary measures of the Mean Deviation (MD) of the initial field in each test series: the mean of the MDs was -6.81 dB (SD 6.01 dB), the median was -5.43 dB and the range was -22.40 dB to +1.07 dB.

6.3.2 Testing strategy

All tests were performed on a standard Humphrey automated perimeter. The 30-2 full threshold program with standard 4-2 dB single reversal strategy was used throughout. Tests were performed at intervals of approximately four months.

6.3.3 Progression criteria

PROGRESSOR performs linear regression of sensitivity on time for each test location. For each field in the series, each location is ascribed a change slope (in decibels per year) and a p value for a two tailed t-test of the slope against zero (i.e. the null hypothesis of no deterioration). A field series was regarded as progressing if any non-edge location showed a deterioration of 1 dB per year or worse with $p < 0.05$. A more stringent slope criterion of 2 dB per year was applied to the edge points of the 30-2 test grid. These slope criteria have been demonstrated to compare closely with the Humphrey Statpac 2 Glaucoma Change Probability analysis, and the p value criterion is stricter than that required to emulate Statpac 2.[119]

6.3.4 Detection time

The detection time for a field series was defined as the time interval between the initial field and the field when the progression criteria *(vide supra)* were first satisfied.

6.3.5 Spatial filtering

Gaussian low-pass filtering is widely used to remove noise from pixel-based images.[171 172] In order to apply the filter to a 30-2 field, each test location is regarded as the centre of a 3 x 3 neighbourhood to which the filter (convolution mask) is applied (Figure 6.1). Thus each point is influenced by the weighted sensitivity of its contiguous neighbours. At the edges of the field the mask is only partial since edge points do not have a full complement of neighbours.

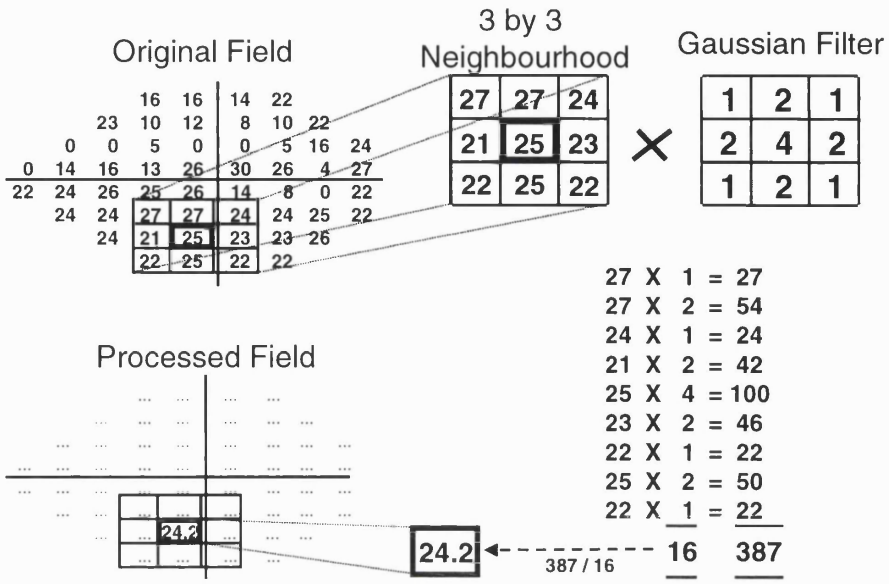


Figure 6.1 *Gaussian filtering process.*

6.3.6 Statistical analysis

For each field series, detection time (see 6.3.4) was calculated for both filtered and unfiltered data. Detection times for the unfiltered data were compared with their correlates for the filtered data using a non-parametric test for paired data from two related samples (Wilcoxon Signed Rank Z test). In addition the following summary measures of progression, automatically calculated by PROGRESSOR for Windows, were compared for all field series before and after spatial filtering: number of progressing points at detection time, mean slope for all test locations at detection time and mean slope for progressing points at detection time.

Statistical analysis was performed using the software package SPSS for Windows, version 6.0 (SPSS Inc., Chicago, Ill., USA), except for the power calculation which was performed with Jandel SigmaStat for Windows, version 1.0 (Jandel GmbH, Erkrath, Germany).

6.4 Results

6.4.1 Detection times

All visual field series satisfied the progression criteria both before and after spatial filtering. The unfiltered fields had a mean detection time of 1.077 yr. (SD 0.985 yr.) and the filtered fields had a mean detection time of 0.989 yr. (SD 0.639 yr.). These are not significantly different ($p = 0.779$, Wilcoxon Signed Rank Z test). Kolmogorov - Smirnov goodness of fit normality testing yielded $p = 0.260$ for the detection times of the unfiltered data and $p = 0.394$ for the detection times after filtering. On this basis a paired t-test is justified and gives $p = 0.709$: a power calculation (with α set at 0.05 and SD set at 0.985 yr.) gives a power of 0.797 to detect a difference of 8 months in mean detection time between filtered and unfiltered fields.

The unfiltered fields detected progression earlier in 3 field series: for these 3 series the mean delay in detection for the filtered fields was 1.18 yr. (SD 0.30 yr.). The filtered fields detected progression earlier in 5 field series: for these series the mean delay in detection for the unfiltered fields was 1.04 yr. (SD 1.48 yr.). Both filtered and unfiltered fields detected progression at the same test in 11 field series. These findings are displayed graphically in Figure 6.2, which is a histogram of the differences between filtered and unfiltered detection times for each field series. The distribution is centred around zero, which suggests no overall delay associated with filtering.

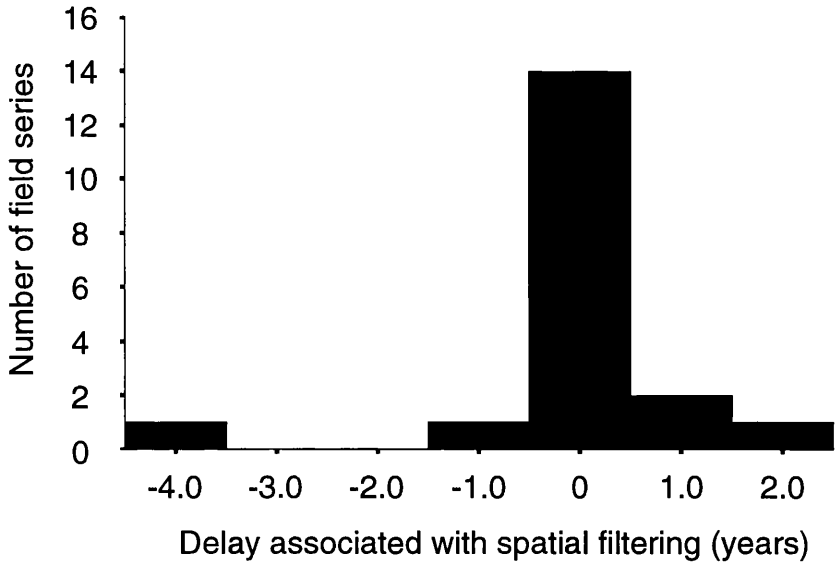


Figure 6.2 *Histogram of delay in detection time associated with spatial filtering.*

6.4.2 Number of progressing points at detection time

The mean number of progressing points which satisfied the progression criteria at detection time was 2.95 (SD 1.81) for the unfiltered fields and 3.00 (SD 2.49) after filtering. This difference was not significant (p = 0.784 , Wilcoxon Signed Rank Z test).

6.4.3 Mean slope (whole field) at detection time

The mean slope of all test locations at detection time had a mean value of -3.99 dB/year (SD 9.94 dB/ year) for the unfiltered fields and -3.76 dB/ year (SD 9.84 dB/ year) after filtering. This difference was not significant (p = 0.334, Wilcoxon Signed Rank Z test).

6.4.4 Mean slope (progressing points) at detection time

The mean slope of the locations which satisfied the progression criteria at detection time had a mean value of -22.47 dB/ year (SD 56.44 dB/ year) for the

unfiltered fields and -9.15 dB/ year (SD 14.14 dB/ year) after filtering. This difference was significant at the 5% level ($p = 0.030$, Wilcoxon Signed Rank Z test).

6.5 Discussion

Fluctuation owing to variable patient response and other factors is the main obstacle to the accurate quantification of the results of automated perimetry. Any technique which may reduce this fluctuation is therefore worthy of study. Spatial filtering has been shown to be effective in reducing noise in other two-dimensional digital image processing applications[171 172] and seems to be of benefit in the analysis of glaucomatous visual field progression.[129 130] However, by its very nature Gaussian filtering obscures elements of a digital image which have high spatial frequency. If the digital image concerned is a computerized visual field, these elements are small isolated scotomas and the edges of existing scotomas: these are precisely the areas where progression is most likely to occur first. Thus it is important to ascertain whether the process of spatial filtering leads to a clinically significant delay in detecting progression, since this would be a high price to pay for improved repeatability and predictability.

This study suggests that Gaussian filtering is unlikely to entail a significant delay in the detection of progression.

This study was performed on a highly selected group of patients: the selection criteria were necessarily strict in order to isolate cases in which spatial filtering was most likely to cause a delay in detection. Since detection time was not affected in this group it is likely that other, more stable visual field series will be similarly unaffected. However, further work is required in order to determine whether these results may be generalised to patients with other forms of glaucoma and to those who have received or are receiving medical treatment.

The Gaussian filter used in the study was chosen because it is a standard image processing convolution mask, and has been used in previous work on computerized visual fields.[129 130] However, it is unlikely to be the ideal mask for the special environment of automated perimetry. Firstly, the mask is usually intended for use with pixels which have a range of values (bandwidth) of between 0 and 255: the range of sensitivity values produced by automated perimetry is almost an order of magnitude smaller than this. Secondly, the mask is limited to contiguous neighbours and is symmetrical. This does not conform to our

knowledge of the patterns of glaucomatous field loss seen in clinical practice. For example, it seems likely that test locations will be more influenced by the behaviour of other locations within the same retinal nerve fibre bundle distribution than by those which are outside it: the closest topological neighbours may not be the closest 'functional' neighbours. It is possible that each test location will require its own unique convolution mask for optimum performance.

The difference between the filtered and unfiltered data with regard to the mean slope of progressing points at detection time reflects the fact that Gaussian filtering tends to lessen the slopes of the locations which are progressing at the greatest rate. Notwithstanding this, the fact that both the number of progressing points and the mean slope for the whole field at detection time are not affected by filtering suggests that, for these progression criteria, the overall 'progression status' of the field series are retained after filtering.

In summary, previous work has shown spatial filtering to be of potential benefit in the analysis of glaucomatous visual field progression.[129 130] This study has examined the combined use of spatial filtering and pointwise linear regression in the context of a software package, PROGRESSOR for Windows. The study suggests that the 'blurring' effect inherent in the Gaussian filtering process is not clinically significant.

CHAPTER 7. DISCUSSION

7.1 Work to date

The core of this thesis is the PROGRESSOR software, which provides an environment within which to apply the technique of pointwise linear regression to the analysis of serial glaucomatous visual field examinations. It is described in Chapter 3 and full source code is listed in appendix A. A demonstration version of the program is provided on the CDROM which accompanies this thesis.

The PROGRESSOR software has several advantages over existing methods used to chart the progress of visual function in glaucoma patients. As demonstrated in this thesis, PROGRESSOR detects visual field deterioration earlier than Statpac 2 (Chapter 4) and allows for a higher level of agreement between expert clinicians about the degree of progression in a given series of visual field tests than when standard clinical methods are used (Chapter 5). PROGRESSOR also allows the use of Gaussian filtering, which lessens the effects of long-term fluctuation without incurring a delay in the detection of progression (Chapter 6). Furthermore, PROGRESSOR is convenient to use in the clinical setting: since the algorithms used are rapid, an analysis tailored to the patient may be performed in real time by the clinician during a clinic visit without the imposition of extra testing.

However, PROGRESSOR also has both theoretical and practical weaknesses. As mentioned previously (1.3.1.4(iii)), PROGRESSOR analyses the behaviour of each test location without reference to the behaviour of the rest of the visual field. Whilst this approach may be sensitive in the detection of the first signs of glaucomatous visual field damage, it fails to take into account the characteristic patterns of visual field loss seen in glaucoma which are familiar to clinicians. Although Gaussian filtering may provide a means for the influence of neighbouring test locations to be taken into account, the Gaussian filter has several shortcomings, as detailed in 6.5, and is unlikely to be the most appropriate filter for the analysis of automated visual field tests in glaucoma. Potential methods for improving the filtering process are described in 7.2.3. From a practical standpoint,

the adoption of PROGRESSOR into a glaucoma service poses problems: clinicians require the necessary expertise to use the system, and a computer network must be in place in order for each clinician to be able to access a central database of visual field tests downloaded from automated perimeters. From experience at Moorfields Eye Hospital, new users of PROGRESSOR who are familiar with the Windows operating systems do not require a long training period to master the software, as it conforms to Windows standards. The resources required to equip each clinician's desk with a networked computer are unlikely to be justified purely for the analysis of visual field tests. Fortunately, however, this is unlikely to be the case: the impetus for large-scale networking within hospital clinics in the UK is likely to come from the growth of electronic patient records systems.

7.2 Future work

7.2.1 Frequency of visual field testing

The study described in Chapter 2 has demonstrated that, with respect to the time intervals between visual field tests commonly used in clinical practice, more frequent visual field testing detects glaucomatous visual field progression sooner. Although this is a useful finding which may inform clinical decision-making, the retrospective nature of the study precluded determination of the optimum frequency of visual field testing since the intervals between visual field tests which could be studied were limited by the data available.

A more sophisticated analysis might enable an estimate of the optimum frequency of visual field testing to be made for a particular patient. This analysis would be based on information obtained either from a prospective study or from the investigation of intratest intervals on the ability to detect glaucomatous deterioration in visual fields produced by software modelling. As applied to the individual patient, it would not only involve a careful analysis of all areas of the visual field series in question, in order to ascertain which test locations may be undergoing relatively rapid change requiring close monitoring, but would also

need to address other factors as mentioned in 1.3.2.4: the type of glaucoma, recent changes in glaucoma therapy, the availability of perimetric resources and the ability of the patient to produce reliable tests. It is also possible that the emerging information about the genetics of glaucoma will be of prognostic use in the future: this is also likely to guide clinicians as to how closely to monitor individual glaucoma patients.

The development of adaptive strategies in visual field testing may allow for more frequent but less time consuming visual field tests. Testing will be directed mainly towards obtaining an accurate measure of those areas of the visual field predicted to be at risk of deterioration from previous testing, with less time spent on the examination of those areas of the visual field thought to be stable.

7.2.2 Quality of life

The study described in Chapter 5 indicates that there is a strong association between some types of perceived visual disability and binocular visual field status, even in patients with only mild to moderate glaucoma. An extension of the work described in Chapter 5 would enable identification of the levels of binocular visual field damage associated with various types of visual disability. These "milestones to blindness" might be used in the clinical situation as undesirable end points of visual deterioration. A larger study is now planned to elucidate the detailed patterns and degree of visual field loss associated with these types of perceived visual disability in glaucoma patients. Together with knowledge of glaucoma patients' current visual field status and the rate of deterioration of visual field, "milestones to blindness" may be used to rationalise therapy in those patients who require it and to avoid iatrogenic complications in those who do not.

It is possible to combine the results of bilateral monocular visual field tests to produce an accurate estimate of the central binocular visual field.[156] If a glaucoma patient were found during routine follow-up to have estimated binocular visual field damage consistent with certain types of visual disability, the patient could be asked specifically about their level of performance in these areas and could receive appropriate help sooner than they otherwise might.

Pointwise linear regression provides a measure of the rate of visual field deterioration at each test location. This provides a valid basis for predicting the behaviour of the visual field.[120] Prediction of the future monocular visual field could be combined with modelling of the future binocular visual field. This in turn could be linked to a prediction of the level of visual disability in the future. Depending on the level of visual disability predicted, this information could be used either to reassure the glaucoma patient or to stress that a need for increased compliance or a change in therapy has arisen.

7.2.3 Image processing of visual field data

The study described in Chapter 6 has demonstrated that the increased reproducibility and predictability in visual field series obtained with the use of Gaussian filtering is not associated with a delay in the detection of visual field progression. However, the Gaussian filter suffers from several limitations as described in 6.5. In order to obtain a more suitable filter for the analysis of glaucomatous visual fields it is necessary to determine the functional relationships between the different test locations in the visual field. A pilot study[173] suggests that the technique of stepwise multiple regression may be used to formulate a map of the spatial interdependencies of the test locations in the visual field, together with the relative strengths of these interdependencies. This type of map may be used to determine a customised filter for each test location. Work is currently under way to design customised filters for each test location. The new filtering algorithm will be tested in its ability to improve reproducibility and predictability. Its performance will also be tested against existing methods using study designs similar to those described in Chapters 4 and 6.

7.2.4 PROGRESSOR

The PROGRESSOR software has proven a convenient and effective tool for the analysis of serial glaucomatous visual fields. The modular design of the program should allow new features and analyses to be added relatively easily. PROGRESSOR will continue to be developed both in terms of software design and analytical capability.

7.4.1 Software

The program was originally written for the 16-bit operating systems prevalent at the time. Although still compatible with modern machines, rewriting the code for use with 32-bit systems will allow for increased performance and the ability to take better advantage of features of modern operating systems such as multitasking and threading. Use of the Java™ programming language (Sun Microsystems, Palo Alto, CA, USA) will enable a greater degree of platform independence and will allow the program to be run over the World Wide Web.

7.4.2 Analysis

As new techniques such as prediction of the future visual field (see 7.2.2) and improved image analysis (see 7.2.3) are developed and validated, they will be incorporated into the software. Other refinements will include the generation of Total and Pattern Deviation plots, global indices and Glaucoma Hemifield Test, as available on the standard Humphrey perimeter at present. Work will also focus on the detection of potential outliers within visual field series, and methods for discriminating between progressive visual field loss caused by cataract and that caused by glaucoma.

REFERENCES

1. Harrington DO. *The Visual Fields. A Textbook and Atlas of Clinical Perimetry.* 4th ed. St.Louis: Mosby CV, 1976.
2. Hollows FC, Graham PA. Intra-ocular pressure, glaucoma, and glaucoma suspects in a defined population. *: Br J Ophthalmol* 1966;50(10):570-86.
3. Henson DB. *Visual Fields.* 1st ed: Oxford University Press, 1993.
4. Sommer A, Tielsch JM, Katz J, Quigley HA, Gottsch JD, Javitt J, et al. Relationship between intraocular pressure and primary open angle glaucoma among white and black Americans. The Baltimore Eye Survey. *: Arch Ophthalmol* 1991;109(8):1090-5.
5. Hayreh SS. Progress in the understanding of the vascular etiology of glaucoma. *Curr Opinion Ophthalmol* 1994;5(11):26-35.
6. von Graefe A. Ueber die Iridectomie bei Glaucom und uber den glaucoatosen prozess. *Graefe's Arch Ophthalmol* 1857;3:456-560.
7. Quigley HA, Sanchez RM, Dunkelberger GR, L'Hernault NL, Baginski TA. Chronic glaucoma selectively damages large optic nerve fibers. *: Invest Ophthalmol Vis Sci* 1987;28(6):913-20.
8. Pederson JE, Gaasterland DE. Laser-induced primate glaucoma. I. Progression of cupping. *: Arch Ophthalmol* 1984;102(11):1689-92.
9. Quigley HA. Reappraisal of the mechanisms of glaucomatous optic nerve damage. *: Eye* 1987;1(Pt 2):318-22.
10. Smith RJ. The Lang lecture 1986. The enigma of primary open-angle glaucoma. *: Trans Ophthalmol Soc U K* 1986;105(Pt 6):618-33.
11. Migdal C, Hitchings R. Control of chronic simple glaucoma with primary medical, surgical and laser treatment. *: Trans Ophthalmol Soc U K* 1986;105(Pt 6):653-6.
12. Jay JL, Murray SB. Early trabeculectomy versus conventional management in primary open angle glaucoma. *: Br J Ophthalmol* 1988;72(12):881-9.
13. Sommer A. Intraocular pressure and glaucoma. *: Am J Ophthalmol* 1989;107(2):186-8.

14. Vogel R, Crick RP, Newson RB, Shipley M, Blackmore H, Bulpitt CJ. Association between intraocular pressure and loss of visual field in chronic simple glaucoma. : *Br J Ophthalmol* 1990;74(1):3-6.

15. Sommer A, Katz J, Quigley HA, Miller NR, Robin AL, Richter RC, et al. Clinically detectable nerve fiber atrophy precedes the onset of glaucomatous field loss. : *Arch Ophthalmol* 1991;109(1):77-83.

16. Mao LK, Stewart WC, Shields MB. Correlation between intraocular pressure control and progressive glaucomatous damage in primary open-angle glaucoma. : *Am J Ophthalmol* 1991;111(1):51-5.

17. Weber J, Koll W, Krieglstein GK. Intraocular pressure and visual field decay in chronic glaucoma. *Ger J Ophthalmol* 1993;2(3):165-9.

18. Airaksinen PJ, Tuulonen A, Alanko HI. Rate and pattern of neuroretinal rim area decrease in ocular hypertension and glaucoma. : *Arch Ophthalmol* 1992;110(2):206-10.

19. Cartwright MJ, Anderson DR. Correlation of asymmetric damage with asymmetric intraocular pressure in normal-tension glaucoma (low-tension glaucoma). : *Arch Ophthalmol* 1988;106(7):898-900.

20. Holmin C, Krakau CE. Regression analysis of the central visual field in chronic glaucoma cases. A follow-up study using automatic perimetry. *Acta Ophthalmol Copenh* 1982;60(2):267-74.

21. O'Brien C, Schwartz B, Takamoto T, Wu DC. Intraocular pressure and the rate of visual field loss in chronic open-angle glaucoma. *Am J Ophthalmol* 1991;111(4):491-500.

22. Chauhan BC, Drance SM. The relationship between intraocular pressure and visual field progression in glaucoma. *Graefes Arch Clin Exp Ophthalmol* 1992;230(6):521-6.

23. Crichton A, Drance SM, Douglas GR, Schulzer M. Unequal intraocular pressure and its relation to asymmetric visual field defects in low-tension glaucoma. : *Ophthalmology* 1989;96(9):1312-4.

24. Haefliger IO, Hitchings RA. Relationship between asymmetry of visual field defects and intraocular pressure difference in an untreated normal (low) tension glaucoma population. : *Acta Ophthalmol Copenh* 1990;68(5):564-7.

25. Jaeger E. Ueber glaucom und seine heilung durch iridectomie. *Zeitsher Ges Aerzie zu Wien* 1858;**14**:465-491.

26. Kaiser HJ, Flammer J, Graf T, Stumpfig D. Systemic blood pressure in glaucoma patients. *: Graefes Arch Clin Exp Ophthalmol* 1993;231(12):677-80.

27. Hayreh SS, Zimmerman MB, Podhajsky P, Alward WL. Nocturnal arterial hypotension and its role in optic nerve head and ocular ischemic disorders. *: Am J Ophthalmol* 1994;117(5):603-24.

28. Drance SM, Douglas GR, Wijsman K, Schulzer M, Britton RJ. Response of blood flow to warm and cold in normal and low-tension glaucoma patients. *: Am J Ophthalmol* 1988;105(1):35-9.

29. Hoyng PF, de J-N, Oosting H, Stilma J. Platelet aggregation, disc haemorrhage and progressive loss of visual fields in glaucoma. A seven year follow-up study on glaucoma. *: Int Ophthalmol* 1992;16(2):65-73.

30. James CB, Smith SE. Pulsatile ocular blood flow in patients with low tension glaucoma. *: Br J Ophthalmol* 1991;75(8):466-70.

31. Butt Z, McKillop G, O'Brien C, Allan P, Aspinall P. Measurement of ocular blood flow velocity using colour Doppler imaging in low tension glaucoma. *: Eye* 1995;9(Pt 1):29-33.

32. Kitazawa Y, Shirai H, Go FJ. The effect of Ca2(+) -antagonist on visual field in low-tension glaucoma. *: Graefes Arch Clin Exp Ophthalmol* 1989;227(5):408-12.

33. Netland PA, Chaturvedi N, Dreyer EB. Calcium channel blockers in the management of low-tension and open-angle glaucoma. *: Am J Ophthalmol* 1993;115(5):608-13.

34. Schulzer M, Drance SM, Carter CJ, Brooks DE, Douglas GR, Lau W. Biostatistical evidence for two distinct chronic open angle glaucoma populations [see comments]. *: Br J Ophthalmol* 1990;74(4):196-200.

35. Foster A, Johnson GJ. Magnitude and causes of blindness in the developing world. *: Int Ophthalmol* 1990;14(3):135-40.

36. Quigley HA. Number of people with glaucoma worldwide. *: Br J Ophthalmol* 1996;80(5):389-93.

37. Thylefors B, Negrel AD. The global impact of glaucoma. *: Bull World Health Organ* 1994;72(3):323-6.

38. Coffey M, Reidy A, Wormald R, Xian WX, Wright L, Courtney P. Prevalence of glaucoma in the west of Ireland. *: Br J Ophthalmol* 1993;77(1):17-21.

39. Leibowitz HM, Krueger DE, Maunder LR, Milton RC, Kini MM, Kahn HA, et al. The Framingham Eye Study monograph: An ophthalmological and epidemiological study of cataract, glaucoma, diabetic retinopathy, macular degeneration, and visual acuity in a general population of 2631 adults, 1973-1975. *: Surv Ophthalmol* 1980;24(Suppl):335-610.

40. Bengtsson B. The prevalence of glaucoma. *: Br J Ophthalmol* 1981;65(1):46-9.

41. Tielsch JM, Sommer A, Katz J, Royall RM, Quigley HA, Javitt J. Racial variations in the prevalence of primary open-angle glaucoma. The Baltimore Eye Survey. *: Jama* 1991;266(3):369-74.

42. Klein BE, Klein R, Sponsel WE, Franke T, Cantor LB, Martone J, et al. Prevalence of glaucoma. The Beaver Dam Eye Study. *: Ophthalmology* 1992;99(10):1499-504.

43. Quigley HA, Vitale S. Models of open-angle glaucoma prevalence and incidence in the United States. *Invest Ophthalmol Vis Sci* 1997;38(1):83-91.

44. Airaksinen PJ, Heijl A. Visual field and retinal nerve fibre layer in early glaucoma after optic disc haemorrhage. *: Acta Ophthalmol Copenh* 1983;61(2):186-94.

45. Bengtsson BO. Incidence of manifest glaucoma. *: Br J Ophthalmol* 1989;73(7):483-7.

46. Leske MC. The epidemiology of open-angle glaucoma a review. *: Am J Epidemiol* 1983;118(2):166-91.

47. Leske MC, Connell AM, Schachat AP, Hyman L. The Barbados Eye Study. Prevalence of open angle glaucoma. *: Arch Ophthalmol* 1994;112(6):821-9.

48. Poinoosawmy D, Nagasubramanian S, Wormald R, Hitchings R. Glaucoma and race [letter]. *Lancet* 1989;1(8647):1134.

49. Shiose Y, Kitazawa Y, Tsukahara S, Akamatsu T, Mizokami K, Futa R, et al. Epidemiology of glaucoma in Japan--a nationwide glaucoma survey. *: Jpn J Ophthalmol* 1991;35(2):133-55.

50. Tielsch JM, Katz J, Sommer A, Quigley HA, Javitt JC. Family history and risk of primary open angle glaucoma. The Baltimore Eye Survey. *: Arch Ophthalmol* 1994;112(1):69-73.

51. Leske MC, Connell AM, Wu SY, Hyman LG, Schachat AP. Risk factors for open-angle glaucoma. The Barbados Eye Study. *: Arch Ophthalmol* 1995;113(7):918-24.

52. Teikari JM, Airaksinen JP. Twin study on cup/disc ratio of the optic nerve head. *Br J Ophthalmol* 1992;76(4):218-20.

53. Kitsos G, Cote G, Psilas K. [An example of dominant heredity in the transmission of primary open-angle glaucoma in a northwestern region of Greece]. *: J Fr Ophtalmol* 1988;11(12):859-64.

54. Sheffield VC, Stone EM, Alward WL, Drack AV, Johnson AT, Streb LM, et al. Genetic linkage of familial open angle glaucoma to chromosome 1q21-q31. *: Nat Genet* 1993;4(1):47-50.

55. Becker B. Diabetes mellitus and primary open-angle glaucoma. The XXVII Edward Jackson Memorial Lecture. *: Am J Ophthalmol* 1971;71(1):1-16.

56. Klein BE, Klein R, Jensen SC. Open-angle glaucoma and older-onset diabetes. The Beaver Dam Eye Study. *: Ophthalmology* 1994;101(7):1173-7.

57. Tielsch JM, Katz J, Quigley HA, Javitt JC, Sommer A. Diabetes, intraocular pressure, and primary open-angle glaucoma in the Baltimore Eye Survey. *: Ophthalmology* 1995;102(1):48-53.

58. Phelps CD, Corbett JJ. Migraine and low-tension glaucoma. A case-control study. *: Invest Ophthalmol Vis Sci* 1985;26(8):1105-8.

59. Klein BE, Klein R, Meuer SM, Goetz LA. Migraine headache and its association with open-angle glaucoma: the Beaver Dam Eye Study. *Invest Ophthalmol Vis Sci* 1993;34(10):3024-7.

60. Wilson MR, Hertzmark E, Walker AM, Childs S-K, Epstein DL. A case-control study of risk factors in open angle glaucoma. *: Arch Ophthalmol* 1987;105(8):1066-71.

61. Araie M, Arai M, Koseki N, Suzuki Y. Influence of myopic refraction on visual field defects in normal tension and primary open angle glaucoma. *Jpn J Ophthalmol* 1995;39(1):60-4.

62. Quigley HA, Enger C, Katz J, Sommer A, Scott R, Gilbert D. Risk factors for the development of glaucomatous visual field loss in ocular hypertension. *: Arch Ophthalmol* 1994;112(5):644-9.

63. Charliat G, Jolly D, Blanchard F. Genetic risk factor in primary open-angle glaucoma a case-control study. *: Ophthalmic Epidemiol* 1994;1(3):131-8.

64. Reiss GR, Wilensky JT, Higginbotham EJ. Laser trabeculoplasty. *Surv Ophthalmol* 1991;35(6):407-28.

65. Richter CU, Shingleton BJ, Bellows AR, Hutchinson BT, O'Connor T, Brill I. The development of encapsulated filtering blebs. *Ophthalmology* 1988;95(9):1163-8.

66. Cairns JE. Trabeculectomy. Preliminary report of a new method. *Am J Ophthalmol* 1968;66(4):673-9.

67. Khaw PT, Migdal CS. Current techniques in wound healing modulation in glaucoma surgery. *Current Opinion in Ophthalmology* 1996;7(2):24-33.

68. Migdal C, Gregory W, Hitchings R. Long-term functional outcome after early surgery compared with laser and medicine in open-angle glaucoma. *: Ophthalmology* 1994;101(10):1651-6.

69. RCOphth. Guidelines for the Management of Ocular Hypertension and Primary Open Angle Glaucoma: Headley Brothers Ltd., The Invicta Press, UK, 1998.

70. Hitchings RA. Focus: Glaucoma management. London, UK.: Royal College of Ophthalmologists, 1998.

71. Lachenmayr BJ, Vivell PM. *Perimetry and its clinical correlations.* 1st ed. New York: Georg Thieme Verlag, 1993.

72. Traquair HM. Perimetry in the study of glaucoma. *Trans Ophthalmol Soc UK* 1931:585-599.

73. von Graefe A. Ueber die Untersuchung des Gesichtsfeldes bei amblyopischen Affectionen. *Archive für Ophthalmologie* 1856;II(2):258-298.

74. Aubert H, Foerster R. Untersuchungen über den Raumsinn der retina. *Archive für Ophthalmologie* 1857; III(**2**):1-37.

75. Scherk S. Ein neuer Apparat zur Messung des Gesichtsfeldes. *Klin Mbl Augenheilk* 1872;**10**:151-163.

76. Bjerrum J. Om en tilføesle til den sædvanlige synsfeltundersögelse samt om synsfeltet ved glaukom. *Nordisk Ophthalmologisk Tidsskrift* 1889;2:141-185.

77. Rönne H. Ueber das Gesichtsfeld beim Glaukom. *Klin Mbl Augenheilk* 1909;47:12-33.

78. Goldmann H. Ein selbstregistrierendes Projektionskugelperimeter. *Ophthalmologica* 1945:71-79.

78a. Trobe JD, Acosta PC, Shuster JJ, Krischer JP. *Am J Ophthalmol* 1980;90(5):654-60.

79. Greve EL. Performance of computer assisted perimeters. *Doc Ophthalmol* 1982;53(4):343-80.

80. Koerner F, Fankhauser F, Bebie H, Spahr J. Threshold noise and variability of field defects in determinations by manual and automatic perimetry. *Doc Ophthal Proc Ser* 1977;**14**:53-59.

81. Johnson CA, Keltner JL, Balestrery FG. Suprathreshold static perimetry in glaucoma and other optic nerve disease. *Ophthalmology* 1979;86(7):1278-86.

82. Heijl A, Drance SM. A clinical comparison of three computerized automatic perimeters in the detection of glaucoma defects. *Arch Ophthalmol* 1981;99(5):832-6.

83. Weber J, Dobek K. What is the most suitable grid for computer perimetry in glaucoma patients? *: Ophthalmologica* 1986;192(2):88-96.

84. Katz J, Tielsch JM, Quigley HA, Sommer A. Automated perimetry detects visual field loss before manual Goldmann perimetry. *: Ophthalmology* 1995;102(1):21-6.

85. Spahr J. [Automation of the perimeter. I. Application of a computer-regulated perimeter (author's transl)]. *Albrecht Von Graefes Arch Klin Exp Ophthalmol* 1973;188(4):323-38.

86. Heijl A, Krakau CE. An automatic perimeter for glaucoma visual field screening and control. Construction and clinical cases. *Albrecht Von Graefes Arch Klin Exp Ophthalmol* 1975;197(1):13-23.

87. Norren DV. Automation of perimetry: the scoperimeter. *Ophthalmologica* 1976;173(3-4):227-9.

88. Greve EL. Peritest. *Doc Ophthal Proc Ser* 1980;**22**:71-74.

89. Fankhauser F. The development of computerised perimetry. In: Spaeth WRWaGL, editor. *Computerised visual fields: what they are and how to use them.* New Jersey: Slack, 1985:11-27.

90. Wild JM. Techniques and developments in automated perimetry: a review [see comments]. *Ophthalmic Physiol Opt* 1988;8(3):295-308.

91. Heijl A. The Humphrey Field Analyser: Concepts and clinical results. *Doc Ophthal Proc Ser* 1985;43:55-64.

92. Haley MJ. *The Field Analyzer Primer.* 2nd ed. San Leandro, CA.: Allergan Humphrey, 1987.

93. Werner EB. *Manual of visual fields.* New York: Churchill Livingstone, 1991.

94. Aulhorn E, Karmeyer H. Frequency distribution in early glaucomatous visual field defects. *Doc Ophthal Proc Ser* 1977;14:75-83.

95. Caprioli J, Spaeth GL. Static threshold examination of the peripheral nasal visual field in glaucoma. *Arch Ophthalmol* 1985;103(8):1150-4.

96. Heijl A, Lindgren G, Olsson J. A package for the statistical analysis of visual fields. *Doc Ophthal Proc Ser* 1987;49:153-168.

97. Heijl A, Lindgren G, Olsson J. Normal variability of static perimetric threshold values across the central visual field. *Arch Ophthalmol* 1987;105(11):1544-9.

98. Heijl A, Lindgren G, Olsson J, Asman P. Visual field interpretation with empiric probability maps [see comments]. *Arch Ophthalmol* 1989;107(2):204-8.

99. Weber J, Geiger R. Gray scale display of perimetric results. The influence of different interpolation procedures. In: Heijl A, editor. *Perimetry Update 1988/1989.* Amsterdam: Kugler and Ghedini, 1989:447-454.

100. Bebie H. Computerised techniques of visual field analysis. In: Anderson SMDaD, editor. *Automatic Perimetry in Glaucoma.* Florida: Grune and Stratton, 1985:147-160.

101. Flammer J. The concept of visual field indices. *Graefes Arch Clin Exp Ophthalmol* 1986;224(5):389-92.

102. Asman P, Heijl A. Glaucoma Hemifield Test. Automated visual field evaluation. *Arch Ophthalmol* 1992;110(6):812-9.

103. Quigley HA, Tielsch JM, Katz J, Sommer A. Rate of progression in open-angle glaucoma estimated from cross- sectional prevalence of visual field damage [see comments]. *Am J Ophthalmol* 1996;122(3):355-63.

104. Werner EB, Bishop KI, Koelle J, Douglas GR, LeBlanc RP, Mills RP, et al. A comparison of experienced clinical observers and statistical tests in detection of progressive visual field loss in glaucoma using automated perimetry. *Arch Ophthalmol* 1988;106(5):619-23.

105. Chauhan BC, Drance SM, LeBlanc RP, Lieberman MF, Mills RP, Werner EB. Technique for determining glaucomatous visual field progression by using animation graphics. *Am J Ophthalmol* 1994;118(4):485-91.

106. Smith SD, Katz J, Quigley HA. Analysis of progressive change in automated visual fields in glaucoma. *Invest Ophthalmol Vis Sci* 1996;37(7):1419-28.

107. Wegner A, Ugi I, Hofman A. A long-term visual field evaluation of glaucoma patients treated topically with timolol or carteolol. In: Mills-RP, editor. *Perimetry Update 1992/1993*. Amsterdam: Kugler & Ghedini, 1993:143-145.

108. Wu D, Schwartz B, Nagin P. Trend analyses of automated visual fields. *Doc Ophthalmol Proc Ser* 1987(49):175-189.

109. Katz J, Quigley HA, Sommer A. Detection of incident field loss using the glaucoma hemifield test. *Ophthalmology* 1996;103(4):657-63.

110. Chauhan BC, Drance SM, Douglas GR. The use of visual field indices in detecting changes in the visual field in glaucoma. *Invest Ophthalmol Vis Sci* 1990;31(3):512-20.

111. Birch MK, Wishart PK, O'Donnell N. Determining progressive field loss. In: Mills RP, Wall M, editors. *Perimetry Update 1994/1995*. Amsterdam: Kugler & Ghedini, 1995:31-36.

112. Wild JM, Hussey MK, Flanagan JG, Trope GE. Pointwise topographical and longitudinal modeling of the visual field in glaucoma. *Invest Ophthalmol Vis Sci* 1993;34(6):1907-16.

113. Hoskins HD, Jensvold N, Zaretsky M, Hetherington J. Rate of progression of discrete areas of the visual field. In: Heijl-A, editor. *Perimetry Update 1988/1989*. Amsterdam: Kugler & Ghedini, 1989:173-176.

114. O'Brien C, Schwartz B. The visual field in chronic open angle glaucoma: the rate of change in different regions of the field. *Eye* 1990;4(Pt 4):557-62.

115. Schulzer M. Errors in the diagnosis of visual field progression in normal-tension glaucoma. *Ophthalmology* 1994;101(9):1589-94.

116. The effectiveness of intraocular pressure reduction in the treatment of normal-tension glaucoma. Collaborative Normal-Tension Glaucoma Study Group [see comments]. *Am J Ophthalmol* 1998;126(4):498-505.

117. Katz J, Gilbert D, Quigley HA, Sommer A. Estimating progression of visual field loss in glaucoma. *Ophthalmology* 1997;104(6):1017-25.

118. Heijl A, Lindgren G, Lindgren A, et al. Extended empirical statistical package for evaluation of single and multiple fields in glaucoma: Statpac 2. In: Mills RP, Heijl A, editors. *Perimetry Update 1990/1991.* Amsterdam: Kugler & Ghedini, 1991:303-315.

119. Fitzke FW, Hitchings RA, Poinoosawmy D, McNaught AI, Crabb DP. Analysis of visual field progression in glaucoma. *Br J Ophthalmol* 1996(80):40-48.

120. McNaught AI, Crabb DP, Fitzke FW, Hitchings RA. Modelling series of visual fields to detect progression in normal tension glaucoma. *Graefes Arch Clin Exp Ophthalmol* 1995(233):750-755.

121. Noureddin BN, Poinoosawmy D, Fitzke FW, Hitchings RA. Regression analysis of visual field progression in low tension glaucoma. *Br J Ophthalmol* 1991;75(8):493-5.

122. Poinoosawmy D, Wu J, Fitzke FW, Hitchings RA. Discrimination between progression and non-progression visual field loss in low tension glaucoma using MDT. In: Mills-RP, editor. *Perimetry Update 1992/1993.* Amsterdam: Kugler & Ghedini, 1993:109-114.

123. Crabb DP, Hitchings RA, Fitzke FW. Detecting gradual and sudden sensitivity loss in series of visual fields. In: Wall M, Heijl A, editors. *Perimetry Update.* Amsterdam / New York: Kugler Publications, 1998/1999:in press.

124. McNaught AI, Crabb DP, Fitzke FW, Hitchings RA. Visual field progression: comparison of Humphrey Statpac 2 and pointwise linear

regression analysis. *Graefes Arch Clin Exp Ophthalmol* 1996(234):411-418.

125. Wild JM, Hutchings N, Hussey MK, Flanagan JG, Trope GE. Pointwise univariate linear regression of perimetric sensitivity against follow-up time in glaucoma. *: Ophthalmology* 1997;104(5):808-15.

126. Heijl A, Lindgren A, Lindgren G. Test-retest variability in glaucomatous visual fields. *Am J Ophthalmol* 1989;108(2):130-5.

127. Nouri-Mahdavi K, Brigatti L, Weitzman M, Caprioli J. Comparison of methods to detect visual field progression in glaucoma [published erratum appears in Ophthalmology 1998 Jan;105(1):7]. *Ophthalmology* 1997;104(8):1228-36.

128. Flammer J, Drance SM, Zulauf M. Differential light threshold. Short- and long-term fluctuation in patients with glaucoma, normal controls, and patients with suspected glaucoma. *Arch Ophthalmol* 1984;102(5):704-6.

129. Fitzke FW, Crabb DP, McNaught AI, Edgar DF, Hitchings RA. Image processing of computerised visual field data. *: Br J Ophthalmol* 1995;79(3):207-12.

130. Crabb DP, McNaught AI, Fitzke FW, Hitchings RA. Spatially enhanced modelling of sensitivity decay in low-tension glaucoma. In: Wall M, Heijl A, editors. *Perimetry Update.* Amsterdam / New York: Kugler Publications, 1994/1995:73-81.

131. Pacey IE, Wild JM, Cubbidge RP, Hancock S, Cunliffe IA. The between-subject, between-algorithm variability of normal sensitivity with the SITA Standard and SITA Fast threshold algorithms. *Invest. Ophthalmol. Vis. Sci.* 1998;39(4):S493.

132. Bengtsson B, Heijl A. Sensitivity to glaucomatous visual field loss in full threshold, SITA Standard and SITA Fast tests. In: Wall M, Heijl A, editors. *Perimetry Update.* Amsterdam / New York: Kugler Publications, 1998/1999:in press.

133. Goldberg I. The Swedish Interactive Thresholding Algorithm (SITA) in patients with prior experience with the full threshold Humphrey Field Analyzer. In: Wall M, Heijl A, editors. *Perimetry Update.* Amsterdam / New York: Kugler Publications, 1998/1999:in press.

134. Werner EB, Adelson A, Krupin T. Effect of patient experience on the results of automated perimetry in clinically stable glaucoma patients. *Ophthalmology* 1988;95(6):764-7.

135. Mikelberg FS, S.M. D, Schulzer M, Wijsman K. The effect of miosis on visual field indices. *Doc Ophthal Proc Ser* 1987;49:645-649.

136. Lindenmuth KA, Skuta GL, Rabbani R, Musch DC. Effects of pupillary constriction on automated perimetry in normal eyes. *Ophthalmology* 1989;96(9):1298-301.

137. Lindenmuth KA, Skuta GL, Rabbani R, Musch DC, Bergstrom TJ. Effects of pupillary dilation on automated perimetry in normal patients. *Ophthalmology* 1990;97(3):367-70.

138. Herse PR. Factors influencing normal perimetric thresholds obtained using the Humphrey Field Analyzer. *Invest Ophthalmol Vis Sci* 1992;33(3):611-7.

139. Brenton RS, Phelps CD. The normal visual field on the Humphrey field analyzer. *Ophthalmologica* 1986;193(1-2):56-74.

140. Weinreb RN, Perlman JP. The effect of refractive correction on automated perimetric thresholds. *Am J Ophthalmol* 1986;101(6):706-9.

141. Heuer DK, Anderson DR, Feuer WJ, Gressel MG. The influence of refraction accuracy on automated perimetric threshold measurements. *Ophthalmology* 1987;94(12):1550-3.

142. Brenton RS, Phelps CD, Rojas P, Woolson RF. Interocular differences of the visual field in normal subjects. *Invest Ophthalmol Vis Sci* 1986;27(5):799-805.

143. Kosmin AS, Wishart PK, Birch MK. Apparent glaucomatous visual field defects caused by dermatochalasis. *Eye* 1997;11((Pt 5)):682-6.

144. Katz J, Sommer A. Reliability indexes of automated perimetric tests. *Arch Ophthalmol* 1988;106(9):1252-4.

145. Palmberg P. Epidemiology of POAG and rationale for therapy. *Glaucoma Abstracts* 1969;6:10-23.

146. Keltner JL, Johnson CA, Spurr JO, Kass MA, Gordon MO. Confirmation of visual field abnormalities in the Ocular Hypertension Treatment Study (OHTS). *Invest Ophthalmol Vis Sci* 1998;39(4):S493.

147. Bhandari A, Crabb DP, Poinoosawmy D, Fitzke FW, Hitchings RA, Noureddin BN. Effect of surgery on visual field progression in normal-tension glaucoma. *Ophthalmology* 1997;104(7):1131-7.

148. Fontana L, Viswanathan AC, Poinooswamy D, Hitchings RA, Scullica L. Surgery for normal tension glaucoma. Target intraocular pressure and visual field progression. *Acta Ophthalmol Scand Suppl* 1997;224:43-4.

149. Hitchings RA, Wu J, Poinoosawmy D, McNaught A. Surgery for normal tension glaucoma. *: Br J Ophthalmol* 1995;79(5):402-6.

150. Koseki N, Araie M, Shirato S, Yamamoto S. Effect of trabeculectomy on visual field performance in central 30 degrees field in progressive normal-tension glaucoma. *: Ophthalmology* 1997;104(2):197-201.

151. Burke J, Schwartz M. Preclinical evaluation of brimonidine. *Surv Ophthalmol* 1996;41 Suppl 1:S9-18.

152. Weber J, Diestelhorst M. Perimetric follow-up in glaucoma with a reduced set of test points. *Ger J Ophthalmol* 1992;1(6):409-14.

153. Boeglin RJ, Caprioli J, Zulauf M. Long-term fluctuation of the visual field in glaucoma. *Am J Ophthalmol* 1992;113(4):396-400.

154. Werner EB, Krupin T, Adelson A, Feitl ME. Effect of patient experience on the results of automated perimetry in glaucoma suspect patients. *Ophthalmology* 1990;97(1):44-8.

155. Munton G. Chapter 11. Vision. In: Taylor JF, editor. *Medical Aspects of Fitness to Drive*. Fifth ed. London: The Medical Commission on Accident Prevention, 1995:118-132.

156. Crabb DP, Viswanathan AC, McNaught AI, Poinoosawmy D, Fitzke FW, Hitchings RA. Simulating binocular field status in glaucoma. *Br J Ophthalmol* 1998;82(11):1236-1241.

157. Altman DG. *Practical statistics for medical research.* 1st ed. London: Chapman and Hall, 1991.

158. Fleiss JL. Statistical methods for rates and proportions. New York: John Wiley, 1981:212-225.

159. Landis JR, Koch GG. The measurement of observer agreement for categorical data. *Biometrics* 1977;33:159-174.

160. Mikelberg FS, Drance SM. The mode of progression of visual field defects in glaucoma. *Am J Ophthalmol* 1984;98(4):443-5.

161. Viswanathan AC, Hitchings RA, Fitzke FW. Spatial Filtering of Glaucomatous Visual Fields using PROGRESSOR for Windows. In: Wall M, Heijl A, editors. *Perimetry Update.* Amsterdam / New York: Kugler Publications, 1996/1997:311-319.

162. Viswanathan AC, Fitzke FW, Hitchings RA. Early Detection of Visual Field Progression in Glaucoma: A Comparison of PROGRESSOR and Statpac 2. *Br J Ophthalmol* 1997;81(12):1037-1042.

163. Esterman B. Grid for scoring visual fields. II. Perimeter. *Arch Ophthalmol* 1968;79(4):400-6.

164. Esterman B. Functional scoring of the binocular field. *Ophthalmology* 1982;89(11):1226-34.

165. Mills RP, Drance SM. Esterman disability rating in severe glaucoma. *Ophthalmology* 1986;93(3):371-8.

166. Bland JM, Altman DG. Multiple significance tests: the Bonferroni method. *British Medical Journal* 1995;310(6973):170.

167. Choy ES, Mills RP, Drance SM. Automated Esterman testing of disability in glaucoma. In: Greve E, Heijl A, editors. *Seventh International Visual Field Symposium.* Amsterdam: Junk Publishers, 1986:527-535.

168. Parrish RK, 2nd. Visual impairment, visual functioning, and quality of life assessments in patients with glaucoma. *Trans Am Ophthalmol Soc* 1996;94:919-1028.

169. Parrish RK, 2nd, Gedde SJ, Scott IU, Feuer WJ, Schiffman JC, Mangione CM, et al. Visual function and quality of life among patients with glaucoma. *Arch Ophthalmol* 1997;115(11):1447-55.

170. Flammer J, Drance SM, Zulauf M. Differential light threshold. Short- and long-term fluctuation in patients with glaucoma, normal controls, and patients with suspected glaucoma. *: Arch Ophthalmol* 1984;102(5):704-6.

171. Phillips D. *Image Processing in C.* 1st ed. Kansas: R&D Publications Inc., 1994.

172. Gonzales RC, Wintz P. *Digital image processing.* 2nd ed. Massachusetts: Addison-Wesley, 1987.

173. Crabb DP, Fitzke FW, Hitchings RA. A profile of the spatial dependence of pointwise sensitivity across the glaucomatous visual field. In: Wall M, Heijl A, editors. *Perimetry Update.* Amsterdam / New York: Kugler Publications, 1996/1997:302-310.

APPENDIX A. PROGRESSOR SOURCE CODE

This appendix contains the source code for the version of PROGRESSOR written to conduct the research described in Chapters 4, 5 and 6 of this thesis. It is designed to be compiled as a 16-bit Windows™ application based on the Microsoft Foundation Classes within the Visual C++ version 1.52 (© 1993 Microsoft Corporation, Redmond, Washington, USA) programming environment. Only the code created specifically for PROGRESSOR is included: files common to all Microsoft Foundation Class applications such as stdafx.h and stdafx.cpp are not listed. A demonstration version of the program with context-sensitive Help system is contained in the CDROM which accompanies this thesis.

```
// address.cpp : implementation file
//

#include "stdafx.h"
#include "prowin01.h"
#include "address.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
////////////
// CAddressDialog dialog


CAddressDialog::CAddressDialog(CWnd* pParent /*=NULL*/)
      : CDialog(CAddressDialog::IDD, pParent)
{
      //{{AFX_DATA_INIT(CAddressDialog)
      m_Address = "";
      //}}AFX_DATA_INIT
}

void CAddressDialog::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CAddressDialog)
      DDX_Text(pDX, IDC_STATIC_ADDRESS, m_Address);
      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAddressDialog, CDialog)
      //{{AFX_MSG_MAP(CAddressDialog)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////
////////////
// CAddressDialog message handlers
```

```
BOOL CAddressDialog::OnInitDialog()
{
	int x = ::GetSystemMetrics(SM_CXSCREEN);
	int y = ::GetSystemMetrics(SM_CYSCREEN);

	CRect dlgRect;
	GetWindowRect(&dlgRect);
	CDialog::OnInitDialog();

	SetWindowPos(&CWnd::wndTopMost,x/2-dlgRect.Width()/2,y/2-
dlgRect.Height()/2,0,0,SWP_NOSIZE );

	return TRUE;  // return TRUE  unless you set the focus to a
control
}
```

```
// address.h : header file
//

/////////////////////////////////////////////////////////////////////
///////////
// CAddressDialog dialog

class CAddressDialog : public CDialog
{
// Construction
public:
      CAddressDialog(CWnd* pParent = NULL);     // standard
constructor

// Dialog Data
      //{{AFX_DATA(CAddressDialog)
      enum { IDD = IDD_ADDRESS_DIALOG };
      CString     m_Address;
      //}}AFX_DATA

// Implementation
protected:
      virtual void DoDataExchange(CDataExchange* pDX);      //
DDX/DDV support

      // Generated message map functions
      //{{AFX_MSG(CAddressDialog)
      virtual BOOL OnInitDialog();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
```

```
// datedlg.cpp : implementation file
//

#include "stdafx.h"
#include "prowin01.h"
#include "datedlg.h"
#include "prowidoc.h"
#include "progind.h"
#include "extrap.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////
///////////
// CDateDialog dialog


CDateDialog::CDateDialog(CWnd* pParent /*=NULL*/)
      : CDialog(CDateDialog::IDD, pParent)
{
      //{{AFX_DATA_INIT(CDateDialog)
            // NOTE: the ClassWizard will add member
initialization here
      //}}AFX_DATA_INIT
}

void CDateDialog::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CDateDialog)
            // NOTE: the ClassWizard will add DDX and DDV calls
here
      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CDateDialog, CDialog)
      //{{AFX_MSG_MAP(CDateDialog)
      ON_LBN_SELCHANGE(IDC_RIGHT_LIST, OnSelchangeRightList)
      ON_LBN_SELCHANGE(IDC_LEFT_LIST, OnSelchangeLeftList)
      ON_BN_CLICKED(IDC_REANALYZE, OnReanalyze)
      ON_BN_CLICKED(IDC_RESTORE, OnRestore)
```

```
      ON_WM_PAINT()
      ON_BN_CLICKED(IDC_EXTRAPOLATE, OnExtrapolate)
      ON_BN_CLICKED(IDC_ANIMATE, OnAnimate)
      ON_WM_TIMER()
      ON_BN_CLICKED(IDC_HELP, OnClickedHelp)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BOOL CDateDialog::Create()
{     return CDialog::Create( CDateDialog::IDD);
}

////////////////////////////////////////////////////////////////
///////////
// CDateDialog message handlers

BOOL CDateDialog::OnInitDialog()
{   char buffer[5];
      CString right_no;
      CString left_no;
      CButton* p_ReanalyzeButton =
(CButton*)GetDlgItem(IDC_REANALYZE);
      CButton* p_RestoreButton =
(CButton*)GetDlgItem(IDC_RESTORE);
      CWnd* p_RightReanalyzedIcon =
(CWnd*)GetDlgItem(IDC_RIGHT_REANALYZED);
      CWnd* p_LeftReanalyzedIcon =
(CWnd*)GetDlgItem(IDC_LEFT_REANALYZED);
      int i;
      int right_size = m_pDoc -> m_rightorderArray.GetSize();
      int left_size = m_pDoc -> m_leftorderArray.GetSize();

      if (!m_pDoc ->
m_bDateDialogDisplayed)m_AnimateButton.AutoLoad(IDC_ANIMATE,this);

      m_nOldRightFirstDate = m_nOldLeftFirstDate = MAX_RESTD_DATES
+ 1; //Init member variables//
      m_neworderArray.SetSize(0,20);
//Init member variables//

      p_ReanalyzeButton -> EnableWindow(FALSE);
      if ( m_pDoc -> m_rightReanalyzed || m_pDoc ->
m_leftReanalyzed )p_RestoreButton -> EnableWindow(TRUE);
            else p_RestoreButton -> EnableWindow(FALSE);
```

```
      if ( m_pDoc -> m_rightReanalyzed ) p_RightReanalyzedIcon ->
ShowWindow(SW_SHOWNORMAL);
            else p_RightReanalyzedIcon -> ShowWindow(SW_HIDE);
      if ( m_pDoc -> m_leftReanalyzed ) p_LeftReanalyzedIcon ->
ShowWindow(SW_SHOWNORMAL);
            else p_LeftReanalyzedIcon -> ShowWindow(SW_HIDE);

      _itoa(right_size,buffer,10);
      right_no = buffer;
      right_no += right_size == 1 ? " test" : " tests";
      CDateDialog::SetDlgItemText(IDC_RIGHT_TESTS,right_no);

      _itoa(left_size,buffer,10);
      left_no = buffer;
      left_no += left_size == 1 ? " test" : " tests";
      CDateDialog::SetDlgItemText(IDC_LEFT_TESTS,left_no);


      CListBox* p_rightLB = (CListBox*)
GetDlgItem(IDC_RIGHT_LIST);
      CListBox* p_leftLB = (CListBox*) GetDlgItem(IDC_LEFT_LIST);

      p_rightLB -> SetTabStops();
      p_leftLB -> SetTabStops();
      CDateDialog::SetWindowText("DATES: " + m_pDoc -> m_name);
      BeginWaitCursor();

      p_rightLB -> ResetContent();
      p_leftLB -> ResetContent();

      for(  i = 0; i < right_size; i++ )p_rightLB ->
AddString(m_pDoc -> m_righttimestringArray.GetAt(m_pDoc ->
m_rightorderArray.GetAt(i)));
      for(  i = 0; i < left_size; i++ )p_leftLB ->
AddString(m_pDoc -> m_lefttimestringArray.GetAt(m_pDoc ->
m_leftorderArray.GetAt(i)));

      if (right_size)
            {p_rightLB -> SetSel(right_size-1);
            OnSelchangeRightList();
      }
      if (left_size)
            {p_leftLB -> SetSel(left_size-1);
```

```
            OnSelchangeLeftList();
      }


      EndWaitCursor();
      return CDialog::OnInitDialog();  // return TRUE  unless you
set the focus to a control
}

void CDateDialog::OnCancel()
{
      m_pDoc -> m_bDateDialogDisplayed = FALSE;
      m_pDoc -> UpdateAllViews(NULL,3L,NULL);
      CWnd::DestroyWindow();


      //CDialog::OnCancel();
}

void CDateDialog::OnSelchangeRightList()
{   CListBox* p_rightLB = (CListBox*) GetDlgItem(IDC_RIGHT_LIST);
      CButton* p_ReanalyzeButton =
(CButton*)GetDlgItem(IDC_REANALYZE);
      m_nRightSelCount = p_rightLB -> GetSelItems(MAX_RESTD_DATES,
(LPINT)m_nRightRestrictedDates);
      TRACE("m_nRightSelCount = %d\nm_nRightRestrictedDates[0] =
%d\nm_pDoc -> m_rightorderArray.GetSize() = %d\n
",m_nRightSelCount,m_nRightRestrictedDates[0],m_pDoc ->
m_rightorderArray.GetSize());
      if (m_nRightSelCount == 1 && m_nRightRestrictedDates[0] !=
m_nOldRightFirstDate)
            {m_pDoc -> m_nRightCount = m_nRightRestrictedDates[0]
+ 1; //for progression indices dialog
            if (m_pDoc -> m_bProgIndDlgDisplayed)
                  m_pDoc -> m_pProgIndDlg -> OnInitDialog();

            m_pDoc -> UpdateAllViews(NULL,1L,NULL);
            m_nOldRightFirstDate = m_nRightRestrictedDates[0];
      }
      if (m_nRightSelCount > 2 && m_nRightSelCount < m_pDoc ->
m_rightorderArray.GetSize())
            p_ReanalyzeButton -> EnableWindow(TRUE);
            else p_ReanalyzeButton -> EnableWindow(FALSE);
```

```
}

void CDateDialog::OnSelchangeLeftList()
{   CListBox* p_leftLB = (CListBox*) GetDlgItem(IDC_LEFT_LIST);
      CButton* p_ReanalyzeButton =
(CButton*)GetDlgItem(IDC_REANALYZE);
      m_nLeftSelCount = p_leftLB -> GetSelItems(MAX_RESTD_DATES,
(LPINT)m_nLeftRestrictedDates);
      TRACE("m_nLeftSelCount = %d\nm_nLeftRestrictedDates[0] =
%d\nm_pDoc -> m_leftorderArray.GetSize() = %d\n
",m_nLeftSelCount,m_nLeftRestrictedDates[0],m_pDoc ->
m_leftorderArray.GetSize());
      if (m_nLeftSelCount == 1 && m_nLeftRestrictedDates[0] !=
m_nOldLeftFirstDate)
            {m_pDoc -> m_nLeftCount = m_nLeftRestrictedDates[0] +
1;  //for progression indices dialog
            if (m_pDoc -> m_bProgIndDlgDisplayed)
                  m_pDoc -> m_pProgIndDlg -> OnInitDialog();

            m_pDoc -> UpdateAllViews(NULL,2L,NULL);
            m_nOldLeftFirstDate = m_nLeftRestrictedDates[0];
      }
      if (m_nLeftSelCount > 2 && m_nLeftSelCount < m_pDoc ->
m_leftorderArray.GetSize())
            p_ReanalyzeButton -> EnableWindow(TRUE);
            else p_ReanalyzeButton -> EnableWindow(FALSE);

}

void CDateDialog::OnReanalyze()
{
      int i;
      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);

      BeginWaitCursor();
    pStatus->SetPaneText(0,"Reanalyzing ...",TRUE);
      pStatus->UpdateWindow();
      if ( m_nRightSelCount > 2 && m_nRightSelCount < m_pDoc ->
m_rightorderArray.GetSize() )
            {m_neworderArray.RemoveAll();
            for ( i = 0; i < m_nRightSelCount; i++)
                  m_neworderArray.Add(m_pDoc ->
m_rightorderArray[m_nRightRestrictedDates[i]]);
```

```
            m_pDoc -> m_rightorderArray.RemoveAll();
            for ( i = 0; i < m_nRightSelCount; i++)
                  m_pDoc ->
m_rightorderArray.Add(m_neworderArray[i]);
            m_pDoc -> m_nRightCount = m_nRightSelCount; //for
progression indices dialog
            m_pDoc -> m_rightReanalyzed = 1;
            //m_pDoc -> SetModifiedFlag();

      }

            if ( m_nLeftSelCount > 2 && m_nLeftSelCount < m_pDoc -
> m_leftorderArray.GetSize() )
            {m_neworderArray.RemoveAll();
            for ( i = 0; i < m_nLeftSelCount; i++)
                  m_neworderArray.Add(m_pDoc ->
m_leftorderArray[m_nLeftRestrictedDates[i]]);
            m_pDoc -> m_leftorderArray.RemoveAll();
            for ( i = 0; i < m_nLeftSelCount; i++)
                  m_pDoc ->
m_leftorderArray.Add(m_neworderArray[i]);
            m_pDoc -> m_nLeftCount = m_nLeftSelCount;  //for
progression indices dialog
            m_pDoc -> m_leftReanalyzed = 1;
            //m_pDoc -> SetModifiedFlag();

      }

      m_pDoc -> Regress();


      EndWaitCursor();
      OnInitDialog();

    pStatus->SetPaneText(0,"Reanalysis complete. For Help, press
F1",TRUE);
      pStatus->UpdateWindow();


}

void CDateDialog::OnRestore()
{
```

```
      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);
      BeginWaitCursor();
    pStatus->SetPaneText(0,"Restoring full analysis ...",TRUE);
      pStatus->UpdateWindow();
      m_pDoc -> m_rightorderArray.RemoveAll();
      m_pDoc -> m_leftorderArray.RemoveAll();
      m_pDoc -> DoTimeOrder();
      m_pDoc -> Regress();
      m_pDoc -> UpdateAllViews(NULL,3L,NULL);
      m_pDoc -> m_rightReanalyzed = 0;
      m_pDoc -> m_leftReanalyzed = 0;
      //m_pDoc -> SetModifiedFlag();
      EndWaitCursor();
      OnInitDialog();
    pStatus->SetPaneText(0,"Full analysis restored. For Help,
press F1",TRUE);
      pStatus->UpdateWindow();
}

void CDateDialog::OnPaint()
{
      CPaintDC dc(this); // device context for painting

      if (this -> IsIconic())
            {dc.SetMapMode(MM_TEXT);
            dc.DrawIcon(2,2,(HICON)AfxGetApp()-
>LoadIcon(IDI_DATES));
      }

      // Do not call CDialog::OnPaint() for painting messages
}

void CDateDialog::OnExtrapolate()
{
      CListBox* p_leftLB = (CListBox*) GetDlgItem(IDC_LEFT_LIST);
      CListBox* p_rightLB = (CListBox*)
GetDlgItem(IDC_RIGHT_LIST);
      CExtrapDialog dlg;
      CTime extrapTime;
      CTimeSpan extrapTimeSpan;
      CString extrapTimeString;
      CTime refTime(1980,1,1,0,0,0);
      DWORD extrap_seconds;
```

```
      DWORD dwRightLatestSeconds = (DWORD)m_pDoc ->
m_rightsecondArray[m_pDoc -> m_rightorderArray
            [m_pDoc -> m_rightorderArray.GetUpperBound()]];
      DWORD dwLeftLatestSeconds = (DWORD)m_pDoc ->
m_leftsecondArray[m_pDoc -> m_leftorderArray
            [m_pDoc -> m_leftorderArray.GetUpperBound()]];

      //sets up extrap dialog with the latest date from either eye
as default
      if ( dwRightLatestSeconds > dwLeftLatestSeconds )
      p_rightLB -> GetText(p_rightLB -> GetCount() -
1,extrapTimeString);
      else p_leftLB -> GetText(p_leftLB -> GetCount() -
1,extrapTimeString);
      // default is both eyes
      dlg.m_nEye = 2;
      sscanf(extrapTimeString,"%*s %d-%d-%d" ,&dlg.m_nDay,
&dlg.m_nMonth, &dlg.m_nYear);
      if (dlg.m_nYear > 38) dlg.m_nYear +=
1900;////////////////////////////////////tied to CTime
restrictions
      else dlg.m_nYear += 2000;

      if (dlg.DoModal()==IDOK)
            {extrapTime =
CTime(dlg.m_nYear,dlg.m_nMonth,dlg.m_nDay,12,0,0);
            extrapTimeSpan = extrapTime - refTime;
            extrap_seconds = extrapTimeSpan.GetTotalSeconds();
            extrapTimeString = extrapTime.FormatGmt("%a\t%d-%m-
%y");

            if(dlg.m_nEye)          // right or both eyes selected
****m_rightbasetime IS RESET BY OnRestore****
                  {if (extrap_seconds < m_pDoc -> m_rightbasetime
||
                        extrap_seconds - m_pDoc -> m_rightbasetime
> dwRightLatestSeconds)AfxMessageBox("Outside");
                        else AfxMessageBox("Inside");
            }
            //if(dlg.m_nEye != 1)         // left or both eyes
selected

      }
```

```
}

void CDateDialog::OnAnimate()
{
      CListBox* p_leftLB = (CListBox*) GetDlgItem(IDC_LEFT_LIST);
      CListBox* p_rightLB = (CListBox*)
GetDlgItem(IDC_RIGHT_LIST);
      int right_size = m_pDoc -> m_rightorderArray.GetSize();
      int left_size = m_pDoc -> m_leftorderArray.GetSize();

      m_nRightAnimateSel = m_nLeftAnimateSel = 0;

      if (!right_size && !left_size) return;

      DWORD rstart_time, lstart_time, rend_time, lend_time;

      if (right_size)
            {rstart_time = m_pDoc -> m_rightsecondArray[m_pDoc ->
m_rightorderArray[0]];
            rend_time = m_pDoc -> m_rightsecondArray[m_pDoc ->
m_rightorderArray[right_size - 1]];
      }
      else
            {rstart_time = 0xFFFFFFFF;
            rend_time = 0;
      }

      if (left_size)
            {lstart_time = m_pDoc -> m_leftsecondArray[m_pDoc ->
m_leftorderArray[0]];
            lend_time = m_pDoc -> m_leftsecondArray[m_pDoc ->
m_leftorderArray[left_size - 1]];
      }
      else
            {lstart_time = 0xFFFFFFFF;
            lend_time = 0;
      }

      m_dwAnimateStartTime = rstart_time < lstart_time ?
rstart_time : lstart_time;
      m_dwAnimateEndTime = rend_time > lend_time ? rend_time :
lend_time;
```

```
      if (SetTimer(ID_ANIMATION_TIMER,1000,NULL) !=
ID_ANIMATION_TIMER)
            AfxMessageBox("Sorry, your system has no free
timers.\nAnimation is not possible at the moment.");
      else m_bStartingAnimation = TRUE;

      return;
}

void CDateDialog::OnTimer(UINT nIDEvent)
{
      CListBox* p_leftLB = (CListBox*) GetDlgItem(IDC_LEFT_LIST);
      CListBox* p_rightLB = (CListBox*)
GetDlgItem(IDC_RIGHT_LIST);
      int right_size = m_pDoc -> m_rightorderArray.GetSize();
      int left_size = m_pDoc -> m_leftorderArray.GetSize();



      if (nIDEvent == ID_ANIMATION_TIMER)
            {if (m_bStartingAnimation)
                  {p_rightLB -> SetSel(-1,FALSE);
                  p_rightLB -> SetSel(0);
                  OnSelchangeRightList();
                  p_leftLB -> SetSel(-1,FALSE);
                  p_leftLB -> SetSel(0);
                  OnSelchangeLeftList();
                  m_bStartingAnimation = FALSE;
            }
        else
                  {if (m_nRightAnimateSel < right_size &&
m_dwAnimateStartTime >=
                  m_pDoc -> m_rightsecondArray[m_pDoc ->
m_rightorderArray[m_nRightAnimateSel]])
                        {p_rightLB -> SetSel(-1,FALSE);
                        p_rightLB -> SetSel(m_nRightAnimateSel++);
                        OnSelchangeRightList();
                  }
                  if (m_nLeftAnimateSel < left_size &&
m_dwAnimateStartTime >=
                  m_pDoc -> m_leftsecondArray[m_pDoc ->
m_leftorderArray[m_nLeftAnimateSel]])
                        {p_leftLB -> SetSel(-1,FALSE);
                        p_leftLB -> SetSel(m_nLeftAnimateSel++);
```

```
                        OnSelchangeLeftList();
                }
        }

        if (m_dwAnimateStartTime >= m_dwAnimateEndTime)
                        {p_rightLB -> SetSel(-1,FALSE);
                        p_rightLB -> SetSel(right_size - 1);
                        OnSelchangeRightList();
                        p_leftLB -> SetSel(-1,FALSE);
                        p_leftLB -> SetSel(left_size - 1);
                        OnSelchangeLeftList();
                        KillTimer(ID_ANIMATION_TIMER);
        }
        else
                {while (m_dwAnimateStartTime <=
(m_nRightAnimateSel < right_size ? m_pDoc -> m_rightsecondArray
                [m_pDoc -> m_rightorderArray[
m_nRightAnimateSel]] : 0xFFFFFFFF)
                &&
                m_dwAnimateStartTime <= (m_nLeftAnimateSel <
left_size ? m_pDoc -> m_leftsecondArray
                [m_pDoc -> m_leftorderArray[ m_nLeftAnimateSel]]
: 0xFFFFFFFF))
                m_dwAnimateStartTime +=  3600;
        }
    }
    CDialog::OnTimer(nIDEvent);
}

void CDateDialog::OnClickedHelp()
{
    OnHelp();
}
```

```
// datedlg.h : header file
//
#define MAX_RESTD_DATES 100
#define ID_ANIMATION_TIMER 1
/////////////////////////////////////////////////////////////////////
////////////
// CDateDialog dialog

class CDateDialog : public CDialog
{  friend class CProwin01Doc;
private:
      CBitmapButton m_AnimateButton;
// Construction
public:
      CDateDialog(CWnd* pParent = NULL);  // standard constructor

// Dialog Data
      //{{AFX_DATA(CDateDialog)
      enum { IDD = IDD_DATE_DIALOG };
            // NOTE: the ClassWizard will add data members here
      //}}AFX_DATA

// Data members
      CProwin01Doc* m_pDoc;
      CByteArray m_neworderArray;
      int m_nOldRightFirstDate;
      int m_nOldLeftFirstDate;
      int m_nRightRestrictedDates[MAX_RESTD_DATES];
      int m_nLeftRestrictedDates[MAX_RESTD_DATES];
      int m_nRightSelCount;
      int m_nLeftSelCount;

      int m_nRightAnimateSel;
      int m_nLeftAnimateSel;
      DWORD m_dwAnimateStartTime;
      DWORD m_dwAnimateEndTime;
      BOOL m_bStartingAnimation;

// Implementation
      BOOL Create();
protected:
      virtual void DoDataExchange(CDataExchange* pDX);      //
DDX/DDV support
```

```
    // Generated message map functions
    //{{AFX_MSG(CDateDialog)
    virtual BOOL OnInitDialog();
    virtual void OnCancel();
    afx_msg void OnSelchangeRightList();
    afx_msg void OnSelchangeLeftList();
    afx_msg void OnReanalyze();
    afx_msg void OnRestore();
    afx_msg void OnPaint();
    afx_msg void OnExtrapolate();
    afx_msg void OnAnimate();
    afx_msg void OnTimer(UINT nIDEvent);
    afx_msg void OnClickedHelp();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

```
// extrap.cpp : implementation file
//

#include "stdafx.h"
#include "prowin01.h"
#include "extrap.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
///////////
// CExtrapDialog dialog


CExtrapDialog::CExtrapDialog(CWnd* pParent /*=NULL*/)
      : CDialog(CExtrapDialog::IDD, pParent)
{
      //{{AFX_DATA_INIT(CExtrapDialog)
      m_nDay = 1;
      m_nMonth = 1;
      m_nYear = 2000;
      m_nEye = -1;
      //}}AFX_DATA_INIT
}

void CExtrapDialog::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CExtrapDialog)
      DDX_Text(pDX, IDC_EDIT_DAY, m_nDay);
      DDV_MinMaxInt(pDX, m_nDay, 1, 31);
      DDX_Text(pDX, IDC_EDIT_MONTH, m_nMonth);
      DDV_MinMaxInt(pDX, m_nMonth, 1, 12);
      DDX_Text(pDX, IDC_EDIT_YEAR, m_nYear);
      DDV_MinMaxInt(pDX, m_nYear, 1980, 2038);
      DDX_Radio(pDX, IDC_RADIO_LEFT, m_nEye);
      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CExtrapDialog, CDialog)
      //{{AFX_MSG_MAP(CExtrapDialog)
```

```
	ON_BN_CLICKED(IDOK, OnClickedOK)
	ON_WM_HSCROLL()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()


////////////////////////////////////////////////////////////////
///////////
// CExtrapDialog message handlers

BOOL CExtrapDialog::OnInitDialog()
{
	int x = ::GetSystemMetrics(SM_CXSCREEN);
	int y = ::GetSystemMetrics(SM_CYSCREEN);

	CRect dlgRect;
	GetWindowRect(&dlgRect);
	CDialog::OnInitDialog();
	CScrollBar* pScrollbar =
(CScrollBar*)GetDlgItem(IDC_EXTRAP_SCROLLBAR);

	SetWindowPos(&CWnd::wndTopMost,x/2-dlgRect.Width()/2,y/2-
dlgRect.Height()/2,0,0,SWP_NOSIZE );
	pScrollbar -> SetScrollRange(1980, 2038);
	pScrollbar -> SetScrollPos(m_nYear);

	return TRUE;  // return TRUE  unless you set the focus to a
control
}

void CExtrapDialog::OnClickedOK()
{
	int month, day;
	char buffer[5];
	GetDlgItemText(IDC_EDIT_DAY,buffer,5);
	day = atoi(buffer);
	GetDlgItemText(IDC_EDIT_MONTH,buffer,5);
	month = atoi(buffer);
	if (day>30)
		if (month==9 || month==4 || month==6 || month==11)
			{AfxMessageBox("Invalid date: please try
again");
			return;
		}
```

```
      if (day>28)
            if (month==2)
                  {AfxMessageBox("Invalid date: please try
again");
                  return;
            }

      CDialog::OnOK();

}

void CExtrapDialog::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar*
pScrollBar)
{
      char buffer[5];
      int nPrevPos = pScrollBar -> GetScrollPos();
      GetDlgItemText(IDC_EDIT_MONTH,buffer,5);
      int month = atoi(buffer);

      switch(nSBCode)
            {case SB_THUMBPOSITION:
                  pScrollBar -> SetScrollPos(nPos);

      SetDlgItemText(IDC_EDIT_YEAR,_itoa(nPos,buffer,10));
                  break;
            case SB_THUMBTRACK:
                  pScrollBar -> SetScrollPos(nPos);

      SetDlgItemText(IDC_EDIT_YEAR,_itoa(nPos,buffer,10));
                  break;
            case SB_LEFT:
                  SetDlgItemText(IDC_EDIT_DAY,"1");
                  SetDlgItemText(IDC_EDIT_MONTH,"1");
                  SetDlgItemText(IDC_EDIT_YEAR,"1980");
                  break;
            case SB_RIGHT:
                  SetDlgItemText(IDC_EDIT_DAY,"31");
                  SetDlgItemText(IDC_EDIT_MONTH,"12");
                  SetDlgItemText(IDC_EDIT_YEAR,"2038");
                  break;
            case SB_PAGERIGHT:
                  pScrollBar -> SetScrollPos(++nPrevPos);

      SetDlgItemText(IDC_EDIT_YEAR,_itoa(nPrevPos,buffer,10));
```

```
                break;
          case SB_PAGELEFT:
                pScrollBar -> SetScrollPos(--nPrevPos);

      SetDlgItemText(IDC_EDIT_YEAR,_itoa(nPrevPos,buffer,10));
                break;
          case SB_LINERIGHT:
                if (++month == 13)
                      {month = 1;
                      if (nPrevPos != 2038)
                            {pScrollBar ->
SetScrollPos(++nPrevPos);

      SetDlgItemText(IDC_EDIT_YEAR,_itoa(nPrevPos,buffer,10));
                      }
                }

      SetDlgItemText(IDC_EDIT_MONTH,_itoa(month,buffer,10));
                break;
          case SB_LINELEFT:
                if (--month == 0)
                      {month = 12;
                      if (nPrevPos != 1980)
                            {pScrollBar -> SetScrollPos(--
nPrevPos);

      SetDlgItemText(IDC_EDIT_YEAR,_itoa(nPrevPos,buffer,10));
                      }
                }

      SetDlgItemText(IDC_EDIT_MONTH,_itoa(month,buffer,10));
                break;
      }

      CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
}
```

```
// extrap.h : header file
//

/////////////////////////////////////////////////////////////////////////
///////////
// CExtrapDialog dialog

class CExtrapDialog : public CDialog
{
// Construction
public:
      CExtrapDialog(CWnd* pParent = NULL);      // standard
constructor

// Dialog Data
      //{{AFX_DATA(CExtrapDialog)
      enum { IDD = IDD_EXTRAP_DIALOG };
      int         m_nDay;
      int         m_nMonth;
      int         m_nYear;
      int         m_nEye;
      //}}AFX_DATA

// Implementation
protected:
      virtual void DoDataExchange(CDataExchange* pDX);      //
DDX/DDV support

      // Generated message map functions
      //{{AFX_MSG(CExtrapDialog)
      virtual BOOL OnInitDialog();
      afx_msg void OnClickedOK();
      afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar*
pScrollBar);
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
```

```
// getptdlg.cpp : implementation file
//

#include "stdafx.h"
#include "prowin01.h"
#include "getptdlg.h"
#include "prowidoc.h"
#include "mainfrm.h"
#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
///////////
// CGetPtDialog dialog


CGetPtDialog::CGetPtDialog(CWnd* pParent /*=NULL*/)     // was
CGetPtDialog::CGetPtDialog(CWnd* pParent /*=NULL*/)
      : CDialog(CGetPtDialog::IDD, pParent)
{
      //{{AFX_DATA_INIT(CGetPtDialog)
      //}}AFX_DATA_INIT
}

void CGetPtDialog::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CGetPtDialog)
      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CGetPtDialog, CDialog)
      //{{AFX_MSG_MAP(CGetPtDialog)
      ON_BN_CLICKED(IDOK, OnClickedOK)
      ON_LBN_SELCHANGE(IDC_PATIENT_LIST, OnSelchangePatientList)
      ON_WM_PAINT()
      ON_BN_CLICKED(IDC_HELP, OnClickedHelp)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BOOL CGetPtDialog::Create()
{     return CDialog::Create( CGetPtDialog::IDD);
```

```
}


//////////////////////////////////////////////////////////////////
///////////
// CGetPtDialog message handlers



BOOL CGetPtDialog::OnInitDialog()
{   CListBox* pLB = (CListBox*) GetDlgItem(IDC_PATIENT_LIST);
      BeginWaitCursor();
      pLB -> SetTabStops(156);
      while (!m_NameIDList.IsEmpty()) pLB -> AddString((const
char*)m_NameIDList.RemoveTail());
      EndWaitCursor();
      return CDialog::OnInitDialog();
}



  void CGetPtDialog::GetNames()
{     char buffer[MAX_INFILE_LENGTH]="";
      DWORD FileIndex = 0; //gives a zero-based ref to each name
in the open database (not counting header)
      CString strNameID;
      CString oldname = "";
      char* name = buffer + 45;
      char* id = buffer + 70;
      BeginWaitCursor();
      while (m_pFile -> ReadString(buffer,MAX_INFILE_LENGTH))
            {*(name + 24) = '\0';
            *(id + 10) = '\0';
            strNameID = name;
            strNameID  += "\t";
            strNameID  += id;

            if (strNameID != oldname)
                  {m_NameIDList.AddHead(strNameID);
                   oldname = strNameID;
                   m_FileIndexArray.Add(FileIndex);

            }
            FileIndex++;
```

```
	}
	m_FileIndexArray.Add(FileIndex);////////////////for the
upper bound for the last item

	EndWaitCursor();
}



void CGetPtDialog::OnClickedOK()
{   LPINT p_indices = m_index;
	int last;
	CString selection;
	CListBox* pLB = (CListBox*) GetDlgItem(IDC_PATIENT_LIST);
	m_SelCount = pLB -> GetSelCount();
//MAX_LB_SELECT
	if (m_SelCount > MAX_LB_SELECT){AfxMessageBox("Too many
patients selected\nPlease select fewer than 100",MB_ICONSTOP |
MB_OK);
		return;
//MAX_LB_SELECT
	}

	////////////////These lines for crippled
version///////////////
	//if (m_SelCount > 1){AfxMessageBox("Please only select 1
patient at a time\n for the demonstration version",MB_ICONSTOP |
MB_OK);
	//    return;
//MAX_LB_SELECT
	//}
	///////////////End of crippling
lines//////////////////////////

	if (!m_SelCount)return;
	last = pLB -> GetSelItems(MAX_LB_SELECT,p_indices);
	m_index[last] = m_index[last-1] + 1; ////////////////for the
upper bound for the last item

	((CProwin01App*)AfxGetApp())->m_nActiveDatabaseIndex =
m_nDatabaseIndex;
	((CProwin01App*)AfxGetApp())->DoNewDocument();
```

```
}

void CGetPtDialog::OnCancel()
{
      m_pFile -> Close();
      delete m_pFile;
      DestroyWindow();
      ((CProwin01App*)AfxGetApp())-
>m_pGetPtDialogArray[m_nDatabaseIndex] = NULL;//this
      delete this;
}

void CGetPtDialog::OnSelchangePatientList()
{
      CListBox* pLB = (CListBox*) GetDlgItem(IDC_PATIENT_LIST);
      CButton* pButton = (CButton*)GetDlgItem(IDOK);
      if (pLB -> GetSelCount()) pButton -> EnableWindow(TRUE);
            else pButton -> EnableWindow(FALSE);

}

void CGetPtDialog::Destroy()
{
      CGetPtDialog::OnCancel();
}


void CGetPtDialog::OnPaint()
{
      CPaintDC dc(this); // device context for painting

      if (this -> IsIconic())
            {dc.SetMapMode(MM_TEXT);
            dc.DrawIcon(2,2,(HICON)AfxGetApp()-
>LoadIcon(IDI_DATABASE));
      }

      // Do not call CDialog::OnPaint() for painting messages
}

void CGetPtDialog::OnClickedHelp()
{
/*    ((CProwin01App*)AfxGetApp()) -> */OnHelp();
```

```
}
```

```
// getptdlg.h : header file
//
#define MAX_LB_SELECT 100

/////////////////////////////////////////////////////////////////////
///////////
// CGetPtDialog dialog

class CGetPtDialog : public CDialog
{
// Construction
public:
      CGetPtDialog(CWnd* pParent = NULL); // standard constructor
is CGetPtDialog(CWnd* pParent = NULL);

// Dialog Data
      //{{AFX_DATA(CGetPtDialog)
      enum { IDD = IDD_GET_PT_DIALOG };
      //}}AFX_DATA

// Data members
      CStringList m_NameIDList;
      CDWordArray m_FileIndexArray;
      int m_index[MAX_LB_SELECT];
      int m_SelCount;
      int m_nDatabaseIndex;
      CStdioFile* m_pFile;

// Helper functions
      void GetNames();
      void Destroy();

// Implementation
      BOOL Create();
protected:
      virtual void DoDataExchange(CDataExchange* pDX);      //
DDX/DDV support

      // Generated message map functions
      //{{AFX_MSG(CGetPtDialog)
      virtual BOOL OnInitDialog();
      afx_msg void OnClickedOK();
      virtual void OnCancel();
      afx_msg void OnSelchangePatientList();
```

```
    afx_msg void OnPaint();
    afx_msg void OnClickedHelp();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

```
// leftview.cpp : implementation file
//

#include "stdafx.h"
#include "prowin01.h"
#include "leftview.h"
#include "prowidoc.h"
#include "datedlg.h"
#include "legdlg.h"
#include "progcrit.h"
#include "progind.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

#define BOX_DIMENSION 70
#define BOX_MARGIN 10
#define MISSING_VALUE 990
#define GRAY_BOX_DIMENSION 30


/////////////////////////////////////////////////////////////////////////
///////////
// CLeftView

IMPLEMENT_DYNCREATE(CLeftView, CScrollView)

CLeftView::CLeftView()
{
      m_bRightEye = FALSE;
      m_bBinocular = FALSE;
      m_bShowProgressingPoints = FALSE;
      m_bShowGrayscale = FALSE;
      m_bEsterman = FALSE;
      m_LightBlueBrush.CreateSolidBrush(RGB(0,0,255));
      m_LightRedBrush.CreateSolidBrush(RGB(255,0,0));
      m_DarkRedBrush.CreateSolidBrush(RGB(128,0,0));
      m_MagentaBrush.CreateSolidBrush(RGB(255,0,255));
      m_YellowBrush.CreateSolidBrush(RGB(255,255,0));
      m_GrayBrush.CreateSolidBrush(RGB(128,128,128));
      m_BlueGreenBrush.CreateSolidBrush(RGB(0,128,128));
      m_OliveBrush.CreateSolidBrush(RGB(128,128,0));
```

```
	m_DarkGreenBrush.CreateSolidBrush(RGB(0,128,0));
	m_LightGreenBrush.CreateSolidBrush(RGB(0,255,0));
	m_RedHatchBrush.CreateHatchBrush(HS_DIAGCROSS,RGB(255,0,0));
	m_DarkBlueBrush.CreateSolidBrush(RGB(0,0,128));
	m_WhiteBrush.CreateSolidBrush(RGB(255,255,255));
	m_NameColor = RGB(128,128,128);
	m_tracker.SetRectEmpty();

	for (int i = 0; i < 10; i++)
		{sprintf(m_strGrayBrushName,"GRAYBRUSH%d",i);
		m_GrayscaleBitmap[i].LoadBitmap(m_strGrayBrushName);
		TRACE("\tLoading %s\n",m_strGrayBrushName);

	m_GrayscaleBrush[i].CreatePatternBrush(&m_GrayscaleBitmap[i]
);
	}
}

CLeftView::~CLeftView()
{
}


BEGIN_MESSAGE_MAP(CLeftView, CScrollView)
	//{{AFX_MSG_MAP(CLeftView)
	ON_COMMAND(ID_EYE_RIGHT, OnEyeRight)
	ON_UPDATE_COMMAND_UI(ID_EYE_RIGHT, OnUpdateEyeRight)
	ON_COMMAND(ID_EYE_LEFT, OnEyeLeft)
	ON_UPDATE_COMMAND_UI(ID_EYE_LEFT, OnUpdateEyeLeft)
	ON_COMMAND(ID_VIEW_DATES, OnViewDates)
	ON_COMMAND(ID_VIEW_LEGEND, OnViewLegend)
	ON_COMMAND(ID_VIEW_PROGRESSING_POINTS,
OnViewProgressingPoints)
	ON_UPDATE_COMMAND_UI(ID_VIEW_PROGRESSING_POINTS,
OnUpdateViewProgressingPoints)
	ON_COMMAND(ID_VIEW_PROG_CRIT, OnViewProgCrit)
	ON_UPDATE_COMMAND_UI(ID_VIEW_LEGEND, OnUpdateViewLegend)
	ON_UPDATE_COMMAND_UI(ID_VIEW_DATES, OnUpdateViewDates)
	ON_WM_LBUTTONDOWN()
	ON_WM_MOUSEMOVE()
	ON_WM_SETCURSOR()
	ON_COMMAND(ID_FILTER_GAUSSIAN, OnFilterGaussian)
	ON_UPDATE_COMMAND_UI(ID_FILTER_GAUSSIAN,
OnUpdateFilterGaussian)
```

```
	ON_COMMAND(ID_WINDOW_CLOSEALL, OnWindowCloseall)
	ON_COMMAND(ID_VIEW_PROGRESSION_INDICES,
OnViewProgressionIndices)
	ON_UPDATE_COMMAND_UI(ID_VIEW_PROGRESSION_INDICES,
OnUpdateViewProgressionIndices)
	ON_COMMAND(ID_VIEW_GRAYSCALE, OnViewGrayscale)
	ON_UPDATE_COMMAND_UI(ID_VIEW_GRAYSCALE,
OnUpdateViewGrayscale)
	ON_COMMAND(ID_EYE_BINOCULAR, OnEyeBinocular)
	ON_UPDATE_COMMAND_UI(ID_EYE_BINOCULAR, OnUpdateEyeBinocular)
	ON_COMMAND(ID_VIEW_ESTERMAN_DEFECTS, OnViewEstermanDefects)
	ON_UPDATE_COMMAND_UI(ID_VIEW_ESTERMAN_DEFECTS,
OnUpdateViewEstermanDefects)
	ON_COMMAND(ID_EDIT_COPY, OnEditCopy)
	ON_COMMAND(ID_EDIT_COPY_SCREEN, OnEditCopyScreen)
	//}}AFX_MSG_MAP
	// Standard printing commands
	ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_PREVIEW,
CScrollView::OnFilePrintPreview)
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////
///////////
// CLeftView drawing

void CLeftView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo /* =
NULL*/)
{	CRect ClientRect;
	GetClientRect(&ClientRect);
	if (!pDC -> IsPrinting())
		{pDC -> SetMapMode(MM_ISOTROPIC);
		pDC -> SetWindowExt(1000,1000);
		pDC ->
SetViewportExt(ClientRect.right,ClientRect.bottom);
		pDC ->
SetViewportOrg(ClientRect.right/2,ClientRect.bottom/2);
	}

}


void CLeftView::OnInitialUpdate()
```

```
{
	CScrollView::OnInitialUpdate();

	CSize sizeTotal;
	// TODO: calculate the total size of this view
	sizeTotal.cx = sizeTotal.cy = 100;
	SetScrollSizes(MM_TEXT, sizeTotal);
}

void CLeftView::OnDraw(CDC* pDC)
{
	CProwin01Doc* pDoc = GetDocument();
	ASSERT_VALID(pDoc);

	int start_field;
	int bar_width;
	int sensitivity, l_sensitivity, r_sensitivity;
	int x;
	int y;
	int j;
	CFont NameFont;
	CFont EyeFont;
	CFont DateFont;
	CPen AxisPen(PS_DOT,1,RGB(255,255,255)) ;
	CPen Central20Pen(PS_INSIDEFRAME,1,RGB(0,0,255)) ;
	CString strInnerSlope;
	CString strInnerP;
	CString strEdgeSlope;
	CString strEdgeP;
	CRect ClientRect;
	int nInnerP;
	int nEdgeP;
	int lbsel;
	int nBinocLeftCount, nBinocRightCount;
	CBrush* pOldBrush = pDC -> SelectObject(&m_LightRedBrush);

	if(!m_tracker.IsRectNull())
		{pDC -> LPtoDP(m_tracker);
		InvalidateRect(CRect(m_tracker.TopLeft()-
CSize(10,10),m_tracker.Size()+CSize(20,20)),
			m_bShowGrayscale | m_bBinocular ? TRUE : FALSE);
		m_tracker.SetRectEmpty();
	}
```

```
      EyeFont.CreateFont(40,0,0,0,400,FALSE,FALSE,0,ANSI_CHARSET,O
UT_DEFAULT_PRECIS,CLIP_DEFAULT_PRECIS,DEFAULT_QUALITY,DEFAULT_PITC
H | FF_ROMAN, NULL);
      NameFont.CreateFont(50,0,0,0,400,FALSE,FALSE,0,ANSI_CHARSET,
OUT_DEFAULT_PRECIS,CLIP_DEFAULT_PRECIS,DEFAULT_QUALITY,DEFAULT_PIT
CH | FF_ROMAN, NULL);
      CFont* pOldFont = pDC -> SelectObject(&NameFont);
      GetClientRect(&ClientRect);
      pDC -> DPtoLP(ClientRect);

      if ( !(m_bShowGrayscale | m_bBinocular) )
            {if (!pDC -> IsPrinting())pDC ->
FillRect(ClientRect,&m_DarkBlueBrush);
            pDC -> SetBkColor (RGB(0,0,128));
            pDC -> SelectObject(&AxisPen);
            pDC -> MoveTo(0,450);
            pDC -> LineTo(0,-450);
            pDC -> MoveTo(-450,0);
            pDC -> LineTo(450,0);
      }

      if (pDC -> IsPrinting()&&(m_bShowGrayscale |
m_bBinocular))pDC->SetTextColor (RGB(0,0,0));
      else pDC->SetTextColor (m_NameColor);
      pDC -> TextOut(-500,-500,pDoc -> m_name);
      pDC->SetTextColor (m_bShowGrayscale | m_bBinocular ?
RGB(0,0,0) : RGB(255,255,255));
      pDC -> SelectObject(&EyeFont);

            if (!m_bBinocular)
                  {if (m_bRightEye)
                        {pDC -> TextOut(-500,-450,"Right Eye");
                        if (pDoc -> m_rightReanalyzed)
                              {if ( !m_bShowGrayscale )pDC ->
SelectObject(&NameFont);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,0));
                              pDC -> TextOut(-500,-
400,"REANALYZED");
                              if ( m_bShowGrayscale )pDC ->
ExcludeClipRect(-500,-400,-270,-360);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,255));
                        }
```

```
                        if (pDoc -> m_rightGaussianFiltered)
                              {if ( !m_bShowGrayscale )pDC ->
SelectObject(&NameFont);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(0,255,255));
                              pDC -> TextOut(-500,-
350,"FILTERED");
                              if ( m_bShowGrayscale )pDC ->
ExcludeClipRect(-500,-350,-330,-310);
                              pDC -> SelectObject(&EyeFont);
                              pDC -> TextOut(-500,-
310,"(Gaussian)");
                              if ( m_bShowGrayscale )pDC ->
ExcludeClipRect(-500,-310,-345,-270);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,255));
                        }
                  }
                  else
                        {pDC -> TextOut(-500,-450,"Left Eye  ");
                        if (pDoc -> m_leftReanalyzed)
                              {if ( !m_bShowGrayscale )pDC ->
SelectObject(&NameFont);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,0));
                              pDC -> TextOut(-500,-
400,"REANALYZED");
                              if (m_bShowGrayscale)pDC ->
ExcludeClipRect(-500,-400,-270,-360);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,255));
                        }
                        if (pDoc -> m_leftGaussianFiltered)
                              {if ( !m_bShowGrayscale )pDC ->
SelectObject(&NameFont);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(0,255,255));
                              pDC -> TextOut(-500,-
350,"FILTERED");
                              if ( m_bShowGrayscale )pDC ->
ExcludeClipRect(-500,-350,-330,-310);
                              pDC -> SelectObject(&EyeFont);
                              pDC -> TextOut(-500,-
310,"(Gaussian)");
```

```
                              if ( m_bShowGrayscale )pDC ->
ExcludeClipRect(-500,-310,-345,-270);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,255));
                        }
                  }
            }
            else
                  {pDC -> TextOut(-500,-450,"Binocular");
                  if (pDoc -> m_rightReanalyzed || pDoc ->
m_leftReanalyzed)
                        {pDC -> TextOut(-500,-400,"REANALYZED");
                        pDC -> ExcludeClipRect(-500,-400,-270,-
360);
                  }
                  if (pDoc -> m_rightGaussianFiltered || pDoc ->
m_leftGaussianFiltered)
                        {pDC -> TextOut(-500,-350,"FILTERED");
                        pDC -> ExcludeClipRect(-500,-350,-330,-
310);
                        pDC -> TextOut(-500,-310,"(Gaussian)");
                        pDC -> ExcludeClipRect(-500,-310,-345,-
270);
                  }
            }

      for ( int row = 0; row < 12; row++)
            for ( int col = 0; col < 12; col++)
m_nMatrix[row][col] = MISSING_VALUE;

      x = -2*BOX_DIMENSION -3*BOX_MARGIN;
      y = -5*BOX_DIMENSION -9*BOX_MARGIN;

      strInnerSlope = ((CProwin01App*)AfxGetApp()) ->
m_strMinInnerSlope;
      strEdgeSlope = ((CProwin01App*)AfxGetApp()) ->
m_strMinEdgeSlope;
      nInnerP = ((CProwin01App*)AfxGetApp()) -> m_nInnerPValue;
      nEdgeP = ((CProwin01App*)AfxGetApp()) -> m_nEdgePValue;

      if (nInnerP == 0)strInnerP = "(p >= 0.1)";
      if (nInnerP == 1)strInnerP = "(p < 0.1 )";
      if (nInnerP == 2)strInnerP = "(p < 0.05)";
      if (nInnerP == 3)strInnerP = "(p < 0.01)";
```

```
if (nInnerP == 4)strInnerP = "(p < 0.001)";
if (nEdgeP == 0)strEdgeP = "(p >= 0.1)";
if (nEdgeP == 1)strEdgeP = "(p < 0.1 )";
if (nEdgeP == 2)strEdgeP = "(p < 0.05)";
if (nEdgeP == 3)strEdgeP = "(p < 0.01)";
if (nEdgeP == 4)strEdgeP = "(p < 0.001)";


for (int i = 0; i < 76; i++)
      {if ( i == 4 )
            {x = -3*BOX_DIMENSION -5*BOX_MARGIN;
            y = -4*BOX_DIMENSION -7*BOX_MARGIN;
      }
      if ( i == 10 )
            {x = -4*BOX_DIMENSION -7*BOX_MARGIN;
            y = -3*BOX_DIMENSION -5*BOX_MARGIN;
      }
      if ( i == 18 )
            {x = -5*BOX_DIMENSION -9*BOX_MARGIN;
            y = -2*BOX_DIMENSION -3*BOX_MARGIN;
      }
      if ( i == 28 )
            {x = -5*BOX_DIMENSION -9*BOX_MARGIN;
            y = -BOX_DIMENSION -BOX_MARGIN;
      }
      if ( i == 38 )
            {x = -5*BOX_DIMENSION -9*BOX_MARGIN;
            y = BOX_MARGIN;
      }
      if ( i == 48 )
            {x = -5*BOX_DIMENSION -9*BOX_MARGIN;
            y = BOX_DIMENSION +3*BOX_MARGIN;
      }
      if ( i == 58 )
            {x = -4*BOX_DIMENSION -7*BOX_MARGIN;
            y = 2*BOX_DIMENSION +5*BOX_MARGIN;
      }
      if ( i == 66 )
            {x = -3*BOX_DIMENSION -5*BOX_MARGIN;
            y = 3*BOX_DIMENSION +7*BOX_MARGIN;
      }
      if ( i == 72 )
            {x = -2*BOX_DIMENSION -3*BOX_MARGIN;
            y = 4*BOX_DIMENSION +9*BOX_MARGIN;
```

```
            }

            m_BoxArray[i] = CRect(x - BOX_MARGIN/2, y -
BOX_MARGIN/2 , x + BOX_DIMENSION + BOX_MARGIN/2, y + BOX_DIMENSION
+ BOX_MARGIN/2);

            if(!m_bBinocular)
                  {if (m_bRightEye)
                        {if (m_nCount = pDoc ->
m_rightorderArray.GetSize())
                              {if (pDoc -> m_bDateDialogDisplayed)
                                    {lbsel = (int)pDoc ->
m_pDateDialog -> m_nRightSelCount;
                                    if (lbsel == 1)
                                          {m_nCount = (int)pDoc ->
m_pDateDialog -> m_nRightRestrictedDates[0]  + 1;

      OnMouseMove(m_nFlags,m_mousePos);
                                    }
                              }
                              start_field = 0;
                              if (m_nCount > MAX_FIELDS)
start_field = m_nCount - MAX_FIELDS;  // shows the last MAX_FIELDS
fields
                              bar_width =
BOX_DIMENSION/MAX_FIELDS;                              //bar_width
= BOX_DIMENSION/m_nCount; for scaling to fit
                              for ( j = start_field; j < m_nCount;
j++)
                                    if( (sensitivity = (WORD)pDoc
-> m_rightpointArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(j)))
                                          != MISSING_VALUE)
                                          {if (m_bShowGrayscale
&&( j == m_nCount - 1 /*the last field of the series*/) )

      AddToMatrix(&i,&sensitivity);

                                          if
(m_bShowProgressingPoints && ( j == m_nCount - 1 /*the last field
of the series*/) )
                                                if
(IsProgressing(&i,&j,pDoc,&m_bRightEye))
```

```
                                                        {pDC ->
FillRect( m_BoxArray[i], &m_RedHatchBrush);
                                                        if
(m_bShowGrayscale)pDC -> ExcludeClipRect(m_BoxArray[i]);
                                                    }

                                            if
(!m_bShowProgressingPoints && !m_bShowGrayscale)
                                                    pDC->
FillRect(CRect(x + (j-start_field)*bar_width,y,x + (j-
start_field+1)*bar_width,
                                                    y + (300 -
sensitivity)*BOX_DIMENSION/300),GetBrush(&i, &j,pDoc));

                                        }

                        }
                        else if (!i)
                              {pDC -> SelectObject(&NameFont);
                              pDC->SetTextColor (RGB(255,0,0));
                              pDC -> TextOut(-200,-50,"NO VALID
FIELDS" );
                        }

                  }
                  else
                        {if (m_nCount = pDoc ->
m_leftorderArray.GetSize())
                              {if (pDoc -> m_bDateDialogDisplayed)
                                    {lbsel = (int)pDoc ->
m_pDateDialog -> m_nLeftSelCount;
                                    if (lbsel == 1)
                                          {m_nCount = (int)pDoc ->
m_pDateDialog -> m_nLeftRestrictedDates[0]  + 1;

      OnMouseMove(m_nFlags,m_mousePos);
                                    }
                              }
                              start_field = 0;
                              if (m_nCount > MAX_FIELDS)
start_field = m_nCount - MAX_FIELDS;  // shows the last MAX_FIELDS
fields
```

```
                              bar_width =
BOX_DIMENSION/MAX_FIELDS;                                   //bar_width
= BOX_DIMENSION/m_nCount; for scaling to fit
                              for ( j = start_field; j < m_nCount;
j++)
                                    if( (sensitivity = (WORD)pDoc
-> m_leftpointArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(j)))
                                                != MISSING_VALUE)
                                                {if (m_bShowGrayscale
&&( j == m_nCount - 1 /*the last field of the series*/) )

      AddToMatrix(&i,&sensitivity);

                                                if
(m_bShowProgressingPoints && ( j == m_nCount - 1 /*the last field
of the series*/) )
                                                      if
(IsProgressing(&i,&j,pDoc,&m_bRightEye))
                                                            {pDC ->
FillRect( m_BoxArray[i], &m_RedHatchBrush);
                                                            if
(m_bShowGrayscale)pDC -> ExcludeClipRect(m_BoxArray[i]);
                                                      }

                                                if
(!m_bShowProgressingPoints && !m_bShowGrayscale)
                                                      pDC->
FillRect(CRect(x + (j-start_field)*bar_width,y,x + (j-
start_field+1)*bar_width,
                                                      y + (300 -
sensitivity)*BOX_DIMENSION/300),GetBrush(&i, &j,pDoc));

                                    }

                        }
                        else if (!i)
                              {pDC -> SelectObject(&NameFont);
                              pDC->SetTextColor (RGB(255,0,0));
                              pDC -> TextOut(-200,-50,"NO VALID
FIELDS" );
                        }

                  }
```

```
            }
            else
                  {if ((nBinocLeftCount = pDoc ->
m_leftorderArray.GetSize()) &&
                        (nBinocRightCount = pDoc ->
m_rightorderArray.GetSize()))
                        {if (pDoc -> m_bDateDialogDisplayed)
                              {lbsel = (int)pDoc -> m_pDateDialog
-> m_nLeftSelCount;
                              if (lbsel == 1)
                                    {nBinocLeftCount = (int)pDoc -
> m_pDateDialog -> m_nLeftRestrictedDates[0]  + 1;
                              }
                              lbsel = (int)pDoc -> m_pDateDialog -
> m_nRightSelCount;
                              if (lbsel == 1)
                                    {nBinocRightCount = (int)pDoc
-> m_pDateDialog -> m_nRightRestrictedDates[0]  + 1;
                              }
                        }

                        l_sensitivity = (WORD)pDoc ->
m_leftpointArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(nBinocLeftCount - 1));
                        r_sensitivity = (WORD)pDoc ->
m_rightpointArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(nBinocRightCount - 1));
                        sensitivity = l_sensitivity >
r_sensitivity ? l_sensitivity : r_sensitivity;
                        AddToMatrix(&i,&sensitivity);
                        if (m_bEsterman && sensitivity < 100 )
                              {pDC -> FillRect( m_BoxArray[i],
&m_DarkBlueBrush);
                              pDC -> Ellipse( m_BoxArray[i]);
                              pDC -> SelectObject(&EyeFont);
                              pDC->SetTextColor (RGB(0,0,0));
                              pDC -> TextOut(x+7,y+15,"<10");
                              pDC ->
ExcludeClipRect(m_BoxArray[i]);
                        }
                  }
            }
```

```
            x += BOX_DIMENSION + 2*BOX_MARGIN;

      }

      if (m_bShowGrayscale | m_bBinocular)Interpolate(pDC);

      if (m_bBinocular && m_bEsterman)
            {pDC -> SelectStockObject(HOLLOW_BRUSH);
            pDC -> SelectObject(&Central20Pen);
            pDC -> Ellipse(-300,-300,300,300);
      }

      if (m_bShowProgressingPoints && !m_bBinocular)
            {pDC -> SelectObject(&EyeFont);
            pDC->SetTextColor (m_bShowGrayscale ? RGB(0,0,0) :
RGB(255,255,255));
            pDC -> TextOut(-500,460,"Inner: < -" + strInnerSlope +
" dB / yr " + strInnerP);
            pDC -> TextOut(0,460,"Edge: < -" + strEdgeSlope + " dB
/ yr " + strEdgeP );
      }

      if ( !pDoc -> m_bDateDialogDisplayed )
//for progression indices dialog
            {pDoc -> m_nLeftCount = pDoc ->
m_leftorderArray.GetSize(); //for progression indices dialog
            pDoc -> m_nRightCount = pDoc ->
m_rightorderArray.GetSize();//for progression indices dialog
      }
      pDC -> SelectStockObject(NULL_PEN);
      pDC -> SelectObject(pOldFont);
      pDC -> SelectObject(pOldBrush);
}

#pragma optimize ("",off)
CBrush* CLeftView::GetBrush(int* point, int* field, CProwin01Doc*
pDoc)
{     BYTE pvalue;
      CString slopeString;
      long double slope;

      if (m_bRightEye)pvalue = (BYTE)pDoc ->
m_rightpvalueArray.GetAt(*point + 76*pDoc ->
m_rightorderArray.GetAt(*field));
```

```
            else pvalue = (BYTE)pDoc ->
m_leftpvalueArray.GetAt(*point + 76*pDoc ->
m_leftorderArray.GetAt(*field));

      if (m_bRightEye)slopeString = (CString)pDoc ->
m_rightslopeArray.GetAt(*point + 76*pDoc ->
m_rightorderArray.GetAt(*field));
            else slopeString = (CString)pDoc ->
m_leftslopeArray.GetAt(*point + 76*pDoc ->
m_leftorderArray.GetAt(*field));

      slope = _atold((const char*)slopeString);
      if ( (fabsl(slope) <= 1) && (pvalue != 10)/* ie excluded in
Doc*/ ) return &m_GrayBrush; /////////cutoff for 'flat'

      switch (pvalue)
            {case 11: return &m_WhiteBrush;
            case 10: return &m_LightBlueBrush;
            case 9: return &m_LightRedBrush;
            case 8: return &m_DarkRedBrush;
            case 7: return &m_MagentaBrush;
            case 6: return &m_YellowBrush;
            case 5: return &m_GrayBrush;
            case 4: return &m_BlueGreenBrush;
            case 3: return &m_OliveBrush;
            case 2: return &m_DarkGreenBrush;
            case 1: return &m_LightGreenBrush;
            default: return &m_LightBlueBrush;
      }

}
#pragma optimize ("",on)

BOOL CLeftView::IsProgressing(int* point,int*
last_field,CProwin01Doc* pDoc,BOOL* pRightEye)
{     // characteristics of the point from the Doc
      BOOL retval = FALSE;
      BYTE pvalue;
      CString slopeString;
      long double slope;

      // progression criteria from the App
      int p_crit;
      CString slope_critString;
```

```
      long double slope_crit;

      if (*pRightEye)pvalue = (BYTE)pDoc ->
m_rightpvalueArray.GetAt(*point + 76*pDoc ->
m_rightorderArray.GetAt(*last_field));
            else pvalue = (BYTE)pDoc ->
m_leftpvalueArray.GetAt(*point + 76*pDoc ->
m_leftorderArray.GetAt(*last_field));

      if (*pRightEye)slopeString = (CString)pDoc ->
m_rightslopeArray.GetAt(*point + 76*pDoc ->
m_rightorderArray.GetAt(*last_field));
            else slopeString = (CString)pDoc ->
m_leftslopeArray.GetAt(*point + 76*pDoc ->
m_leftorderArray.GetAt(*last_field));

      slope = _atold((const char*)slopeString);
      if (slope < 0)
            {// Get appropriate progression criteria
            if ( *point < 5 || *point == 9 || *point == 10 ||
*point == 17 || *point == 18 || *point == 27 || *point == 28
            || *point == 37|| *point == 38 || *point == 47 ||
*point == 48 || *point == 57 || *point == 58 || *point == 65
            || *point == 66|| *point > 70 )
                  {p_crit = ((CProwin01App*)AfxGetApp()) ->
m_nEdgePValue;
                  slope_critString = ((CProwin01App*)AfxGetApp())
-> m_strMinEdgeSlope;
            }
            else
                  {p_crit = ((CProwin01App*)AfxGetApp()) ->
m_nInnerPValue;
                  slope_critString = ((CProwin01App*)AfxGetApp())
-> m_strMinInnerSlope;

            }
            slope_crit = _atold((const char*)slope_critString);

            // Compare
            if (fabsl(slope) > slope_crit )
                  {if (p_crit == 0 && (pvalue == 6 || pvalue == 7
|| pvalue == 8 || pvalue == 9 || pvalue == 11) )retval = TRUE;
                  if (p_crit == 1 && (pvalue == 7 || pvalue == 8
|| pvalue == 9 || pvalue == 11) )retval = TRUE;
```

```
                if (p_crit == 2 && (pvalue == 8 || pvalue == 9
|| pvalue == 11) )retval = TRUE;
                if (p_crit == 3 && (pvalue == 9 || pvalue ==
11))retval = TRUE;
                if (p_crit == 4 && pvalue == 11)retval = TRUE;
            }
      }

    return retval;
}

void CLeftView::OnActivateView(BOOL bActivate, CView*
pActivateView, CView* pDeactiveView)
{
      CClientDC dc(this);
      CRect InvalidRect(-500,-500,500,-450);
      OnPrepareDC(&dc);
      dc.LPtoDP(InvalidRect);
      if (bActivate)
            {if ( !(m_bShowGrayscale || m_bBinocular) )m_NameColor
= RGB(255,255,255);
            else m_NameColor = RGB(0,0,0);
      }
      else
            {m_NameColor = RGB(128,128,128);
            if(!m_tracker.IsRectNull())
                  {dc.LPtoDP(m_tracker);
                  InvalidateRect(CRect(m_tracker.TopLeft()-
CSize(10,10),m_tracker.Size()+CSize(20,20)),
                        (m_bShowGrayscale) ? TRUE : FALSE);
                  m_tracker.SetRectEmpty();
            }
      }

      InvalidateRect(InvalidRect,FALSE);
}

void CLeftView::OnUpdate(CView* pView, LPARAM lHint, CObject*
pHint)
{
      if(!m_bBinocular)
            {if (lHint == 1 && m_bRightEye)Invalidate(FALSE);
            if (lHint == 2 && !m_bRightEye)Invalidate(FALSE);
            if (lHint == 3)Invalidate(FALSE);
```

```
	}
	else Invalidate(FALSE);
}



////////////////////////////////////////////////////////////////
///////////
// CLeftView printing

BOOL CLeftView::OnPreparePrinting(CPrintInfo* pInfo)
{
	if (m_bBinocular) pInfo -> SetMaxPage(1);
	else pInfo -> SetMaxPage(2);
	BOOL bRet = DoPreparePrinting(pInfo);
	if (m_bBinocular)pInfo -> m_nNumPreviewPages = 1;
	else pInfo -> m_nNumPreviewPages = 2;
	return bRet;
}

void CLeftView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo*
/*pInfo*/)
{
	// TODO: add extra initialization before printing
}

void CLeftView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: add cleanup after printing
}

////////////////////////////////////////////////////////////////
///////////
// CLeftView diagnostics

#ifdef _DEBUG
void CLeftView::AssertValid() const
{
	CScrollView::AssertValid();
}

void CLeftView::Dump(CDumpContext& dc) const
{
	CScrollView::Dump(dc);
```

```
}

CProwin01Doc* CLeftView::GetDocument() // non-debug version is
inline
{
      ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CProwin01Doc)));
      return (CProwin01Doc*)m_pDocument;
}
#endif //_DEBUG




////////////////////////////////////////////////////////////////
///////////
// CLeftView message handlers

void CLeftView::OnEyeRight()
{
      m_bRightEye = TRUE;
      Invalidate(FALSE);
}

void CLeftView::OnUpdateEyeRight(CCmdUI* pCmdUI)
{
      if(m_bBinocular)pCmdUI -> Enable(FALSE);
      else pCmdUI -> Enable(!m_bRightEye);
}

void CLeftView::OnEyeLeft()
{
      m_bRightEye = FALSE;
      Invalidate(FALSE);
}

void CLeftView::OnUpdateEyeLeft(CCmdUI* pCmdUI)
{
      if(m_bBinocular)pCmdUI -> Enable(FALSE);
      else pCmdUI -> Enable(m_bRightEye);
}

void CLeftView::OnViewDates()
{
      CProwin01Doc* pDoc = GetDocument();
```

```
      if (!(pDoc -> m_bDateDialogDisplayed))
            {pDoc -> m_pDateDialog -> m_pDoc = pDoc;
            pDoc -> m_pDateDialog -> Create();
            pDoc -> m_bDateDialogDisplayed = TRUE;
      }
}

void CLeftView::OnViewLegend()
{   //CProwin01App* pProwin01App = theApp;

      if (!((CProwin01App*)AfxGetApp()) ->
m_bLegendDialogDisplayed)
            {CLegendDialog* pLegendDialog = new CLegendDialog;
            pLegendDialog -> Create();
            ((CProwin01App*)AfxGetApp()) ->
m_bLegendDialogDisplayed = TRUE;
      }

      //delete pLegendDialog is done in CLegendDialog::OnCancel()
as: delete this;
}

void CLeftView::OnViewProgressingPoints()
{
      m_bShowProgressingPoints = m_bShowProgressingPoints ^ 1;
      Invalidate(FALSE);
}

void CLeftView::OnUpdateViewProgressingPoints(CCmdUI* pCmdUI)
{
      if(m_bBinocular)pCmdUI -> Enable(FALSE);
      else pCmdUI -> SetCheck(m_bShowProgressingPoints);
}

void CLeftView::OnViewProgCrit()
{   CProwin01Doc* pDoc = GetDocument();
      CProgCritDialog* pDlg = new CProgCritDialog;
      if( pDlg -> DoModal() == IDOK )
            {pDoc -> UpdateAllViews(NULL,3L,NULL);
            if (pDoc -> m_bProgIndDlgDisplayed) pDoc ->
m_pProgIndDlg -> OnInitDialog();
      }
      delete pDlg;
}
```

```
void CLeftView::OnUpdateViewLegend(CCmdUI* pCmdUI)
{
      pCmdUI -> Enable(!((CProwin01App*)AfxGetApp()) ->
m_bLegendDialogDisplayed);
}

void CLeftView::OnUpdateViewDates(CCmdUI* pCmdUI)
{
      CProwin01Doc* pDoc = GetDocument();
      pCmdUI -> Enable(!(pDoc -> m_bDateDialogDisplayed));
}



void CLeftView::OnLButtonDown(UINT nFlags, CPoint point)
{
      CProwin01Doc* pDoc = GetDocument();

      CClientDC dc(this);
      CRect rect,ClientRect,dummyRect;
      CLeftSquareTracker recttracker;

      GetClientRect(&ClientRect);
      OnPrepareDC(&dc);

      int bar_width,bar_length,j;
      WORD sensitivity;

      if (!m_bBinocular)
            {if(m_tracker.IsRectNull())
                  {for(int i = 0; i < 76; i++)
                        {rect = m_BoxArray[i];
                        dc.LPtoDP(rect);
                        if (rect.PtInRect(point))
                              {::SetCursor((HCURSOR)AfxGetApp() ->
LoadStandardCursor(IDC_CROSS));
                              CPen
FramePen(PS_SOLID,0,RGB(0,0,255)) ;
                              recttracker.m_rect = rect;

      recttracker.TrackRubberBand(this,rect.TopLeft(),FALSE);
                              recttracker.m_rect.NormalizeRect();
```

```
                                if (recttracker.m_rect.Width() >
rect.Width() &&
dummyRect.IntersectRect(ClientRect,recttracker.m_rect))
                                        {dc.DPtoLP(rect);
                                        dc.DPtoLP(recttracker.m_rect);
                                        m_tracker =
recttracker.m_rect;

      recttracker.m_rect.InflateRect(-m_tracker.Width()*0.1,-
m_tracker.Height()*0.1);
                                        if(m_nCount > MAX_FIELDS)
bar_width = recttracker.m_rect.Width()/m_nCount;
                                                else bar_width =
recttracker.m_rect.Width()/MAX_FIELDS;
                                        if (m_bRightEye)

      {dc.FillRect(CRect(m_tracker.left,m_tracker.top,m_tracker.ri
ght,m_tracker.bottom),&m_DarkBlueBrush);
                                                for ( j = 0; j <
m_nCount; j++)
                                                        {sensitivity =
(WORD)pDoc -> m_rightpointArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(j));
                                                        if (sensitivity !=
MISSING_VALUE )
                                                                {bar_length
= recttracker.m_rect.Height() * (300 - (float) sensitivity) / 300;
                                                                if
(bar_length < m_tracker.top - recttracker.m_rect.top)

      bar_length = m_tracker.top - recttracker.m_rect.top;

      dc.FillRect(CRect(recttracker.m_rect.left + (j)*bar_width,

      recttracker.m_rect.top,recttracker.m_rect.left +
(j+1)*bar_width,

      recttracker.m_rect.top + bar_length),GetBrush(&i, &j,pDoc));
                                                        }
                                                }
                                        }
                                        else
```

```
      {dc.FillRect(CRect(m_tracker.left,m_tracker.top,m_tracker.ri
ght,m_tracker.bottom),&m_DarkBlueBrush);
                                          for ( j = 0; j <
m_nCount; j++)
                                                {sensitivity =
(WORD)pDoc -> m_leftpointArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(j));
                                                if (sensitivity !=
MISSING_VALUE )
                                                      {bar_length
= recttracker.m_rect.Height() * (300 - (float) sensitivity) / 300;
                                                      if
(bar_length < m_tracker.top - recttracker.m_rect.top)

      bar_length = m_tracker.top - recttracker.m_rect.top;

      dc.FillRect(CRect(recttracker.m_rect.left + (j)*bar_width,

      recttracker.m_rect.top,recttracker.m_rect.left +
(j+1)*bar_width,

      recttracker.m_rect.top + bar_length),GetBrush(&i, &j,pDoc));
                                                }
                                          }
                                    }
                                    CPen* pOldPen =
dc.SelectObject(&FramePen);

      dc.MoveTo(m_tracker.BottomRight());

      dc.LineTo(m_tracker.left,m_tracker.bottom);

      dc.LineTo(m_tracker.TopLeft());

      dc.LineTo(m_tracker.BottomRight().x,m_tracker.TopLeft().y);

      dc.LineTo(m_tracker.BottomRight());
                                    dc.SelectObject(pOldPen);
                              }
                        }
                  }
            }
            else
```

```
                  {rect = m_tracker;
                  dc.LPtoDP(rect);
                  if (rect.PtInRect(point))
                        {InvalidateRect(CRect(rect.TopLeft()-
CSize(10,10),rect.Size()+CSize(20,20)),
                              (m_bShowGrayscale) ? TRUE : FALSE);
                        m_tracker.SetRectEmpty();
                  }
            }
      }

}

void CLeftView::OnMouseMove(UINT nFlags, CPoint point)
{
      CClientDC dc(this);
      CRect rect;
      CProwin01Doc* pDoc = GetDocument();
      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);

      float sensitivity;
      BYTE pvalue;
      CString pstring;
      CString slopeString;
      long double slope;
      char message[150];

      m_nFlags = nFlags;
      m_mousePos = point;

      OnPrepareDC(&dc);

      if(!m_bBinocular)
            {if(m_tracker.IsRectNull())
                  {for(int i = 0; i < 76; i++)
                        {rect = m_BoxArray[i];
                        dc.LPtoDP(rect);
                        if (rect.PtInRect(point))
                              {if (m_bRightEye)
                                    {if (pDoc ->
m_rightorderArray.GetSize())
```

```
                                        {sensitivity =
(WORD)pDoc -> m_rightpointArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(m_nCount - 1));
                                        if (sensitivity ==
MISSING_VALUE)sprintf(message,"30 - 2 location: not tested on this
date");
                                        else
                                              {pvalue =
(BYTE)pDoc -> m_rightpvalueArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(m_nCount - 1));
                                              if ((pvalue ==
4)||(pvalue == 6))pstring = "p >= 0.1";
                                              if ((pvalue ==
3)||(pvalue == 7))pstring = "p < 0.1";
                                              if ((pvalue ==
2)||(pvalue == 8))pstring = "p < 0.05";
                                              if ((pvalue ==
1)||(pvalue == 9))pstring = "p < 0.01";
                                              if ((pvalue ==
11))pstring = "p < 0.001";
                                              slopeString =
(CString)pDoc -> m_rightslopeArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(m_nCount - 1));
                                              if
(slopeString.IsEmpty())

      {sprintf(message,"Sensitivity: %2.1f dB  Slope is undefined:
too few fields for analysis",sensitivity/10);
                                              }
                                              else
                                                    {slope =
_atold((const char*)slopeString);

      sprintf(message,"Sensitivity: %2.1f dB    Slope: %.2Lf dB /
yr    %s",sensitivity/10,slope,pstring);
                                              }
                                        }
                                        pStatus-
>SetPaneText(0,message,TRUE);
                                        pStatus->UpdateWindow();
                                        }
                                  }
                            else
```

```
                                    {if (pDoc ->
m_leftorderArray.GetSize())
                                        {sensitivity =
(WORD)pDoc -> m_leftpointArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(m_nCount - 1));
                                        if (sensitivity ==
MISSING_VALUE)sprintf(message,"30 - 2 location: not tested on this
date");
                                        else
                                            {pvalue =
(BYTE)pDoc -> m_leftpvalueArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(m_nCount - 1));
                                            if ((pvalue ==
4)||(pvalue == 6))pstring = "p >= 0.1";
                                            if ((pvalue ==
3)||(pvalue == 7))pstring = "p < 0.1";
                                            if ((pvalue ==
2)||(pvalue == 8))pstring = "p < 0.05";
                                            if ((pvalue ==
1)||(pvalue == 9))pstring = "p < 0.01";
                                            if ((pvalue ==
11))pstring = "p < 0.001";
                                            slopeString =
(CString)pDoc -> m_leftslopeArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(m_nCount - 1));
                                            if
(slopeString.IsEmpty())

      {sprintf(message,"Sensitivity: %2.1f dB  Slope is undefined:
too few fields for analysis",sensitivity/10);
                                            }
                                            else
                                                {slope =
_atold((const char*)slopeString);

      sprintf(message,"Sensitivity: %2.1f dB    Slope: %.2Lf dB /
yr    %s",sensitivity/10,slope,pstring);
                                            }
                                        }
                                        pStatus-
>SetPaneText(0,message,TRUE);
                                        pStatus->UpdateWindow();
                                        }
                                    }
```

```
                              m_bCursorInRect = TRUE;
                              break;
                        }
                        else
                              {//pStatus->SetPaneText(0,"For Help,
press F1",TRUE);
                              //pStatus->UpdateWindow();
                              m_bCursorInRect = FALSE;
                        }
                  }
            }
            else
                  {rect = m_tracker;
                  dc.LPtoDP(rect);
                  if (rect.PtInRect(point))m_bCursorInRect = TRUE;
                  else m_bCursorInRect = FALSE;
            }

      }
      else
            {sprintf(message,"Binocular simulation");
            pStatus->SetPaneText(0,message,TRUE);
            pStatus->UpdateWindow();
      }

      CScrollView::OnMouseMove(nFlags, point);
}

BOOL CLeftView::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT
message)
{
      if (!m_bBinocular)
            {if(m_tracker.IsRectNull())

      {if(m_bCursorInRect)::SetCursor((HCURSOR)AfxGetApp() ->
LoadCursor(IDC_MAGNIFY));
                  else ::SetCursor((HCURSOR)AfxGetApp() ->
LoadStandardCursor(IDC_ARROW));
            }
            else

      {if(m_bCursorInRect)::SetCursor((HCURSOR)AfxGetApp() ->
LoadCursor(IDC_MINIFY));
```

```
                else ::SetCursor((HCURSOR)AfxGetApp() ->
LoadStandardCursor(IDC_ARROW));
            }
      }
      else ::SetCursor((HCURSOR)AfxGetApp() ->
LoadStandardCursor(IDC_ARROW));

      return TRUE;
}

void CLeftSquareTracker::AdjustRect(int nHandle, LPRECT lpRect)
{   CRect rect = *lpRect;
    int width = rect.Width();
      int height = rect.Height();
      if (width > height)rect.SetRect(rect.TopLeft().x,
rect.BottomRight().y -
width,rect.BottomRight().x,rect.BottomRight().y);
      if (height > width)rect.SetRect(rect.BottomRight().x -
height,
rect.TopLeft().y,rect.BottomRight().x,rect.BottomRight().y);
      *lpRect = rect;
}

void CLeftView::OnFilterGaussian()
{
      BYTE filter[3][3] ={{1,2,1},
                                  {2,4,2},
                                  {1,2,1}};
      int fx,fy;
      BYTE divisor;
      WORD filtered;

      int current_field, no_of_fields, pointno;

      int row,col;
      WORD matrix[12][12];

      CProwin01Doc* pDoc = GetDocument();

      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);

      if ( m_bRightEye )
```

```
            {no_of_fields = pDoc -> m_rightpointArray.GetSize() /
76;
            if (!no_of_fields)return;
                  BeginWaitCursor();
                  if (!(pDoc -> m_rightGaussianFiltered))
                        {pStatus->SetPaneText(0,"Applying Gaussian
filter...",TRUE);
                        pStatus->UpdateWindow();

                        for (int i = 0; i < pDoc ->
m_rightpointArray.GetSize(); i++)
                              pDoc ->
m_rightUndoArray.SetAtGrow(i, pDoc -> m_rightpointArray[i]);

                  for ( current_field = 0; current_field <
no_of_fields; current_field++ )
                        {for ( row = 0; row < 12; row++)
                              for ( col = 0; col < 12;
col++)matrix[row][col] = MISSING_VALUE;

                        for ( pointno = 0; pointno < 76;
pointno++)
                              { if ( pointno < 4 )
matrix[1][pointno + 4] = pDoc -> m_rightpointArray.GetAt( pointno
+ current_field * 76);
                              else if ( pointno > 3 && pointno <
10 ) matrix[2][pointno - 1] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                              else if ( pointno > 9 && pointno <
18 ) matrix[3][pointno - 8] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 17 && pointno < 28
) matrix[4][pointno - 17] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 27 && pointno < 38
) matrix[5][pointno - 27] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 37 && pointno < 48
) matrix[6][pointno - 37] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 47 && pointno < 58
) matrix[7][pointno - 47] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
```

```
                        else if ( pointno > 57 && pointno < 66
) matrix[8][pointno - 56] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                        else if ( pointno > 65 && pointno < 72
) matrix[9][pointno - 63] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                        else if ( pointno > 71 )
matrix[10][pointno - 68] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                    }
                    matrix[5][8] = matrix[6][8] =
MISSING_VALUE;///////blind spot excluded


                    for ( row = 1; row < 11; row++)
                        for ( col = 1; col < 11; col++)
                            {if ( matrix[row][col] ==
MISSING_VALUE )continue;
                            divisor = 0;
                            filtered = 0;
                            for ( fx = 0; fx < 3; fx++)
                                for ( fy = 0; fy < 3;
fy++)
                                    {if
(matrix[row+fx-1][col+fy-1] != MISSING_VALUE)
                                        {filtered +=
matrix[row+fx-1][col+fy-1] * filter[fx][fy];
                                        divisor +=
filter[fx][fy];
                                    }
                                }
                            filtered /= divisor;
                            if ( row == 1 && col > 3 &&
col < 8 )pDoc -> m_rightpointArray.SetAt( col - 4 + current_field
* 76, filtered);
                            else if ( row == 2 && col > 2
&& col < 9 )pDoc -> m_rightpointArray.SetAt( col + 1 +
current_field * 76, filtered);
                            else if ( row == 3 && col > 1
&& col < 10 )pDoc -> m_rightpointArray.SetAt( col + 8 +
current_field * 76, filtered);
                            else if ( row == 4 && col > 0
&& col < 11 )pDoc -> m_rightpointArray.SetAt( col + 17 +
current_field * 76, filtered);
```

```
                                    else if ( row == 5 && col > 0
&& col < 11 )pDoc -> m_rightpointArray.SetAt( col + 27 +
current_field * 76, filtered);
                                    else if ( row == 6 && col > 0
&& col < 11 )pDoc -> m_rightpointArray.SetAt( col + 37 +
current_field * 76, filtered);
                                    else if ( row == 7 && col > 0
&& col < 11 )pDoc -> m_rightpointArray.SetAt( col + 47 +
current_field * 76, filtered);
                                    else if ( row == 8 && col > 1
&& col < 10 )pDoc -> m_rightpointArray.SetAt( col + 56 +
current_field * 76, filtered);
                                    else if ( row == 9 && col > 2
&& col < 9 )pDoc -> m_rightpointArray.SetAt( col + 63 +
current_field * 76, filtered);
                                    else if ( row == 10 && col > 3
&& col < 8 )pDoc -> m_rightpointArray.SetAt( col + 68 +
current_field * 76, filtered);
                                }
                }
                pDoc -> Regress();
                //pDoc -> SetModifiedFlag();
                pDoc -> m_rightGaussianFiltered = 1;
                pDoc -> UpdateAllViews(NULL,1L,NULL);

            }
            else
                {pStatus->SetPaneText(0,"Removing Gaussian
filter...",TRUE);
                pStatus->UpdateWindow();

                for (int i = 0; i < pDoc ->
m_rightpointArray.GetSize(); i++)
                        pDoc -> m_rightpointArray[i] =  pDoc ->
m_rightUndoArray[i];

                pDoc -> m_rightUndoArray.RemoveAll();
                pDoc -> Regress();
                //pDoc -> SetModifiedFlag();
                pDoc -> m_rightGaussianFiltered = 0;
                pDoc -> UpdateAllViews(NULL,1L,NULL);
            }
      }
      else
```

```
            {no_of_fields = pDoc -> m_leftpointArray.GetSize() /
76;
            if (!no_of_fields)return;
                  BeginWaitCursor();
                  if (!(pDoc -> m_leftGaussianFiltered))
                        {pStatus->SetPaneText(0,"Applying Gaussian
filter...",TRUE);
                        pStatus->UpdateWindow();

                        for (int i = 0; i < pDoc ->
m_leftpointArray.GetSize(); i++)
                              pDoc ->
m_leftUndoArray.SetAtGrow(i, pDoc -> m_leftpointArray[i]);

                  for ( current_field = 0; current_field <
no_of_fields; current_field++ )
                        {for ( row = 0; row < 12; row++)
                              for ( col = 0; col < 12;
col++)matrix[row][col] = MISSING_VALUE;

                        for ( pointno = 0; pointno < 76;
pointno++)
                              { if ( pointno < 4 )
matrix[1][pointno + 4] = pDoc -> m_leftpointArray.GetAt( pointno +
current_field * 76);
                              else if ( pointno > 3 && pointno <
10 ) matrix[2][pointno - 1] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                              else if ( pointno > 9 && pointno <
18 ) matrix[3][pointno - 8] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 17 && pointno < 28
) matrix[4][pointno - 17] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                              else if ( pointno > 27 && pointno < 38
) matrix[5][pointno - 27] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                              else if ( pointno > 37 && pointno < 48
) matrix[6][pointno - 37] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                              else if ( pointno > 47 && pointno < 58
) matrix[7][pointno - 47] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
```

```
                        else if ( pointno > 57 && pointno < 66
) matrix[8][pointno - 56] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                        else if ( pointno > 65 && pointno < 72
) matrix[9][pointno - 63] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                        else if ( pointno > 71 )
matrix[10][pointno - 68] = pDoc -> m_leftpointArray.GetAt( pointno
+ current_field * 76);
                    }
                    matrix[5][8] = matrix[6][8] =
MISSING_VALUE;///////blind spot excluded


                    for ( row = 1; row < 11; row++)
                        for ( col = 1; col < 11; col++)
                            {if ( matrix[row][col] ==
MISSING_VALUE )continue;
                            divisor = 0;
                            filtered = 0;
                            for ( fx = 0; fx < 3; fx++)
                                for ( fy = 0; fy < 3;
fy++)
                                    {if
(matrix[row+fx-1][col+fy-1] != MISSING_VALUE)
                                        {filtered +=
matrix[row+fx-1][col+fy-1] * filter[fx][fy];
                                        divisor +=
filter[fx][fy];
                                    }
                                }
                            filtered /= divisor;
                            if ( row == 1 && col > 3 &&
col < 8 )pDoc -> m_leftpointArray.SetAt( col - 4 + current_field *
76, filtered);
                            else if ( row == 2 && col > 2
&& col < 9 )pDoc -> m_leftpointArray.SetAt( col + 1 +
current_field * 76, filtered);
                            else if ( row == 3 && col > 1
&& col < 10 )pDoc -> m_leftpointArray.SetAt( col + 8 +
current_field * 76, filtered);
                            else if ( row == 4 && col > 0
&& col < 11 )pDoc -> m_leftpointArray.SetAt( col + 17 +
current_field * 76, filtered);
```

```
                                    else if ( row == 5 && col > 0
&& col < 11 )pDoc -> m_leftpointArray.SetAt( col + 27 +
current_field * 76, filtered);
                                    else if ( row == 6 && col > 0
&& col < 11 )pDoc -> m_leftpointArray.SetAt( col + 37 +
current_field * 76, filtered);
                                    else if ( row == 7 && col > 0
&& col < 11 )pDoc -> m_leftpointArray.SetAt( col + 47 +
current_field * 76, filtered);
                                    else if ( row == 8 && col > 1
&& col < 10 )pDoc -> m_leftpointArray.SetAt( col + 56 +
current_field * 76, filtered);
                                    else if ( row == 9 && col > 2
&& col < 9 )pDoc -> m_leftpointArray.SetAt( col + 63 +
current_field * 76, filtered);
                                    else if ( row == 10 && col > 3
&& col < 8 )pDoc -> m_leftpointArray.SetAt( col + 68 +
current_field * 76, filtered);
                                }
                }
                pDoc -> Regress();
                //pDoc -> SetModifiedFlag();
                pDoc -> m_leftGaussianFiltered = 1;
                pDoc -> UpdateAllViews(NULL,2L,NULL);

            }
            else
                  {pStatus->SetPaneText(0,"Removing Gaussian
filter...",TRUE);
                pStatus->UpdateWindow();

                for (int i = 0; i < pDoc ->
m_leftpointArray.GetSize(); i++)
                        pDoc -> m_leftpointArray[i] =  pDoc ->
m_leftUndoArray[i];

                pDoc -> m_leftUndoArray.RemoveAll();
                pDoc -> Regress();
                //pDoc -> SetModifiedFlag();
                pDoc -> m_leftGaussianFiltered = 0;
                pDoc -> UpdateAllViews(NULL,2L,NULL);
            }
      }
```

```
	if (pDoc -> m_bProgIndDlgDisplayed) pDoc -> m_pProgIndDlg ->
OnInitDialog();
	if ((m_bRightEye && pDoc -> m_rightGaussianFiltered) ||
(!m_bRightEye && pDoc -> m_leftGaussianFiltered))
		pStatus->SetPaneText(0,"Gaussian filter applied. For
Help, press F1",TRUE);
	else pStatus->SetPaneText(0,"Gaussian filter removed. For
Help, press F1",TRUE);
	pStatus->UpdateWindow();
	EndWaitCursor();

}

void CLeftView::OnUpdateFilterGaussian(CCmdUI* pCmdUI)
{
	CProwin01Doc* pDoc = GetDocument();
	if(m_bBinocular)pCmdUI -> Enable(FALSE);
	else
		{if (m_bRightEye)pCmdUI -> SetCheck(pDoc ->
m_rightGaussianFiltered);
		else pCmdUI -> SetCheck(pDoc ->
m_leftGaussianFiltered);
	}

}

void CLeftView::OnWindowCloseall()
{
	//if (AfxMessageBox("All windows will be closed and unsaved
data will be lost.\n\n\tDo you wish to continue?",
	//MB_YESNO | MB_DEFBUTTON2 | MB_ICONQUESTION) == IDYES)
		((CProwin01App*)AfxGetApp()) ->
CloseAllDocuments(FALSE);

}

void CLeftView::OnViewProgressionIndices()
{
	CProwin01Doc* pDoc = GetDocument();
	CLeftView* pLeftView = this;

	if (!(pDoc -> m_bProgIndDlgDisplayed))
		{pDoc -> m_pProgIndDlg -> m_pDoc = pDoc;
		pDoc -> m_pProgIndDlg -> m_pLeftView = pLeftView;
```

```
            pDoc -> m_pProgIndDlg -> m_pProwin01View = NULL;
            pDoc -> m_pProgIndDlg -> Create();
            pDoc -> m_bProgIndDlgDisplayed = TRUE;
      }

}

void CLeftView::OnUpdateViewProgressionIndices(CCmdUI* pCmdUI)
{
      CProwin01Doc* pDoc = GetDocument();
      pCmdUI -> Enable(!(pDoc -> m_bProgIndDlgDisplayed));

}

void CLeftView::OnViewGrayscale()
{
      m_bShowGrayscale = m_bShowGrayscale ^ 1;
      m_NameColor = m_bShowGrayscale ? RGB(0,0,0) :
RGB(255,255,255);
      Invalidate();

}

void CLeftView::OnUpdateViewGrayscale(CCmdUI* pCmdUI)
{
      if(m_bBinocular)pCmdUI -> Enable(FALSE);
      else pCmdUI -> SetCheck(m_bShowGrayscale);

}

CBrush* CLeftView::GetGrayscaleBrush(int* sens)
{
      int i = 0;

      if (*sens ==0) i = 9;
      else if (*sens > 0 && *sens < 51) i = 8;
      else if (*sens > 50 && *sens < 101) i = 7;
      else if (*sens > 100 && *sens < 151) i = 6;
      else if (*sens > 150 && *sens < 201) i = 5;
      else if (*sens > 200 && *sens < 251) i = 4;
      else if (*sens > 250 && *sens < 301) i = 3;
      else if (*sens > 300 && *sens < 351) i = 2;
      else if (*sens > 350 && *sens < 401) i = 1;
      else if (*sens > 400) i = 0;
```

```
	return &m_GrayscaleBrush[i];

}

void CLeftView::AddToMatrix(int* point,int* sensitivity)
{
	if ( *point < 4 ) m_nMatrix[1][*point + 4] = *sensitivity;
		else if ( *point > 3 && *point < 10 )
m_nMatrix[2][*point - 1] = *sensitivity;
		else if ( *point > 9 && *point < 18 )
m_nMatrix[3][*point - 8] = *sensitivity;
		else if ( *point > 17 && *point < 28 )
m_nMatrix[4][*point - 17] = *sensitivity;
		else if ( *point > 27 && *point < 38 )
m_nMatrix[5][*point - 27] = *sensitivity;
		else if ( *point > 37 && *point < 48 )
m_nMatrix[6][*point - 37] = *sensitivity;
		else if ( *point > 47 && *point < 58 )
m_nMatrix[7][*point - 47] = *sensitivity;
		else if ( *point > 57 && *point < 66 )
m_nMatrix[8][*point - 56] = *sensitivity;
		else if ( *point > 65 && *point < 72 )
m_nMatrix[9][*point - 63] = *sensitivity;
	  else if ( *point > 71 ) m_nMatrix[10][*point - 68] =
*sensitivity;

}

void CLeftView::Interpolate(CDC* pDC)
{
	BOOL bBigCircle = FALSE;
	CRgn boundcircle;
	CRgn nasal_24_2;
	CPen* pOldPen;
	CPen GrayAxisPen(PS_SOLID,1,RGB(0,0,0));
	CPen WhiteAxisPen(PS_SOLID,9,RGB(255,255,255));
	CDC* pDisplayMemDC = new CDC;
	CBitmap* pBitmap = new CBitmap;
	CRect clientRect;
	CRect clipRect;
	int row,col,x,y,sens,nsens;
	int matrix[36][36];
```

```
      GetClientRect(clientRect);
      pDisplayMemDC -> CreateCompatibleDC(pDC);
      OnPrepareDC(pDisplayMemDC);
      pBitmap -> CreateCompatibleBitmap(pDC,clientRect.right -
clientRect.left,clientRect.bottom - clientRect.top);
      CBitmap* pOldBitmap = (CBitmap*)(pDisplayMemDC ->
SelectObject(pBitmap));
      pDisplayMemDC -> FillRect(CRect(-500,-
500,500,500),&m_WhiteBrush);

      for (row = 0; row < 36; row++)
            for (col = 0; col < 36; col++)
                  matrix[row][col] = MISSING_VALUE;

      for (row = 0; row < 12; row++)
            for (col = 0; col < 12; col++)
                  {sens = m_nMatrix[row][col];
                  matrix[row*3+1][col*3+1]  = sens;
                  if (row == 1 && sens != MISSING_VALUE)bBigCircle
= TRUE;
            }

      for (row = 4; row < 34; row+=3)
            for (col = 4; col < 34; col+=3)
                  {sens = matrix[row][col];
                  nsens = matrix[row-3][col];
                  if (sens != MISSING_VALUE)
                        {if (nsens == MISSING_VALUE)
                              {matrix[row-1][col] = sens;
                              matrix[row-2][col] = sens;
                        }
                        else matrix[row-1][col] = sens*2/3 +
nsens*1/3;
                  }

                  nsens = matrix[row+3][col];
                  if (sens != MISSING_VALUE )
                        {if (nsens == MISSING_VALUE)
                              {matrix[row+1][col] = sens;
                              matrix[row+2][col] = sens;
                        }
                        else matrix[row+1][col] = sens*2/3 +
nsens*1/3;
                  }
```

```
                nsens = matrix[row][col-3];
                if (sens != MISSING_VALUE )
                      {if (nsens == MISSING_VALUE)
                            {matrix[row][col-1] = sens;
                            matrix[row][col-2] = sens;
                      }
                      else matrix[row][col-1] = sens*2/3 +
nsens*1/3;
                }

                nsens = matrix[row][col+3];
                if (sens != MISSING_VALUE )
                      {if (nsens == MISSING_VALUE)
                            {matrix[row][col+1] = sens;
                            matrix[row][col+2] = sens;
                      }
                      else matrix[row][col+1] = sens*2/3 +
nsens*1/3;
                }

            }

      for (row = 3; row < 34; row++)
            for (col = 4; col < 34; col+=3)
                {if (row%3 == 1)break;
                sens = matrix[row][col];
                nsens = matrix[row][col-3];
                if (sens != MISSING_VALUE )
                      {if (nsens == MISSING_VALUE)
                            {matrix[row][col-1] = sens;
                            matrix[row][col-2] = sens;
                      }
                      else matrix[row][col-1] = sens*2/3 +
nsens*1/3;
                }

                nsens = matrix[row][col+3];
                if (sens != MISSING_VALUE )
                      {if (nsens == MISSING_VALUE)
                            {matrix[row][col+1] = sens;
                            matrix[row][col+2] = sens;
                      }
```

```
                        else matrix[row][col+1] = sens*2/3 +
nsens*1/3;
                  }


            }

      if (bBigCircle)boundcircle.CreateEllipticRgn(-450,-
450,450,450);
            else
                  {boundcircle.CreateEllipticRgn(-390,-
390,390,400);
                  nasal_24_2.CreateEllipticRgn(-450,-290,450,295);

      boundcircle.CombineRgn(&boundcircle,&nasal_24_2,RGN_OR);
            }


      for (row = 3; row < 33; row++)
            for (col = 3; col < 33; col++)
                  {sens = matrix[row][col];
                  if (sens == MISSING_VALUE)continue;
                  x = -540 + col*(GRAY_BOX_DIMENSION);
                  y = -540 + row*(GRAY_BOX_DIMENSION);
                  if
(boundcircle.PtInRegion(x+GRAY_BOX_DIMENSION/2,y+GRAY_BOX_DIMENSIO
N/2))
                  pDisplayMemDC -> FillRect( CRect(CPoint(x,y),

      CSize(GRAY_BOX_DIMENSION,GRAY_BOX_DIMENSION)),

      GetGrayscaleBrush(&sens));
            }


      pOldPen = pDisplayMemDC -> SelectObject(&WhiteAxisPen);
      pDisplayMemDC -> MoveTo(0,450);
      pDisplayMemDC -> LineTo(0,-450);
      pDisplayMemDC -> MoveTo(-450,0);
      pDisplayMemDC -> LineTo(450,0);
      pDisplayMemDC -> SelectObject(&GrayAxisPen);
      pDisplayMemDC -> MoveTo(0,450);
      pDisplayMemDC -> LineTo(0,-450);
      pDisplayMemDC -> MoveTo(-450,0);
```

```
      pDisplayMemDC -> LineTo(450,0);
      pDisplayMemDC -> SelectObject(pOldPen);

      pDC -> ExcludeClipRect(-500,-500,500,-445);          //name
      pDC -> ExcludeClipRect(-500,-450,-350,-410);     //eye
      pDC -> BitBlt(-500,-500,1000,1000,pDisplayMemDC,-500,-
500,SRCCOPY);
    pDisplayMemDC -> SelectObject(pOldBitmap);
      boundcircle.DeleteObject();
      nasal_24_2.DeleteObject();
      pDC -> SelectClipRgn(NULL);
      delete pDisplayMemDC;
    delete pBitmap;
}

void CLeftView::OnEyeBinocular()
{
      m_bBinocular = m_bBinocular ^ 1;
      m_NameColor = (m_bBinocular | m_bShowGrayscale) ? RGB(0,0,0)
: RGB(255,255,255);
      Invalidate();

}

void CLeftView::OnUpdateEyeBinocular(CCmdUI* pCmdUI)
{
      pCmdUI -> SetCheck(m_bBinocular);

}

void CLeftView::OnViewEstermanDefects()
{
      m_bEsterman = m_bEsterman ^ 1;
      Invalidate(FALSE);

}

void CLeftView::OnUpdateViewEstermanDefects(CCmdUI* pCmdUI)
{
      pCmdUI -> Enable(m_bBinocular);
      pCmdUI -> SetCheck(m_bEsterman);
}

void CLeftView::OnEditCopy()
```

```
{
	CDC* pDC = new CDC;
	CClientDC dc(this);
	CBitmap* pOldBitmap;
	CBitmap* pBitmap = new CBitmap;
	CRect clientRect;
	GetClientRect(clientRect);

	pDC -> CreateCompatibleDC(&dc);
	pBitmap -> CreateCompatibleBitmap(&dc,clientRect.right -
clientRect.left,clientRect.bottom - clientRect.top);
	pOldBitmap = pDC -> SelectObject(pBitmap);
	pDC -> BitBlt(0,0,1100,1100,&dc,0,0,SRCCOPY);
	pDC -> SelectObject(pOldBitmap);

	OpenClipboard();
	::EmptyClipboard();
	::SetClipboardData(CF_BITMAP,pBitmap -> Detach());

	CloseClipboard();
	pBitmap -> DeleteObject();
	delete pBitmap;
	delete pDC;
}

void CLeftView::OnEditCopyScreen()
{
	CDC* pDC = new CDC;
	CWindowDC dc(AfxGetApp() -> m_pMainWnd);
	CBitmap* pOldBitmap;
	CBitmap* pBitmap = new CBitmap;
	CRect MainWndRect;
	AfxGetApp() -> m_pMainWnd ->GetWindowRect(MainWndRect);

	pDC -> CreateCompatibleDC(&dc);
	pBitmap -> CreateCompatibleBitmap(&dc,MainWndRect.right -
MainWndRect.left,MainWndRect.bottom - MainWndRect.top);
	pOldBitmap = pDC -> SelectObject(pBitmap);
	pDC -> BitBlt(0,0,MainWndRect.right -
MainWndRect.left,MainWndRect.bottom -
MainWndRect.top,&dc,0,0,SRCCOPY);
	pDC -> SelectObject(pOldBitmap);

	OpenClipboard();
```

```
	::EmptyClipboard();
	::SetClipboardData(CF_BITMAP,pBitmap -> Detach());

	CloseClipboard();
	pBitmap -> DeleteObject();
	delete pBitmap;
	delete pDC;
}

void CLeftView::OnPrint(CDC* pDC, CPrintInfo* pInfo)
{
	CRect printRect = pInfo -> m_rectDraw;
	pDC -> LPtoDP(printRect);
	pDC -> SetMapMode(MM_ISOTROPIC);
	pDC -> SetWindowExt(1000,1000);
	pDC -> SetViewportExt(printRect.right,printRect.bottom);
	pDC -> SetViewportOrg(printRect.right/2,printRect.bottom/2);
	pDC -> DPtoLP(printRect);
	if ( !(m_bShowGrayscale | m_bBinocular) )
		{pDC -> FillRect(printRect,&m_DarkBlueBrush);
	}
	if (pInfo -> m_nCurPage == 1)
		{OnDraw(pDC);
	}
	else
		{m_bRightEye = m_bRightEye ^ 1;
		OnDraw(pDC);
		m_bRightEye = m_bRightEye ^ 1;
	}
}
```

```
// leftview.h : header file
//

/////////////////////////////////////////////////////////////////////
///////////
// CLeftView view

class CLeftView : public CScrollView
{   class CProwin01Doc;
      friend class CProgIndDlg;
private:
      BOOL m_bRightEye;
      BOOL m_bBinocular;
      BOOL m_bEsterman;
      BOOL m_bShowProgressingPoints;
      BOOL m_bShowGrayscale;
      BOOL m_bCursorInRect;
      UINT m_nFlags;
      CPoint m_mousePos;
      int m_nCount;

      DECLARE_DYNCREATE(CLeftView)
protected:
      CLeftView();                  // protected constructor used
by dynamic creation
// Data members
public:
      CBrush m_LightBlueBrush;
      CBrush m_LightRedBrush;
      CBrush m_DarkRedBrush;
      CBrush m_MagentaBrush;
      CBrush m_YellowBrush;
      CBrush m_GrayBrush;
      CBrush m_BlueGreenBrush;
      CBrush m_OliveBrush;
      CBrush m_DarkGreenBrush;
      CBrush m_LightGreenBrush;
      CBrush m_RedHatchBrush;
      CBrush m_DarkBlueBrush;
      CBrush m_WhiteBrush;
      COLORREF m_NameColor;
      CRect m_BoxArray[76];
      CRect m_tracker;
```

```
      char m_strGrayBrushName[11];
      CBrush m_GrayscaleBrush[10];
      CBitmap m_GrayscaleBitmap[10];

      int m_nMatrix[12][12];
// Attributes
public:
      CProwin01Doc* GetDocument();


// Operations
public:
      CBrush* GetBrush(int* point, int* field, CProwin01Doc*
pDoc);
      CBrush* GetGrayscaleBrush(int* sens);
      BOOL IsProgressing(int* point,int* last_field,CProwin01Doc*
pDoc,BOOL* pbRightEye);
      void AddToMatrix(int* point,int* sensitivity);
      void Interpolate(CDC* pDC);
// Implementation
protected:
#ifdef _DEBUG
      virtual void AssertValid() const;
      virtual void Dump(CDumpContext& dc) const;
#endif

      virtual ~CLeftView();
      virtual     void OnDraw(CDC* pDC);        // overridden to
draw this view
      virtual     void OnInitialUpdate();       // first time
after construct
      virtual void OnActivateView(BOOL bActivate, CView*
pActivateView, CView* pDeactiveView);
      void OnPrepareDC(CDC* pDC, CPrintInfo* pInfo = NULL);
      virtual void OnUpdate(CView* pView, LPARAM lHint, CObject*
pHint);
    // Printing support
      virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
      virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
      virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
      virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);


      // Generated message map functions
```

```
        //{{AFX_MSG(CLeftView)
        afx_msg void OnEyeRight();
        afx_msg void OnUpdateEyeRight(CCmdUI* pCmdUI);
        afx_msg void OnEyeLeft();
        afx_msg void OnUpdateEyeLeft(CCmdUI* pCmdUI);
        afx_msg void OnViewDates();
        afx_msg void OnViewLegend();
        afx_msg void OnViewProgressingPoints();
        afx_msg void OnUpdateViewProgressingPoints(CCmdUI* pCmdUI);
        afx_msg void OnViewProgCrit();
        afx_msg void OnUpdateViewLegend(CCmdUI* pCmdUI);
        afx_msg void OnUpdateViewDates(CCmdUI* pCmdUI);
        afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
        afx_msg void OnMouseMove(UINT nFlags, CPoint point);
        afx_msg BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT
message);
        afx_msg void OnFilterGaussian();
        afx_msg void OnWindowCloseall();
        afx_msg void OnUpdateFilterGaussian(CCmdUI* pCmdUI);
        afx_msg void OnViewProgressionIndices();
        afx_msg void OnUpdateViewProgressionIndices(CCmdUI* pCmdUI);
        afx_msg void OnViewGrayscale();
        afx_msg void OnUpdateViewGrayscale(CCmdUI* pCmdUI);
        afx_msg void OnEyeBinocular();
        afx_msg void OnUpdateEyeBinocular(CCmdUI* pCmdUI);
        afx_msg void OnViewEstermanDefects();
        afx_msg void OnUpdateViewEstermanDefects(CCmdUI* pCmdUI);
        afx_msg void OnEditCopy();
        afx_msg void OnEditCopyScreen();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in leftview.cpp
inline CProwin01Doc* CLeftView::GetDocument()
   { return (CProwin01Doc*)m_pDocument; }
#endif


/////////////////////////////////////////////////////////////////////
///////////
// CLeftSquareTracker class - analogous to CSquareTracker class
def in prowivw.h
```

```
class CLeftSquareTracker : public CRectTracker
{
public:
      void AdjustRect(int nHandle, LPRECT lpRect);
};
```

```
// legdlg.cpp : implementation file
//

#include "stdafx.h"
#include "prowin01.h"
#include "legdlg.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
///////////
// CLegendDialog dialog


CLegendDialog::CLegendDialog(CWnd* pParent /*=NULL*/)
      : CDialog(CLegendDialog::IDD, pParent)
{
      //{{AFX_DATA_INIT(CLegendDialog)
            // NOTE: the ClassWizard will add member
initialization here
      //}}AFX_DATA_INIT
}

void CLegendDialog::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CLegendDialog)
            // NOTE: the ClassWizard will add DDX and DDV calls
here
      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CLegendDialog, CDialog)
      //{{AFX_MSG_MAP(CLegendDialog)
      ON_WM_PAINT()
      ON_WM_CTLCOLOR()
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BOOL CLegendDialog::Create()
{     return CDialog::Create( CLegendDialog::IDD);
```

```
}
////////////////////////////////////////////////////////////////////
///////////
// CLegendDialog message handlers

void CLegendDialog::OnPaint()
{
      CPaintDC dc(this); // device context for painting

      CBrush LightGreenBrush;
      CWnd* pStatic = GetDlgItem(IDC_STATIC1);
      CDC* pDC = pStatic -> GetDC();
      pStatic -> Invalidate();
      pStatic -> UpdateWindow();
      LightGreenBrush.CreateSolidBrush(RGB(0,255,0));
      pDC-> FillRect(CRect(0,0,40,20),&LightGreenBrush);
      pStatic -> ReleaseDC(pDC);

      CBrush DarkGreenBrush;
       pStatic = GetDlgItem(IDC_STATIC2);
       pDC = pStatic -> GetDC();
      pStatic -> Invalidate();
      pStatic -> UpdateWindow();
      DarkGreenBrush.CreateSolidBrush(RGB(0,128,0));
      pDC-> FillRect(CRect(0,0,40,20),&DarkGreenBrush);
      pStatic -> ReleaseDC(pDC);

      CBrush OliveBrush;
       pStatic = GetDlgItem(IDC_STATIC3);
       pDC = pStatic -> GetDC();
      pStatic -> Invalidate();
      pStatic -> UpdateWindow();
      OliveBrush.CreateSolidBrush(RGB(128,128,0));
      pDC-> FillRect(CRect(0,0,40,20),&OliveBrush);
      pStatic -> ReleaseDC(pDC);

      CBrush BlueGreenBrush;
       pStatic = GetDlgItem(IDC_STATIC4);
       pDC = pStatic -> GetDC();
      pStatic -> Invalidate();
      pStatic -> UpdateWindow();
      BlueGreenBrush.CreateSolidBrush(RGB(0,128,128));
      pDC-> FillRect(CRect(0,0,40,20),&BlueGreenBrush);
      pStatic -> ReleaseDC(pDC);
```

```
CBrush GrayBrush;
 pStatic = GetDlgItem(IDC_STATIC5);
 pDC = pStatic -> GetDC();
pStatic -> Invalidate();
pStatic -> UpdateWindow();
GrayBrush.CreateSolidBrush(RGB(128,128,128));
pDC-> FillRect(CRect(0,0,40,20),&GrayBrush);
pStatic -> ReleaseDC(pDC);

CBrush YellowBrush;
 pStatic = GetDlgItem(IDC_STATIC6);
 pDC = pStatic -> GetDC();
pStatic -> Invalidate();
pStatic -> UpdateWindow();
YellowBrush.CreateSolidBrush(RGB(255,255,0));
pDC-> FillRect(CRect(0,0,40,20),&YellowBrush);
pStatic -> ReleaseDC(pDC);

CBrush MagentaBrush;
 pStatic = GetDlgItem(IDC_STATIC7);
 pDC = pStatic -> GetDC();
pStatic -> Invalidate();
pStatic -> UpdateWindow();
MagentaBrush.CreateSolidBrush(RGB(255,0,255));
pDC-> FillRect(CRect(0,0,40,20),&MagentaBrush);
pStatic -> ReleaseDC(pDC);

CBrush DarkRedBrush;
 pStatic = GetDlgItem(IDC_STATIC8);
 pDC = pStatic -> GetDC();
pStatic -> Invalidate();
pStatic -> UpdateWindow();
DarkRedBrush.CreateSolidBrush(RGB(128,0,0));
pDC-> FillRect(CRect(0,0,40,20),&DarkRedBrush);
pStatic -> ReleaseDC(pDC);

CBrush LightRedBrush;
 pStatic = GetDlgItem(IDC_STATIC9);
 pDC = pStatic -> GetDC();
pStatic -> Invalidate();
pStatic -> UpdateWindow();
LightRedBrush.CreateSolidBrush(RGB(255,0,0));
pDC-> FillRect(CRect(0,0,40,20),&LightRedBrush);
```

```
	pStatic -> ReleaseDC(pDC);

	CBrush LightBlueBrush;
	 pStatic = GetDlgItem(IDC_STATIC10);
	 pDC = pStatic -> GetDC();
	pStatic -> Invalidate();
	pStatic -> UpdateWindow();
	LightBlueBrush.CreateSolidBrush(RGB(0,0,255));
	pDC-> FillRect(CRect(0,0,40,20),&LightBlueBrush);
	pStatic -> ReleaseDC(pDC);

	CBrush WhiteBrush;
	 pStatic = GetDlgItem(IDC_STATIC11);
	 pDC = pStatic -> GetDC();
	pStatic -> Invalidate();
	pStatic -> UpdateWindow();
	WhiteBrush.CreateSolidBrush(RGB(255,255,255));
	pDC-> FillRect(CRect(0,0,40,20),&WhiteBrush);
	pStatic -> ReleaseDC(pDC);


	// Do not call CDialog::OnPaint() for painting messages
}

HBRUSH CLegendDialog::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT
nCtlColor)
{
	if ( nCtlColor == CTLCOLOR_DLG )
		{pDC -> SetBkColor(RGB(0,0,128));
		return m_hbrDarkBlueBrush;
	}
	if ( nCtlColor == CTLCOLOR_STATIC )
		{pDC -> SetBkColor(RGB(0,0,128));
		pDC->SetTextColor (RGB(255,255,255));
		return m_hbrDarkBlueBrush;
	}
	return CDialog::OnCtlColor( pDC,  pWnd,  nCtlColor);

}

BOOL CLegendDialog::OnInitDialog()
{
	CDialog::OnInitDialog();
```

```
	m_hbrDarkBlueBrush = CreateSolidBrush(RGB(0,0,128));

	return TRUE;  // return TRUE  unless you set the focus to a
control
}

void CLegendDialog::OnCancel()
{
	((CProwin01App*)AfxGetApp()) -> m_bLegendDialogDisplayed =
FALSE;
	 CWnd::DestroyWindow();
	 delete this;

	//CDialog::OnCancel();
}
```

```
// legdlg.h : header file
//

/////////////////////////////////////////////////////////////////////////
///////////
// CLegendDialog dialog

class CLegendDialog : public CDialog
{
// Construction
public:
      CLegendDialog(CWnd* pParent = NULL);      // standard
constructor

// Dialog Data
      //{{AFX_DATA(CLegendDialog)
      enum { IDD = IDD_LEGEND_DIALOG };
            // NOTE: the ClassWizard will add data members here
      //}}AFX_DATA

// Data members
      HBRUSH m_hbrDarkBlueBrush;

// Implementation
public:
      BOOL Create();
protected:
      virtual void DoDataExchange(CDataExchange* pDX);      //
DDX/DDV support

      // Generated message map functions
      //{{AFX_MSG(CLegendDialog)
      afx_msg void OnPaint();
      afx_msg HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT
nCtlColor);
      virtual BOOL OnInitDialog();
      virtual void OnCancel();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
```

```
// mainfrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "prowin01.h"
#include <dos.h>
#include <direct.h>
#include "mainfrm.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////
///////////
// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
      //{{AFX_MSG_MAP(CMainFrame)
      ON_WM_CREATE()
      ON_COMMAND(ID_ADMIN_ADD, OnAdminAdd)
      ON_COMMAND(ID_ADMIN_CREATE, OnAdminCreate)
      ON_COMMAND(ID_ADMIN_BULKTRANSFER, OnAdminBulkTransfer)
      ON_COMMAND(ID_ADMIN_CLEAN, OnAdminClean)
      //}}AFX_MSG_MAP
      // Global help commands
      ON_COMMAND(ID_HELP_INDEX, CMDIFrameWnd::OnHelpIndex)
      ON_COMMAND(ID_HELP_USING, CMDIFrameWnd::OnHelpUsing)
      ON_COMMAND(ID_HELP, CMDIFrameWnd::OnHelp)
      ON_COMMAND(ID_CONTEXT_HELP, CMDIFrameWnd::OnContextHelp)
      ON_COMMAND(ID_DEFAULT_HELP, CMDIFrameWnd::OnHelpIndex)
END_MESSAGE_MAP()

BEGIN_MESSAGE_MAP(CEasyToolBar, CToolBar)
      ON_WM_MOUSEMOVE()
      ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////
///////////
// arrays of IDs used to initialize control bars

// toolbar buttons - IDs are command buttons
```

```
static UINT BASED_CODE buttons[] =
{
	// same order as in the bitmap 'toolbar.bmp'
	ID_FILE_NEW,
		ID_SEPARATOR,
	ID_EDIT_COPY,
	ID_EDIT_COPY_SCREEN,
		ID_SEPARATOR,
	ID_FILE_PRINT,
		ID_SEPARATOR,
	ID_EYE_RIGHT,
	ID_EYE_LEFT,
	ID_EYE_BINOCULAR,
		ID_SEPARATOR,
	ID_VIEW_DATES,
	ID_VIEW_LEGEND,
		ID_SEPARATOR,
	ID_VIEW_PROGRESSING_POINTS,
	ID_VIEW_ESTERMAN_DEFECTS,
	ID_VIEW_PROGRESSION_INDICES,
		ID_SEPARATOR,
	ID_FILTER_GAUSSIAN,
		ID_SEPARATOR,
	ID_VIEW_GRAYSCALE,
		ID_SEPARATOR,
	ID_WINDOW_CLOSEALL,
		ID_SEPARATOR,
	ID_APP_ABOUT,
	ID_CONTEXT_HELP,
};

static UINT BASED_CODE indicators[] =
{
	ID_SEPARATOR,           // status line indicator
	ID_INDICATOR_CAPS,
	ID_INDICATOR_NUM,
	ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////
///////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
```

```
{
      m_pToolBarLabelWnd = new CFrameWnd;
}

CMainFrame::~CMainFrame()
{
      //m_pToolBarLabelWnd -> DestroyWindow(); // framework does
these
      //delete m_pToolBarLabelWnd;
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
      if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
            return -1;

      if (!m_wndToolBar.Create(this) ||
            !m_wndToolBar.LoadBitmap(IDR_MAINFRAME) ||
            !m_wndToolBar.SetButtons(buttons,
              sizeof(buttons)/sizeof(UINT)))
      {
            TRACE("Failed to create toolbar\n");
            return -1;      // fail to create
      }

      m_wndToolBar.m_pLabelWnd = m_pToolBarLabelWnd;
      m_pToolBarLabelWnd ->
Create(AfxRegisterWndClass(CS_BYTEALIGNWINDOW ,0,0,0),"Label
window", WS_BORDER | WS_POPUP | WS_DISABLED ,
            CRect(100,100,120,110),this,NULL,0,NULL);
      m_wndToolBar.m_nCurLabelIndex = 0xFFFF;
      m_wndToolBar.m_LabelFont.CreateFont(15,0,0,0,400,FALSE,FALSE
,0,ANSI_CHARSET,OUT_DEFAULT_PRECIS,CLIP_DEFAULT_PRECIS,DEFAULT_QUA
LITY,DEFAULT_PITCH | FF_SWISS, NULL);
      m_wndToolBar.m_LabelBrush.CreateSolidBrush(RGB(0,0,255));
    m_wndToolBar.m_bCaptured = FALSE;
    m_wndToolBar.m_CaptureTime = CTime::GetCurrentTime();

      if (!m_wndStatusBar.Create(this) ||
            !m_wndStatusBar.SetIndicators(indicators,
              sizeof(indicators)/sizeof(UINT)))
      {
            TRACE("Failed to create status bar\n");
            return -1;      // fail to create
```

```
	}

	return 0;
}

/////////////////////////////////////////////////////////////////////////
////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
	CMDIFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
	CMDIFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////
////////////
// CMainFrame message handlers

void CMainFrame::OnAdminAdd()
{   char buffer[MAX_INFILE_LENGTH]="";
	char legalpath[100]="";
	char filetitle[15]="";
	CStdioFile infile,outfile;
	CFileDialog dlg(TRUE,NULL,NULL,OFN_ALLOWMULTISELECT |
OFN_READONLY,"All files (*.*)|*.*||",NULL);
	char sourcefiles[FILENAME_BUFFER_SIZE]="";

	dlg.m_ofn.lpstrTitle = "Add to Database: source file(s)";
	dlg.m_ofn.lpstrFile = sourcefiles;
	dlg.m_ofn.nMaxFile = sizeof(sourcefiles);

	CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);


	if (dlg.DoModal()== IDOK)
```

```
            {if
(_fstrlen(dlg.m_ofn.lpstrFile)<MAX_MULTISELECT_STRING_LENGTH)
                  {CFileDialog
dlg1(FALSE,"txt",NULL,OFN_HIDEREADONLY | OFN_PATHMUSTEXIST |
OFN_FILEMUSTEXIST ,"ProWin databases (*.txt)|*.txt||",NULL);
                  dlg1.m_ofn.lpstrTitle = "Add to Database:
destination file";
                  if (dlg1.DoModal()==IDOK)
                        {if (outfile.Open(dlg1.m_ofn.lpstrFile,
CFile::modeReadWrite | CFile::typeText))

      {outfile.ReadString(buffer,MAX_INFILE_LENGTH);
                              outfile.SeekToEnd();
                                    if
(!_fstrcmp(buffer,COLUMN_NAMES_STRING))
                                          {do
                                                {if
(sscanf(dlg.m_ofn.lpstrFile,"%s %s",legalpath,filetitle) !=1 )
////////does what CFileDialog::GetPathName() ought to do!

      {_fstrcat(legalpath,"\\");
////////

      _fstrcat(legalpath,filetitle);
////////
                                                }

      if(infile.Open((const char*)legalpath, CFile::modeRead |
CFile::typeText))
                                                      {if
(IsHFAFile(&infile))

      {infile.Read(buffer,MAX_INFILE_LENGTH);

      DoHeader(buffer);

      DoPoints(buffer);

      outfile.WriteString(buffer);
                                                      }
                                                      else
AfxMessageBox(_fstrcat(legalpath,"\nis not a suitable HFA
file"),MB_ICONEXCLAMATION | MB_OK);
```

```
      infile.Close();
                                                  }
                                                  else
AfxMessageBox(_fstrcat(legalpath," cannot be opened"),MB_ICONSTOP
| MB_OK);

                                              }while
(GetNextFile(dlg.m_ofn.lpstrFile));
                                    }
                                    else
AfxMessageBox("Destination file is not a ProWin
database",MB_ICONSTOP | MB_OK);
                              outfile.Close();
                              pStatus->SetPaneText(0,"Finished:
files added",TRUE);
                              pStatus->UpdateWindow();

                        }
                        else AfxMessageBox("Cannot open
destination file",MB_ICONSTOP | MB_OK);

                  }
            }
            else AfxMessageBox("    Too many files
selected!\nSelect fewer or use Bulk Transfer",MB_ICONSTOP |
MB_OK);

      }

}

void CMainFrame::OnAdminCreate()
{   char buffer[MAX_INFILE_LENGTH]="";
      char legalpath[100]="";
      char filetitle[15]="";
      CStdioFile infile,outfile;
      CFileDialog dlg(TRUE,NULL,NULL,OFN_ALLOWMULTISELECT |
OFN_READONLY,"All files (*.*)|*.*||",NULL);
      char sourcefiles[FILENAME_BUFFER_SIZE]="";

      dlg.m_ofn.lpstrTitle = "Create Database: source file(s)";
      dlg.m_ofn.lpstrFile = sourcefiles;
      dlg.m_ofn.nMaxFile = sizeof(sourcefiles);
```

```
      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);

      if (dlg.DoModal()== IDOK)
            {if
(_fstrlen(dlg.m_ofn.lpstrFile)<MAX_MULTISELECT_STRING_LENGTH)
                  {CFileDialog
dlg1(FALSE,"txt","newdata.txt",OFN_HIDEREADONLY |
OFN_OVERWRITEPROMPT ,"Text files (*.txt)|*.txt||",NULL);
                  dlg1.m_ofn.lpstrTitle = "Create Database:
destination file";
                  if (dlg1.DoModal()==IDOK)
                        {if (outfile.Open(dlg1.m_ofn.lpstrFile,
CFile::modeCreate | CFile::modeWrite | CFile::typeText))

      {outfile.WriteString(COLUMN_NAMES_STRING);
                              do
                                    {if
(sscanf(dlg.m_ofn.lpstrFile,"%s %s",legalpath,filetitle) !=1 )
////////does what CFileDialog::GetPathName() ought to do!

      {_fstrcat(legalpath,"\\");
////////

      _fstrcat(legalpath,filetitle);
////////
                                    }
                                    if(infile.Open((const
char*)legalpath, CFile::modeRead | CFile::typeText))
                                          {if (IsHFAFile(&infile))

      {infile.Read(buffer,MAX_INFILE_LENGTH);
                                                DoHeader(buffer);
                                                DoPoints(buffer);

      outfile.WriteString(buffer);
                                          }
                                          else
AfxMessageBox(_fstrcat(legalpath," is not a suitable HFA
file"),MB_ICONEXCLAMATION | MB_OK);
                                          infile.Close();

                              }
```

```
                                    else
AfxMessageBox(_fstrcat(legalpath," cannot be opened"),MB_ICONSTOP
| MB_OK);

                                }while
(GetNextFile(dlg.m_ofn.lpstrFile));
                                outfile.Close();
                                pStatus->SetPaneText(0,"Finished:
database created",TRUE);
                                pStatus->UpdateWindow();

                        }
                        else AfxMessageBox("Cannot open
destination file",MB_ICONSTOP | MB_OK);

                }
            }
            else AfxMessageBox("    Too many files
selected!\nSelect fewer or use Bulk Transfer",MB_ICONSTOP |
MB_OK);

      }

}

void CMainFrame::OnAdminBulkTransfer()
{
      char *buffer;
      buffer = new char[MAX_INFILE_LENGTH];
      char start_path [100]="";           //////////will be 1st
input directory
      char end_path [100]="";             //////////will be last
input directory
      char output_filename [200]="";
      char output_filetitle [15]="";
      char error_message[500];
      int start;
      int end;
      int pathsdifferent ;

      CStdioFile *infile;
      infile = new CStdioFile;
      CStdioFile *outfile;
      outfile = new CStdioFile;
```

```
      CFileDialog start_input(TRUE,NULL,NULL,OFN_READONLY,"All
files (*.*)|*.*||",NULL);
      start_input.m_ofn.lpstrTitle = "Bulk Transfer: First input
directory";
      start_input.m_ofn.lpstrFile = start_path;
      start_input.m_ofn.nMaxFile = sizeof(start_path);

      CFileDialog end_input(TRUE,NULL,NULL,OFN_READONLY,"All files
(*.*)|*.*||",NULL);
      end_input.m_ofn.lpstrTitle = "Bulk Transfer: Last input
directory";
      end_input.m_ofn.lpstrFile = end_path;
      end_input.m_ofn.nMaxFile = sizeof(end_path);

      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);

      struct _find_t input_file;

      long int error_sum = 0L;


      if (start_input.DoModal()== IDOK)
            {*(start_path+start_input.m_ofn.nFileOffset-1)='\0';
            if (start = IsNumeric(start_path))
                  {if (end_input.DoModal()== IDOK)
                        {*(end_path+end_input.m_ofn.nFileOffset-
1)='\0';
                        if (end = IsNumeric(end_path))
                              {if (!(end < start))
                                    {if
(SameRoot(start_path,end_path))
                                          {CFileDialog
dest(FALSE,"txt",NULL,OFN_HIDEREADONLY ,"Text files
(*.txt)|*.txt||",NULL);
                                          dest.m_ofn.lpstrTitle =
"Bulk Transfer: destination file";
                                          dest.m_ofn.lpstrFile =
output_filename;

      dest.m_ofn.lpstrFileTitle = output_filetitle;
                                          if
(dest.DoModal()==IDOK)
```

```
                                                {if (outfile ->
Open(output_filename, CFile::modeReadWrite  | CFile::typeText))
                                                        {outfile ->
ReadString(buffer,MAX_INFILE_LENGTH);
                                                        outfile ->
SeekToEnd();

      //AfxMessageBox("APPENDING UNLESS OVERWRITE FOLLOWS");
                                                        if
(_fstrcmp(buffer,COLUMN_NAMES_STRING))
                                                                {if
(AfxMessageBox(_fstrcat(output_filename," is not a ProWin
database\n\nDo you wish to overwrite it?"),MB_ICONEXCLAMATION |
MB_YESNO | MB_DEFBUTTON2) == IDYES)

      {outfile -> Close();

      if (outfile -> Open(output_filename, CFile::modeCreate |
CFile::modeWrite | CFile::typeText))

      {//AfxMessageBox("OVERWRITE FROM YESNO");

      outfile -> WriteString(COLUMN_NAMES_STRING);

      }

      else AfxMessageBox("Cannot open destination file"
,MB_ICONSTOP | MB_OK);
                                                                }
                                                                else
return;
                                                        }


                                                }
                                                else  if (outfile
-> Open(output_filename, CFile::modeCreate | CFile::modeWrite |
CFile::typeText))

      {//AfxMessageBox("OVERWRITE FROM READWRITE FAILURE");

      outfile -> WriteString(COLUMN_NAMES_STRING);
                                                                }
```

```
                                                        else
AfxMessageBox("Cannot open destination file" ,MB_ICONSTOP |
MB_OK);


      //AfxMessageBox("FILE PROCESSING HERE");

      _chdrive(*start_path - 'A' + 1);

      ::SetCursor(::LoadCursor(NULL,IDC_WAIT));
                                              do

      {_chdir(start_path);
                                                    pStatus-
>SetPaneText(0,start_path,TRUE);
                                                    pStatus-
>UpdateWindow();
                                                    if
(!_dos_findfirst("*.*",_A_NORMAL,&input_file))

      {if(infile ->Open((const char*)input_file.name,
CFile::modeRead | CFile::typeText))

      {if (IsHFAFile(infile))

      {infile ->Read(buffer,MAX_INFILE_LENGTH);

      DoHeader(buffer);

      DoPoints(buffer);

      outfile -> WriteString(buffer);

      }

      else error_sum +=
StoreError(output_filetitle,start_path,input_file.name);
////////////// ERROR TRAPPING HERE FOR !IsHFAFile

      infile ->Close();


      while (!_dos_findnext(&input_file))
```

```
      {if(infile ->Open((const char*)input_file.name,
CFile::modeRead | CFile::typeText))

            {if (IsHFAFile(infile))

                  {infile ->Read(buffer,MAX_INFILE_LENGTH);

                  DoHeader(buffer);

                  DoPoints(buffer);

                  outfile -> WriteString(buffer);

            }

            else error_sum +=
StoreError(output_filetitle,start_path,input_file.name);
////////// ERROR TRAPPING HERE FOR !IsHFAFile

      }else error_sum +=
StoreError(output_filetitle,start_path,input_file.name);//////////
//////////////////ERROR TRAPPING HERE FOR infile.Open2

      infile ->Close();

      }////////////////////////////ERROR TRAPPING HERE FOR
_dos_findnext
                                                      }else
error_sum +=
StoreError(output_filetitle,start_path,input_file.name);//////////
//////////////////ERROR TRAPPING HERE FOR infile.Open1

      }////////////////////////////ERROR TRAPPING HERE FOR
_dos_findfirst

      pathsdifferent = _fstrcmp(start_path,end_path);

      IncrementDir(start_path);

      }while(pathsdifferent);
                                          outfile ->
Close();
```

```
                                                    pStatus-
>SetPaneText(0,"Finished",TRUE);
                                                    pStatus-
>UpdateWindow();
                                                          if
(error_sum)

      {sprintf(error_message,"%li conversion errors
found\nFilenames stored in
%c:\\ERRORS\\%s",error_sum,*start_path,output_filetitle);

      AfxMessageBox(error_message);
                                                          }


                                              }



                              }
                              else AfxMessageBox("First and last
input directories\n  must share the same root" ,MB_ICONSTOP |
MB_OK);
                            }
                            else AfxMessageBox("Last input
directory must not be\n    numerically less than first"
,MB_ICONSTOP | MB_OK);
                        }
                        else AfxMessageBox(_fstrcat(end_path," \n
does not terminate with a\npurely numeric nonzero
directory"),MB_ICONSTOP | MB_OK);
                  }
            }
            else AfxMessageBox(_fstrcat(start_path," \n   does not
terminate with a\npurely numeric nonzero directory"),MB_ICONSTOP |
MB_OK);
      }
      delete infile,outfile,buffer;

}
```

```
void CMainFrame::OnAdminClean()
{
      DWORD nlines,gap,i;
      LONG j;
      char buffer1[COLUMN_NAMES_STRING_LENGTH]="";
      char buffer2[LINE_LENGTH]="";
      char tmpfilename[20]="prowin.tmp";
      CTime start_time, end_time;
      CTimeSpan interval;
      CString last_unique_string = "";
      CString interval_string;
      CString infile_string;
      CStdioFile infile;
      CStdioFile tmpfile;
      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);
      CFileDialog
select_database_dlg(TRUE,"txt",NULL,OFN_HIDEREADONLY |
OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST ,"ProWin databases
(*.txt)|*.txt||",NULL);
                  select_database_dlg.m_ofn.lpstrTitle = "Database
to be cleaned";
                  if (select_database_dlg.DoModal()==IDOK)
                        {if
(infile.Open(select_database_dlg.m_ofn.lpstrFile,
CFile::modeReadWrite | CFile::typeText))

      {infile.ReadString(buffer1,COLUMN_NAMES_STRING_LENGTH);
                                    if
(!_fstrcmp(buffer1,COLUMN_NAMES_STRING))
                                          {start_time  =
CTime::GetCurrentTime();
                                          BeginWaitCursor();
                                          infile_string =
select_database_dlg.m_ofn.lpstrFile;
                                          pStatus-
>SetPaneText(0,"Sorting "+ infile_string,TRUE);
                                          pStatus->UpdateWindow();

                                          ////////////Shell
sort//////////
```

```
                                        nlines =
(infile.GetLength() - COLUMN_NAMES_STRING_LENGTH) / LINE_LENGTH;
                                         for ( gap = nlines / 2;
gap > 0; gap /= 2)
                                              for ( i = gap; i <
nlines; i++)
                                                   for ( j = i
- gap; (j >= 0) &&
(GetNameIDTestTime(&infile,j,buffer1)>GetNameIDTestTime(&infile,j+
gap,buffer2)); j -= gap)

      {infile.Seek(COLUMN_NAMES_STRING_LENGTH + j *
LINE_LENGTH,CFile::begin);

      infile.Write(buffer2, LINE_LENGTH-1);

      infile.Flush();

      infile.Seek(COLUMN_NAMES_STRING_LENGTH + (j+gap) *
LINE_LENGTH,CFile::begin);

      infile.Write( buffer1, LINE_LENGTH-1);

      infile.Flush();

                                                   }
                                        pStatus-
>SetPaneText(0,"Removing duplicates from "+ infile_string,TRUE);
                                        pStatus->UpdateWindow();

                                        ////////////Remove
duplicates////////

      tmpfile.Open(tmpfilename, CFile::modeCreate |
CFile::modeReadWrite | CFile::typeText);
                                        infile.SeekToBegin();
                                        while
(infile.ReadString(buffer1,COLUMN_NAMES_STRING_LENGTH))
                                              {if(buffer1 !=
last_unique_string)

      {tmpfile.WriteString(buffer1);
```

```
      tmpfile.Flush();

      last_unique_string = buffer1;
                                                }
                                          }

      infile.Remove(select_database_dlg.m_ofn.lpstrFile);
                                          infile.Close();
                                          tmpfile.Close();

      tmpfile.Rename(tmpfilename,select_database_dlg.m_ofn.lpstrFi
le);

                                          ////////////Get unique
identifiers//////////

                                          EndWaitCursor();
                                          end_time =
CTime::GetCurrentTime();
                                          interval  =  end_time -
start_time;
                                          interval_string =
interval.Format("    Time taken -  Hours: %H  Minutes: %M
Seconds: %S");

                                          pStatus-
>SetPaneText(0,"Finished cleaning."+interval_string,TRUE);
                                          pStatus->UpdateWindow();
                                    }
                                    else AfxMessageBox("File
selected is not a ProWin database",MB_ICONSTOP | MB_OK);

                         }
                         else {AfxMessageBox("Cannot open
database",MB_ICONSTOP | MB_OK);

                         }

                  }


}
```

```
CString CMainFrame::GetNameIDTestTime(CStdioFile* file,LONG index,
char* buffer)
{
    file -> Seek(COLUMN_NAMES_STRING_LENGTH + index * LINE_LENGTH
,CFile::begin);
    file -> Read(buffer,LINE_LENGTH -1);
    CString SortSpec(buffer + NAME_POSITION, NAME_ID_LENGTH);
    CString TestTime(buffer + TIME_POSITION,TIME_LENGTH);
    CString duration(buffer , 8);
    SortSpec += TestTime;
    SortSpec += duration;
    return  SortSpec;
}

int CMainFrame::GetNextFile(char* infiles)  ////////////////This
function removes the first file title from m_ofn.lpstrFile
{     if (_fstrchr(infiles,' ')==NULL)return 0;////////this line
for the case where user only selects 1 file
      char* end_path;
      while(*infiles++ != ' ');
      end_path = infiles;
      while(*infiles++ != ' ')
            {if (*infiles == '\0') return 0;
      }
      while(*end_path++ = *infiles++);
      return 1;
}

void CMainFrame::DoHeader(char* buffer)
{     while (*buffer++ != '*')
            {if (*buffer == ',') *buffer = ' ';
            if (*buffer == '\n') *buffer = ',';
            if (!isprint(*buffer)) *buffer = ' ';

      }
      *buffer = '\0';
}

void CMainFrame::DoPoints(char* buffer)
{     int pointno;
      int x[77];
      int y[77];
      int i=1;
```

```
int j;
float t[77];
const float missing_value = 99;
char* end_of_header;

for (j=1;j<77;j++) t[j]=missing_value;

while (*buffer++ != '*');
end_of_header = buffer;
buffer++;
while (*buffer != '$')
      {x[i] = atoi(buffer);
      buffer += 4;
      y[i] = atoi(buffer);
      buffer += 4;

      if(x[i]==-9 && y[i]==27 ) pointno=1 ;
      if(x[i]==-3 && y[i]==27 ) pointno=2 ;
      if(x[i]==3 && y[i]==27 ) pointno=3 ;
      if(x[i]==9 && y[i]==27 ) pointno=4 ;
      if(x[i]==-15 && y[i]==21 ) pointno=5 ;
      if(x[i]==-9 && y[i]==21 ) pointno=6 ;
      if(x[i]==-3 && y[i]==21 ) pointno=7 ;
      if(x[i]==3 && y[i]==21 ) pointno=8 ;
      if(x[i]==9 && y[i]==21 ) pointno=9 ;
      if(x[i]==15 && y[i]==21 ) pointno=10 ;
      if(x[i]==-21 && y[i]==15 ) pointno=11 ;
      if(x[i]==-15 && y[i]==15 ) pointno=12 ;
      if(x[i]==-9 && y[i]==15 ) pointno=13 ;
      if(x[i]==-3 && y[i]==15 ) pointno=14 ;
      if(x[i]==3 && y[i]==15 ) pointno=15 ;
      if(x[i]==9 && y[i]==15 ) pointno=16 ;
      if(x[i]==15 && y[i]==15 ) pointno=17 ;
      if(x[i]==21 && y[i]==15 ) pointno=18 ;
      if(x[i]==-27 && y[i]==9 ) pointno=19 ;
      if(x[i]==-21 && y[i]==9 ) pointno=20 ;
      if(x[i]==-15 && y[i]==9 ) pointno=21 ;
      if(x[i]==-9 && y[i]==9 ) pointno= 22;
      if(x[i]==-3 && y[i]==9 ) pointno=23 ;
      if(x[i]==3 && y[i]==9 ) pointno=24 ;
      if(x[i]==9 && y[i]==9 ) pointno=25 ;
      if(x[i]==15 && y[i]==9 ) pointno=26 ;
      if(x[i]==21 && y[i]==9 ) pointno=27 ;
      if(x[i]==27 && y[i]==9 ) pointno=28 ;
```

```
if(x[i]==-27 && y[i]==3 ) pointno=29 ;
if(x[i]==-21 && y[i]==3 ) pointno=30 ;
if(x[i]==-15 && y[i]==3 ) pointno=31 ;
if(x[i]==-9 && y[i]==3 ) pointno=32 ;
if(x[i]==-3 && y[i]==3 ) pointno=33 ;
if(x[i]==3 && y[i]==3 ) pointno=34 ;
if(x[i]==9 && y[i]==3 ) pointno=35 ;
if(x[i]==15 && y[i]==3 ) pointno=36 ;
if(x[i]==21 && y[i]==3 ) pointno=37 ;
if(x[i]==27 && y[i]==3 ) pointno=38 ;
if(x[i]==-27 && y[i]==-3 ) pointno=39 ;
if(x[i]==-21 && y[i]==-3 ) pointno=40 ;
if(x[i]==-15 && y[i]==-3 ) pointno=41 ;
if(x[i]==-9 && y[i]==-3 ) pointno=42 ;
if(x[i]==-3 && y[i]==-3 ) pointno=43 ;
if(x[i]==3 && y[i]==-3 ) pointno=44 ;
if(x[i]==9 && y[i]==-3 ) pointno=45 ;
if(x[i]==15 && y[i]==-3 ) pointno=46 ;
if(x[i]==21 && y[i]==-3 ) pointno=47 ;
if(x[i]==27 && y[i]==-3 ) pointno=48 ;
if(x[i]==-27 && y[i]==-9 ) pointno=49 ;
if(x[i]==-21 && y[i]==-9 ) pointno=50 ;
if(x[i]==-15 && y[i]==-9 ) pointno=51 ;
if(x[i]==-9 && y[i]==-9 ) pointno=52 ;
if(x[i]==-3 && y[i]==-9 ) pointno=53 ;
if(x[i]==3 && y[i]==-9 ) pointno=54 ;
if(x[i]==9 && y[i]==-9 ) pointno=55 ;
if(x[i]==15 && y[i]==-9 ) pointno=56 ;
if(x[i]==21 && y[i]==-9 ) pointno=57 ;
if(x[i]==27 && y[i]==-9 ) pointno=58 ;
if(x[i]==-21 && y[i]==-15 ) pointno=59 ;
if(x[i]==-15 && y[i]==-15 ) pointno=60 ;
if(x[i]==-9 && y[i]==-15 ) pointno=61 ;
if(x[i]==-3 && y[i]==-15 ) pointno=62 ;
if(x[i]==3 && y[i]==-15 ) pointno=63 ;
if(x[i]==9 && y[i]==-15 ) pointno=64 ;
if(x[i]==15 && y[i]==-15 ) pointno=65 ;
if(x[i]==21 && y[i]==-15 ) pointno=66 ;
if(x[i]==-15 && y[i]==-21 ) pointno=67 ;
if(x[i]==-9 && y[i]==-21 ) pointno=68 ;
if(x[i]==-3 && y[i]==-21 ) pointno=69 ;
if(x[i]==3 && y[i]==-21 ) pointno=70 ;
if(x[i]==9 && y[i]==-21 ) pointno=71 ;
if(x[i]==15 && y[i]==-21 ) pointno=72 ;
```

```
            if(x[i]==-9 && y[i]==-27 ) pointno=73 ;
            if(x[i]==-3 && y[i]==-27 ) pointno=74 ;
            if(x[i]==3 && y[i]==-27 ) pointno=75 ;
            if(x[i]==9 && y[i]==-27 ) pointno=76 ;

            t[pointno] = (float)atoi(buffer);
            buffer += 3;
            if (*(buffer+1) != ' ')
t[pointno]=(t[pointno]+atoi(buffer))/2;
            while (*buffer++ != '\n');
            i++;

      }
      buffer = end_of_header;
      for (i=1;i<77;i++)
            {j=sprintf(buffer,",%4.1f",t[i]);
            buffer += j;
      }
      *buffer++ = '\n';
      *buffer = '\0';

}

int CMainFrame::IsHFAFile(CStdioFile* infile )
{     char buffer;
      DWORD filelength = infile->GetLength();
      if ((filelength == LENGTH_24_2) || (filelength ==
LENGTH_30_2))
            {infile->Seek(2L,CFile::begin);
            infile->Read(&buffer,1);
            infile->SeekToBegin();
            if (buffer == ':')
                  {infile->Seek(5L,CFile::begin);
                  infile->Read(&buffer,1);
                  infile->SeekToBegin();
                  if (buffer == ':')
                        {return 1;
                  }
                  else return 0;
            }
            else return 0;
      }
      else return 0;
}
```

```
int CMainFrame::IsNumeric(char* buffer)
{     while (*buffer++);
      while (*--buffer != '\\');
      return atoi(buffer+1);
}

int CMainFrame::SameRoot(char* a, char* b)
{     char* runner = a;
      while (*runner++);
      while (*--runner != '\\');
      return !_fstrncmp(a,b,runner-a);
}

void CMainFrame::IncrementDir(char* start_path)
{     int taildir;
      char newtaildir[4];
      newtaildir[3] = '\0';
      char* runner = start_path;
      while (*runner++);
      while (*--runner != '\\');
      runner++;
      taildir = atoi(runner);
      *runner = '\0';
      taildir++;
      _itoa(taildir,newtaildir,10);
      if (_fstrlen(newtaildir)==1)
                  {newtaildir[2]=newtaildir[0];
                  newtaildir[1]='0';
                  newtaildir[0]='0';
      }
      if (_fstrlen(newtaildir)==2)
                  {newtaildir[2]=newtaildir[1];
                  newtaildir[1]=newtaildir[0];
                  newtaildir[0]='0';
      }
      _fstrcat(start_path,newtaildir);
}

LONG CMainFrame::StoreError(char* output_filetitle,char*
path,char* file)
{     CStdioFile errorfile;
      char cwd[100];
      char fullpath[100];
```

```
	_getcwd(cwd,100);
	_fstrcpy(fullpath,path);
	_fstrcat(fullpath,"\\");
	_fstrcat(fullpath,file);
	_fstrcat(fullpath,"\n");
	_mkdir("\\errors");
	_chdir("\\errors");
	if(errorfile.Open(output_filetitle, CFile::modeReadWrite  |
CFile::typeText))
	errorfile.SeekToEnd();
	else errorfile.Open(output_filetitle,CFile::modeCreate |
CFile::modeWrite | CFile::typeText);
	errorfile.WriteString(fullpath);
	_chdir(cwd);
	return 1L;

}

void CEasyToolBar::OnMouseMove(UINT nFlags, CPoint point)
{
	CWindowDC dc(m_pLabelWnd);
	CRect buttonRect, wndRect;
	CFont* pOldFont = dc.SelectObject(&m_LabelFont);
	int nLabelWidth;
	UINT nCommandID;
	CString strLabel, strStatusMessage;
	CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);
	CTimeSpan captureInterval;

	GetWindowRect(wndRect);

	ClientToScreen(&point);

	if (GetCapture() != this)
		{SetCapture();
		m_bCaptured = TRUE;
		m_CaptureTime = CTime::GetCurrentTime();
	}

	captureInterval = CTime::GetCurrentTime() - m_CaptureTime;

	if (captureInterval.GetTotalSeconds() < 1 )
		{PostMessage(WM_MOUSEMOVE);
```

```
            m_nCurLabelIndex = 0xFFFF;
            return;
      }

      dc.SetTextColor(RGB(255,255,0));
      dc.SetBkColor(RGB(0,0,255));

      for (int i = 0; i<26 ; i++)
            {GetItemRect(i,buttonRect);
            buttonRect.InflateRect(-1,0);
            ClientToScreen(buttonRect);
            if (buttonRect.PtInRect(point) && i !=
m_nCurLabelIndex)
                  {nCommandID = GetItemID(i);
                  switch (nCommandID)
                        {case (ID_SEPARATOR):
                              {strLabel = "";
                              nLabelWidth = 0;
                              break;
                        }
                        case (ID_FILE_NEW):
                              {strLabel = "Open database";
                              nLabelWidth = 95;
                              break;
                        }
                        case (ID_FILE_OPEN):
                              {strLabel = "Open file";
                              nLabelWidth = 65;
                              break;
                        }
                        case (ID_FILE_SAVE):
                              {strLabel = "Save";
                              nLabelWidth = 40;
                              break;
                        }
                        case (ID_EDIT_COPY):
                              {strLabel = "Copy pane";
                              nLabelWidth = 70;
                              break;
                        }
                        case (ID_EDIT_COPY_SCREEN):
                              {strLabel = "Copy screen";
                              nLabelWidth = 80;
                              break;
```

```
}
case (ID_FILE_PRINT):
      {strLabel = "Print";
      nLabelWidth = 40;
      break;
}
case (ID_EYE_RIGHT):
      {strLabel = "Right eye";
      nLabelWidth = 60;
      break;
}
case (ID_EYE_LEFT):
      {strLabel = "Left eye";
      nLabelWidth = 55;
      break;
}
case (ID_EYE_BINOCULAR):
      {strLabel = "Binocular";
      nLabelWidth = 65;
      break;
}
case (ID_VIEW_DATES):
      {strLabel = "Dates";
      nLabelWidth = 45;
      break;
}
case (ID_VIEW_LEGEND):
      {strLabel = "Legend";
      nLabelWidth = 55;
      break;
}
case (ID_VIEW_PROGRESSING_POINTS):
      {strLabel = "Progressing points";
      nLabelWidth = 115;
      break;
}
case (ID_VIEW_ESTERMAN_DEFECTS):
      {strLabel = "Esterman defects";
      nLabelWidth = 110;
      break;
}
case (ID_VIEW_PROGRESSION_INDICES):
      {strLabel = "Indices";
      nLabelWidth = 50;
```

```
                        break;
                    }
                    case (ID_FILTER_GAUSSIAN):
                        {strLabel = "Gaussian filter";
                        nLabelWidth = 90;
                        break;
                    }
                    case (ID_VIEW_GRAYSCALE):
                        {strLabel = "Grayscale";
                        nLabelWidth = 65;
                        break;
                    }
                    case (ID_WINDOW_CLOSEALL):
                        {strLabel = "Close all";
                        nLabelWidth = 60;
                        break;
                    }
                    case (ID_APP_ABOUT):
                        {strLabel = "About";
                        nLabelWidth = 45;
                        break;
                    }
                    case (ID_CONTEXT_HELP):
                        {strLabel = "Help";
                        nLabelWidth = 40;
                        break;
                    }
                }
                m_pLabelWnd -
>MoveWindow(point.x+10,point.y+20,nLabelWidth,25);
                m_pLabelWnd ->ShowWindow(SW_SHOWNOACTIVATE);
                dc.FillRect(CRect(1,1,nLabelWidth-
1,24),&m_LabelBrush);
                dc.TextOut(5,5,strLabel);
                AfxFormatString1(strStatusMessage,nCommandID ==
ID_SEPARATOR ? AFX_IDS_IDLEMESSAGE : nCommandID,"");
                if (nCommandID == ID_SEPARATOR)pStatus-
>SetPaneText(0,strStatusMessage,TRUE);
                else pStatus->SetPaneText(0,"Button function:
"+strStatusMessage,TRUE);
                pStatus->UpdateWindow();
                m_nCurLabelIndex = i;
            }
```

```
        }

        if (!wndRect.PtInRect(point))
                {ReleaseCapture();
                m_pLabelWnd ->ShowWindow(SW_HIDE);

        AfxFormatString1(strStatusMessage,AFX_IDS_IDLEMESSAGE,"");
                pStatus->SetPaneText(0,strStatusMessage,TRUE);
                pStatus->UpdateWindow();
                m_nCurLabelIndex = 0xFFFF;
                m_bCaptured = FALSE;
        }

        dc.SelectObject(pOldFont);
        CWnd::OnMouseMove(nFlags, point);
}


void CEasyToolBar::OnLButtonDown(UINT nFlags, CPoint point)
{
        m_pLabelWnd -> ShowWindow(SW_HIDE);
        CToolBar::OnLButtonDown(nFlags, point);
}
```

```
// mainfrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////
////////////
#include <ctype.h>
#include<direct.h>
#define MAX_MULTISELECT_STRING_LENGTH 1900
#define FILENAME_BUFFER_SIZE 3000
#define MAX_INFILE_LENGTH 3000
#define LENGTH_30_2 2148
#define LENGTH_24_2 1620
#define COLUMN_NAMES_STRING
"\"Duration\",\"DOB\",\"TestDate\",\"TestTime\",\"RefDate\",\"name
\",\"ID\",\
\"Pupil\",\"TestName\",\"Rx\",\"StimHue\",\"StimSize\",\"FixnTarg\
",\"BSSize\",\"StimInt\",\"ThrStrat\",\
\"ScrStrat\",\"BckField\",\"AcSuStIn\",\"FoThTeSt\",\"Flucts\",\"F
NegPres\",\"FNegErr\",\"FPosPres\",\"FPosErr\",\
\"QsAsked\",\"FixnLoss\",\"ToFiPoSh\",\"Eye\",\"RelDefs\",\"PtsSee
n\",\"AbsDefs\",\"FileType\",\"TestCode\",\
\"TestType\",\"BSxCoord\",\"BSyCoord\",\"BSFlag\",\"BSFndFlg\",\"S
crLvInc\",\"VA\",\"SoftVers\",\"ExFoThr1\",\
\"ThrSpcng\",\"ArcStep\",\"ArcStart\",\"ArcRad\",\"ProMerid\",\"Me
asFoTh\",\"ExFoThr2\",\"ArcEnd\",\"NoOut\",\
\"FileConv\",\"EndInfo\",\"Point1\",\"Point2\",\"Point3\",\"Point4
\",\"Point5\",\"Point6\",\"Point7\",\"Point8\",\
\"Point9\",\"Point10\",\"Point11\",\"Point12\",\"Point13\",\"Point
14\",\"Point15\",\"Point16\",\"Point17\",\
\"Point18\",\"Point19\",\"Point20\",\"Point21\",\"Point22\",\"Poin
t23\",\"Point24\",\"Point25\",\"Point26\",\
\"Point27\",\"Point28\",\"Point29\",\"Point30\",\"Point31\",\"Poin
t32\",\"Point33\",\"Point34\",\"Point35\",\
\"Point36\",\"Point37\",\"Point38\",\"Point39\",\"Point40\",\"Poin
t41\",\"Point42\",\"Point43\",\"Point44\",\
\"Point45\",\"Point46\",\"Point47\",\"Point48\",\"Point49\",\"Poin
t50\",\"Point51\",\"Point52\",\"Point53\",\
\"Point54\",\"Point55\",\"Point56\",\"Point57\",\"Point58\",\"Poin
t59\",\"Point60\",\"Point61\",\"Point62\",\
\"Point63\",\"Point64\",\"Point65\",\"Point66\",\"Point67\",\"Poin
t68\",\"Point69\",\"Point70\",\"Point71\",\
\"Point72\",\"Point73\",\"Point74\",\"Point75\",\"Point76\"\n"
#define COLUMN_NAMES_STRING_LENGTH 1287   // these both include
#define LINE_LENGTH 648                   // the terminal CR and
LF
```

```
#define NAME_ID_LENGTH 35
#define NAME_LENGTH 24
#define NAME_POSITION 45
#define TIME_LENGTH 17
#define TIME_POSITION 18

class CEasyToolBar : public CToolBar
{
public:
      CFrameWnd* m_pLabelWnd;
      int m_nCurLabelIndex;
      CFont m_LabelFont;
      CBrush m_LabelBrush;
      BOOL m_bCaptured;
      CTime m_CaptureTime;
protected:
      //{{AFX_MSG(CEasyToolBar)
      afx_msg void OnMouseMove(UINT nFlags, CPoint point);
      afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()

};

class CMainFrame : public CMDIFrameWnd
{
      DECLARE_DYNAMIC(CMainFrame)
public:
      CMainFrame();

// Attributes
public:
      CFrameWnd* m_pToolBarLabelWnd; //label for toolbar

// Operations
public:
      int GetNextFile(char* infiles);  ////////////////This
function removes the first file title from m_ofn.lpstrFile
      void DoHeader(char* buffer);
      void DoPoints(char* buffer);
      int IsHFAFile(CStdioFile* infile );
      int IsNumeric(char* buffer);
      int SameRoot(char* a, char* b);
      void IncrementDir(char* start_path);
```

```
      LONG StoreError(char* output_filetitle,char* path,char*
file);
      CString GetNameIDTestTime(CStdioFile* file,LONG index, char*
buffer);


// Implementation
public:
      virtual ~CMainFrame();
#ifdef _DEBUG
      virtual void AssertValid() const;
      virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
      CStatusBar  m_wndStatusBar;
      CEasyToolBar    m_wndToolBar;

// Generated message map functions
protected:
      //{{AFX_MSG(CMainFrame)
      afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
      afx_msg void OnAdminAdd();
      afx_msg void OnAdminCreate();
      afx_msg void OnAdminBulkTransfer();
      afx_msg void OnAdminClean();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////
///////////
```

```
// progcrit.cpp : implementation file
//

#include "stdafx.h"
#include "prowin01.h"
#include "progcrit.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
///////////
// CProgCritDialog dialog


CProgCritDialog::CProgCritDialog(CWnd* pParent /*=NULL*/)
      : CDialog(CProgCritDialog::IDD, pParent)
{
      //{{AFX_DATA_INIT(CProgCritDialog)
      m_nEdgeP = -1;
      m_strEdgeSlope = "";
      m_nInnerP = -1;
      m_strInnerSlope = "";
      //}}AFX_DATA_INIT
}

void CProgCritDialog::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CProgCritDialog)
      DDX_Radio(pDX, IDC_EDGE_P, m_nEdgeP);
      DDX_CBString(pDX, IDC_EDGE_SLOPE, m_strEdgeSlope);
      DDV_MaxChars(pDX, m_strEdgeSlope, 4);
      DDX_Radio(pDX, IDC_INNER_P, m_nInnerP);
      DDX_CBString(pDX, IDC_INNER_SLOPE, m_strInnerSlope);
      DDV_MaxChars(pDX, m_strInnerSlope, 4);
      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CProgCritDialog, CDialog)
      //{{AFX_MSG_MAP(CProgCritDialog)
      ON_BN_CLICKED(IDC_SET_DEFAULT, OnSetDefault)
```

```
      ON_BN_CLICKED(IDOK, OnClickedOK)
      ON_BN_CLICKED(IDC_HELP, OnClickedHelp)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////
///////////
// CProgCritDialog message handlers

void CProgCritDialog::OnSetDefault()
{
      char message[500];
      CString innerp;
      CString edgep;
      OnClickedOK();
      if (m_nInnerP == 0)innerp = "p >= 0.1";
      if (m_nInnerP == 1)innerp = "p < 0.1 ";
      if (m_nInnerP == 2)innerp = "p < 0.05";
      if (m_nInnerP == 3)innerp = "p < 0.01";
      if (m_nEdgeP == 0)edgep = "p >= 0.1";
      if (m_nEdgeP == 1)edgep = "p < 0.1 ";
      if (m_nEdgeP == 2)edgep = "p < 0.05";
      if (m_nEdgeP == 3)edgep = "p < 0.01";
    wsprintf(message,"Progression criteria have been set
as:\n\n\tInner points\t\tEdge points\n\nSlope\t %s dB / yr\t\t %s
dB / yr\np value\t %s \t\t %s \n\n"
    "Do you really want these as the default progression
criteria?",
    (const char*)m_strInnerSlope,(const
char*)m_strEdgeSlope,(const char*)innerp,(const char*)edgep);
    if (AfxMessageBox(message, MB_YESNO | MB_ICONQUESTION |
MB_DEFBUTTON2) == IDYES)
      {AfxGetApp() -> WriteProfileString("Progression
criteria","InnerSlope",m_strInnerSlope);
      AfxGetApp() -> WriteProfileString("Progression
criteria","EdgeSlope",m_strEdgeSlope);
      AfxGetApp() -> WriteProfileInt("Progression
criteria","InnerP",m_nInnerP);
      AfxGetApp() -> WriteProfileInt("Progression
criteria","EdgeP",m_nEdgeP);
    }
    else CDialog::DoModal();
}
```

```
BOOL CProgCritDialog::OnInitDialog()
{

      m_strInnerSlope = ((CProwin01App*)AfxGetApp()) ->
m_strMinInnerSlope;
      m_strEdgeSlope = ((CProwin01App*)AfxGetApp()) ->
m_strMinEdgeSlope;
      m_nInnerP = ((CProwin01App*)AfxGetApp()) -> m_nInnerPValue;
      m_nEdgeP = ((CProwin01App*)AfxGetApp()) -> m_nEdgePValue;


      return CDialog::OnInitDialog();  // return TRUE  unless you
set the focus to a control
}

void CProgCritDialog::OnClickedOK()
{
      CDialog::OnOK();
      ((CProwin01App*)AfxGetApp()) -> m_strMinInnerSlope =
m_strInnerSlope;
      ((CProwin01App*)AfxGetApp()) -> m_strMinEdgeSlope =
m_strEdgeSlope;
      ((CProwin01App*)AfxGetApp()) -> m_nInnerPValue = m_nInnerP;
      ((CProwin01App*)AfxGetApp()) -> m_nEdgePValue = m_nEdgeP;

}

void CProgCritDialog::OnClickedHelp()
{
      OnHelp();

}
```

```
// progcrit.h : header file
//

/////////////////////////////////////////////////////////////////////////
////////////
// CProgCritDialog dialog

class CProgCritDialog : public CDialog
{
// Construction
public:
      CProgCritDialog(CWnd* pParent = NULL);    // standard
constructor

// Dialog Data
      //{{AFX_DATA(CProgCritDialog)
      enum { IDD = IDD_PROG_CRIT_DIALOG };
      int         m_nEdgeP;
      CString     m_strEdgeSlope;
      int         m_nInnerP;
      CString     m_strInnerSlope;
      //}}AFX_DATA

// Implementation
protected:
      virtual void DoDataExchange(CDataExchange* pDX);      //
DDX/DDV support

      // Generated message map functions
      //{{AFX_MSG(CProgCritDialog)
      afx_msg void OnSetDefault();
      virtual BOOL OnInitDialog();
      afx_msg void OnClickedOK();
      afx_msg void OnClickedHelp();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
```

```
// progind.cpp : implementation file
//

#include "stdafx.h"
#include "prowin01.h"
#include "progind.h"
#include "prowidoc.h"
#include "prowivw.h"
#include "leftview.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////
///////////
// CProgIndDlg dialog


CProgIndDlg::CProgIndDlg(CWnd* pParent /*=NULL*/)
      : CDialog(CProgIndDlg::IDD, pParent)
{
      //{{AFX_DATA_INIT(CProgIndDlg)
            // NOTE: the ClassWizard will add member
initialization here
      //}}AFX_DATA_INIT
}

void CProgIndDlg::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CProgIndDlg)
            // NOTE: the ClassWizard will add DDX and DDV calls
here
      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CProgIndDlg, CDialog)
      //{{AFX_MSG_MAP(CProgIndDlg)
      ON_WM_PAINT()
      ON_BN_CLICKED(IDC_HELP, OnClickedHelp)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
BOOL CProgIndDlg::Create()
{     return CDialog::Create( CProgIndDlg::IDD);
}

////////////////////////////////////////////////////////////////////
///////////
// CProgIndDlg message handlers

void CProgIndDlg::OnCancel()
{
      m_pDoc -> m_bProgIndDlgDisplayed = FALSE;
      CWnd::DestroyWindow();

      //CDialog::OnCancel();
}

BOOL CProgIndDlg::OnInitDialog()
{
      int leftNumber = 0;
      int rightNumber = 0;
      int leftNumberProgressing = 0;
      int rightNumberProgressing = 0;
      int point;
      int left_field = (m_pDoc -> m_nLeftCount) - 1;
      int right_field = (m_pDoc -> m_nRightCount) - 1;
      BOOL leftMeanSlopeWholeValid = FALSE;
      BOOL leftMeanSlopeProgValid = FALSE;
      BOOL rightMeanSlopeWholeValid = FALSE;
      BOOL rightMeanSlopeProgValid = FALSE;
      BOOL rightBackProjYearValid = FALSE;
      BOOL leftBackProjYearValid = FALSE;
      long double slope;
      long double leftMeanSlopeWhole = 0;
      long double leftMeanSlopeProg = 0;
      long double rightMeanSlopeWhole = 0;
      long double rightMeanSlopeProg = 0;
      WORD sensitivity;
      WORD intercept;
      long double dwBackProjYear;
      long double dwRightBackProjYear = 0xFFFFFFFF;
      long double dwLeftBackProjYear = 0xFFFFFFFF;
      CString slopeString;
      char text[50];
```

```
      BOOL bRightEye;

      SetWindowText("PROGRESSION INDICES: " + m_pDoc -> m_name);

      for ( point=0; point<76; point++)
            {bRightEye = TRUE;
            if ( m_pDoc -> m_rightorderArray.GetSize())
                  {slopeString = (CString)m_pDoc ->
m_rightslopeArray.GetAt(point + 76*m_pDoc ->
m_rightorderArray.GetAt(right_field));
                  sensitivity = (WORD)m_pDoc ->
m_rightpointArray.GetAt(point + 76*m_pDoc ->
m_rightorderArray.GetAt(right_field));
                  if (sensitivity != MISSING_VALUE )
                        {if (!slopeString.IsEmpty())
                              {rightMeanSlopeWholeValid = TRUE;
                              slope = _atold((const
char*)slopeString);
                              /*if ( slope < -1 && sensitivity >
50)
                                    {rightBackProjYearValid =
TRUE;
                                    intercept = (WORD)m_pDoc ->
m_rightinterceptArray.GetAt(point + 76*m_pDoc ->

      m_rightorderArray.GetAt(right_field));
                                    dwBackProjYear = (30 -
intercept/10.0)/slope;
                                    if ( dwBackProjYear <
dwRightBackProjYear )  dwRightBackProjYear = dwBackProjYear;
                                    TRACE("\n\tslope =
%.2Lf\n\tintercept = %d\n\tdwBackProjYear =
%.2Lf\n\tdwRightBackProjYear = %.2Lf\n",

      slope,intercept,dwBackProjYear,dwRightBackProjYear);
                              }*/
                              rightNumber++;
                              rightMeanSlopeWhole += slope;
                              if ( m_pProwin01View )
                                    if ( m_pProwin01View  ->
IsProgressing(&point,&right_field,m_pDoc,&bRightEye))
                                          {rightMeanSlopeProgValid
= TRUE;
```

```
      rightNumberProgressing++;
                                          rightMeanSlopeProg +=
slope;
                                          if (sensitivity > 100 )
// to avoid points below 'dynamic range'

      {rightBackProjYearValid = TRUE;
                                                intercept =
(WORD)m_pDoc -> m_rightinterceptArray.GetAt(point + 76*m_pDoc ->

      m_rightorderArray.GetAt(right_field));
                                                dwBackProjYear =
(30 - intercept/10.0)/slope;
                                                if (
dwBackProjYear < dwRightBackProjYear )  dwRightBackProjYear =
dwBackProjYear;
                                          }
                                    }
                              if ( m_pLeftView )
                                    if ( m_pLeftView  ->
IsProgressing(&point,&right_field,m_pDoc,&bRightEye))
                                          {rightMeanSlopeProgValid
= TRUE;

      rightNumberProgressing++;
                                          rightMeanSlopeProg +=
slope;
                                          if (sensitivity > 100 )
// to avoid points below 'dynamic range'

      {rightBackProjYearValid = TRUE;
                                                intercept =
(WORD)m_pDoc -> m_rightinterceptArray.GetAt(point + 76*m_pDoc ->

      m_rightorderArray.GetAt(right_field));
                                                dwBackProjYear =
(30 - intercept/10.0)/slope;
                                                if (
dwBackProjYear < dwRightBackProjYear )  dwRightBackProjYear =
dwBackProjYear;
                                          }
                                    }
                        }
```

```
                }
            }
            bRightEye = FALSE;
            if ( m_pDoc -> m_leftorderArray.GetSize())
                {slopeString = (CString)m_pDoc ->
m_leftslopeArray.GetAt(point + 76*m_pDoc ->
m_leftorderArray.GetAt(left_field));
                sensitivity = (WORD)m_pDoc ->
m_leftpointArray.GetAt(point + 76*m_pDoc ->
m_leftorderArray.GetAt(left_field));
                if (sensitivity != MISSING_VALUE )
                    {if (!slopeString.IsEmpty())
                        {leftMeanSlopeWholeValid = TRUE;
                        slope = _atold((const
char*)slopeString);
                        /*if ( slope < -1 && sensitivity >
50)
                            {leftBackProjYearValid = TRUE;
                            intercept = (WORD)m_pDoc ->
m_leftinterceptArray.GetAt(point + 76*m_pDoc ->

      m_leftorderArray.GetAt(left_field));
                            dwBackProjYear = (30 -
intercept/10.0)/slope;
                            if ( dwBackProjYear <
dwLeftBackProjYear )  dwLeftBackProjYear = dwBackProjYear;
                        }*/
                        leftNumber++;
                        leftMeanSlopeWhole += slope;
                        if ( m_pProwin01View )
                            if ( m_pProwin01View  ->
IsProgressing(&point,&left_field,m_pDoc,&bRightEye))
                                {leftMeanSlopeProgValid
= TRUE;
                                leftNumberProgressing++;
                                leftMeanSlopeProg +=
slope;
                                if (sensitivity > 100 )
// to avoid points below 'dynamic range'

      {leftBackProjYearValid = TRUE;
                                    intercept =
(WORD)m_pDoc -> m_leftinterceptArray.GetAt(point + 76*m_pDoc ->
```

```
      m_leftorderArray.GetAt(left_field));
                                                    dwBackProjYear =
(30 - intercept/10.0)/slope;
                                                    if (
dwBackProjYear < dwLeftBackProjYear )  dwLeftBackProjYear =
dwBackProjYear;
                                                }
                                            }
                                    if ( m_pLeftView )
                                            if ( m_pLeftView  ->
IsProgressing(&point,&left_field,m_pDoc,&bRightEye))
                                                {leftMeanSlopeProgValid
= TRUE;
                                                leftNumberProgressing++;
                                                leftMeanSlopeProg +=
slope;
                                                if (sensitivity > 100 )
// to avoid points below 'dynamic range'

      {leftBackProjYearValid = TRUE;
                                                    intercept =
(WORD)m_pDoc -> m_leftinterceptArray.GetAt(point + 76*m_pDoc ->

      m_leftorderArray.GetAt(left_field));
                                                    dwBackProjYear =
(30 - intercept/10.0)/slope;
                                                    if (
dwBackProjYear < dwLeftBackProjYear )  dwLeftBackProjYear =
dwBackProjYear;
                                                }
                                            }
                                    }
                            }
                    }
      }


      SetDlgItemText(IDC_EDIT_LEFT_NO,
_itoa(leftNumberProgressing, text,10));
      SetDlgItemText(IDC_EDIT_RIGHT_NO,
_itoa(rightNumberProgressing, text,10));

      if(leftMeanSlopeWholeValid)
```

```
        {leftMeanSlopeWhole /= leftNumber;
        sprintf(text,"%.2Lf",leftMeanSlopeWhole);
      SetDlgItemText(IDC_EDIT_LEFT_MS_WHOLE, text);
    }
    else SetDlgItemText(IDC_EDIT_LEFT_MS_WHOLE, "Undefined");

    if(leftMeanSlopeProgValid)
        {leftMeanSlopeProg /= leftNumberProgressing;
        sprintf(text,"%.2Lf",leftMeanSlopeProg);
      SetDlgItemText(IDC_EDIT_LEFT_MS_PROG, text);
    }
    else SetDlgItemText(IDC_EDIT_LEFT_MS_PROG, "Undefined");

    if(rightMeanSlopeWholeValid)
        {rightMeanSlopeWhole /= rightNumber;
        sprintf(text,"%.2Lf",rightMeanSlopeWhole);
      SetDlgItemText(IDC_EDIT_RIGHT_MS_WHOLE, text);
    }
    else SetDlgItemText(IDC_EDIT_RIGHT_MS_WHOLE, "Undefined");

    if(rightMeanSlopeProgValid)
        {rightMeanSlopeProg /= rightNumberProgressing;
        sprintf(text,"%.2Lf",rightMeanSlopeProg);
      SetDlgItemText(IDC_EDIT_RIGHT_MS_PROG, text);
    }
    else SetDlgItemText(IDC_EDIT_RIGHT_MS_PROG, "Undefined");

    if(rightBackProjYearValid)
        {dwRightBackProjYear += 1980;
        sprintf(text,"%.0Lf",dwRightBackProjYear);
        SetDlgItemText(IDC_EDIT_RIGHT_BACKPROJ, text);
    }
    else SetDlgItemText(IDC_EDIT_RIGHT_BACKPROJ, "Undefined");

    if(leftBackProjYearValid)
        {dwLeftBackProjYear += 1980;
        sprintf(text,"%.0Lf",dwLeftBackProjYear);
        SetDlgItemText(IDC_EDIT_LEFT_BACKPROJ, text);
    }
    else SetDlgItemText(IDC_EDIT_LEFT_BACKPROJ, "Undefined");

    return TRUE;  // return TRUE  unless you set the focus to a
control
}
```

```
void CProgIndDlg::OnPaint()
{
      CPaintDC dc(this); // device context for painting

      if (this -> IsIconic())
            {dc.SetMapMode(MM_TEXT);
            dc.DrawIcon(2,2,(HICON)AfxGetApp()-
>LoadIcon(IDI_PROGRESSION_INDICES));
      }

      // Do not call CDialog::OnPaint() for painting messages
}

void CProgIndDlg::OnClickedHelp()
{
      OnHelp();

}
```

```
// progind.h : header file
//

/////////////////////////////////////////////////////////////////////
///////////
// CProgIndDlg dialog

class CProgIndDlg : public CDialog
{     friend class CProwin01Doc;
      friend class CProwin01View;
      friend class CDateDialog;
      friend class CLeftView;
// Construction
public:
      CProgIndDlg(CWnd* pParent = NULL);  // standard constructor

// Data members
      CProwin01Doc* m_pDoc;
      CProwin01View* m_pProwin01View;
      CLeftView* m_pLeftView;
// Dialog Data
      //{{AFX_DATA(CProgIndDlg)
      enum { IDD = IDD_PROGRESSION_INDICES_DIALOG };
            // NOTE: the ClassWizard will add data members here
      //}}AFX_DATA

// Implementation
      BOOL Create();
protected:
      virtual void DoDataExchange(CDataExchange* pDX);      //
DDX/DDV support

      // Generated message map functions
      //{{AFX_MSG(CProgIndDlg)
      virtual void OnCancel();
      virtual BOOL OnInitDialog();
      afx_msg void OnPaint();
      afx_msg void OnClickedHelp();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
```

```
// prowidoc.cpp : implementation of the CProwin01Doc class
//

#include "stdafx.h"
#include "prowin01.h"
#include "prowidoc.h"
#include "mainfrm.h"
#include "getptdlg.h"
#include "datedlg.h"
#include "progind.h"
#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////
///////////
// CProwin01Doc

IMPLEMENT_SERIAL(CProwin01Doc, CDocument,0)

BEGIN_MESSAGE_MAP(CProwin01Doc, CDocument)
      //{{AFX_MSG_MAP(CProwin01Doc)
            // NOTE - the ClassWizard will add and remove mapping
macros here.
            //    DO NOT EDIT what you see in these blocks of
generated code!
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////
///////////
// CProwin01Doc construction/destruction

CProwin01Doc::CProwin01Doc()
{
    m_righttimestringArray.SetSize(0, 20); // allocate mem 40
elements at a time
      m_lefttimestringArray.SetSize(0, 20); // allocate mem 40
elements at a time
      m_rightstimsizeArray.SetSize(0, 20);
      m_leftstimsizeArray.SetSize(0, 20);
      m_rightpointArray.SetSize(0, 1520);
      m_rightinterceptArray.SetSize(0, 1520);
```

```
      m_rightUndoArray.SetSize(0, 1520);
      m_leftpointArray.SetSize(0, 1520);
      m_leftinterceptArray.SetSize(0, 1520);
      m_leftUndoArray.SetSize(0, 1520);
      m_rightslopeArray.SetSize(0, 1520);
      m_leftslopeArray.SetSize(0, 1520);
      m_rightpvalueArray.SetSize(0, 1520);
      m_leftpvalueArray.SetSize(0, 1520);
      m_rightTArray.SetSize(0, 1520);
      m_leftTArray.SetSize(0, 1520);
      m_rightsecondArray.SetSize(0, 20);
      m_leftsecondArray.SetSize(0, 20);
      m_rightorderArray.SetSize(0, 20);
      m_leftorderArray.SetSize(0, 20);

    m_bDateDialogDisplayed = FALSE;
    m_pDateDialog = new CDateDialog;
      m_bProgIndDlgDisplayed = FALSE;
    m_pProgIndDlg = new CProgIndDlg;

}

CProwin01Doc::~CProwin01Doc()
{
      delete m_pDateDialog;
      delete m_pProgIndDlg;
}

BOOL CProwin01Doc::OnNewDocument()
{   if (!CDocument::OnNewDocument())
            return FALSE;
      char buffer[MAX_INFILE_LENGTH]="";
      DWORD FileIndex;
      CGetPtDialog* p_gpdlg =
(CGetPtDialog*)((CProwin01App*)AfxGetApp())-
>m_pGetPtDialogArray[((CProwin01App*)AfxGetApp())-
>m_nActiveDatabaseIndex];
      CStdioFile* p_database = (p_gpdlg -> m_pFile);
      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);

                                          BeginWaitCursor();
                                                for (int i = 0; i
< p_gpdlg->m_SelCount; i++)
```

```
                                                    {for
(FileIndex = p_gpdlg->m_FileIndexArray[p_gpdlg->m_index[i]];
FileIndex < p_gpdlg->m_FileIndexArray[p_gpdlg->m_index[i + 1]];
FileIndex++)

      {p_database->Seek(COLUMN_NAMES_STRING_LENGTH + FileIndex *
LINE_LENGTH,CFile::begin);

      p_database->Read(buffer,LINE_LENGTH -2);

      buffer[LINE_LENGTH -2] = '\0';

      GetDocumentData(buffer);
                                                          }
                                                    }
                                                    m_name =
CString(buffer + NAME_POSITION, NAME_LENGTH);

      SetTitle(GetTitle() + ": " + m_name);
                                                    pStatus-
>SetPaneText(0,"Analyzing data for patient " + m_name,TRUE);
                                                    pStatus-
>UpdateWindow();
                                                    m_rightReanalyzed
= 0;
                                                    m_leftReanalyzed =
0;

      m_rightGaussianFiltered = 0;

      m_leftGaussianFiltered = 0;
                                                    DoTimeOrder();
                                                    Regress();

      //::sndPlaySound("done.wav",SND_ASYNC);
                                                    EndWaitCursor();
                                                    pStatus-
>SetPaneText(0,"For Help, press F1",TRUE);
                                                    pStatus-
>UpdateWindow();



      return TRUE;
```

```
}



void CProwin01Doc::GetDocumentData(char* buffer)
{     char eye = *(buffer + 185);
      char* runner = buffer + 267;
      CString timestring;
      DWORD seconds;
      DWORD baseline = 0;
      CTime refTime(1980,1,1,0,0,0);
      int year = atoi(buffer + 24);

      if (*(buffer + 140) != '3') return;////////////////////only
accept stimulus size III

      if (year <= 80 && year >= 37)return;
      if (year > 80) year += 1900;
      if (year < 37) year += 2000;


      int month = atoi(buffer + 21);
      if (month < 1 || month > 12 )return;

      int day = atoi(buffer + 18);
      if (day < 1 || day > 31 )return;

      int hour = atoi(buffer + 27);
      if (hour < 0 || hour > 23 )return;

      int min = atoi(buffer + 30);
      if (min < 0 || min > 59 )return;

      int sec = atoi(buffer + 33);
      if (sec < 0 || sec > 59 )return;
      CTime* pTime = new CTime(year,month,day,hour,min,sec);
      timestring = pTime -> FormatGmt("%a\t%d-%m-%y");
      CTimeSpan* pTimeSpan = new CTimeSpan;
      *pTimeSpan = *pTime - refTime;

      seconds = (DWORD)pTimeSpan -> GetTotalSeconds();
      if (eye == 'R')
            {m_rightsecondArray.Add(seconds);
            m_righttimestringArray.Add(timestring);
```

```
            for (int i = 0; i < 76; i++)
                  {if
(atof(runner)!=99)m_rightpointArray.Add((WORD)10*atof(runner));
                  else m_rightpointArray.Add(MISSING_VALUE);
                  runner += 5;
            }
      }

      else
            {m_leftsecondArray.Add(seconds);
            m_lefttimestringArray.Add(timestring);
            for (int i = 0; i < 76; i++)
                  {if
(atof(runner)!=99)m_leftpointArray.Add((WORD)10*atof(runner));
                  else m_leftpointArray.Add(MISSING_VALUE);
                  runner += 5;
            }

      }



      delete pTime;
      delete pTimeSpan;
}

void CProwin01Doc::DoTimeOrder()
{     DWORD baseline = 0xFFFFFFFF;
      DWORD arraymember;
      DWORD current_head;
      BYTE next_head_index;
      DWORD low_bound = 0;
      BYTE zero;
      int i = 0;
      int rsize = m_rightsecondArray.GetSize();
      int lsize = m_leftsecondArray.GetSize();

      if (rsize)
            {while (i < rsize)
////////////////////////////////////////////////finds lowest member
                  { arraymember = m_rightsecondArray.GetAt(i++);
                  if (arraymember < baseline) baseline =
arraymember;
            }
```

```
            m_rightbasetime = baseline;
            i = 0;
            while (i < rsize)
//////////////////////////////////////////////zeros array and
adds first member (0) to m_rightorderArray
                  { arraymember = m_rightsecondArray.GetAt(i) -
baseline;
                        if (!arraymember)
                              {zero = (BYTE)i;
                              m_rightorderArray.Add(zero);
                              TRACE("m_rightorderArray[0] =
%d\n",zero);
                        }
                  m_rightsecondArray.SetAt(i++,arraymember);
                  TRACE("m_rightsecondArray[%d] = %lu\n",i-
1,m_rightsecondArray[i-1]);
            }

                              //rsize - 1 because zero already
done
            for ( int j = 0; j < rsize - 1; j++)
//////////////////////////////ranks: fills m_rightorderArray
with increasing indices
                  {baseline = 0xFFFFFFFF;
                  current_head =
m_rightsecondArray.GetAt(m_rightorderArray.GetAt(m_rightorderArray
.GetUpperBound()));

                        for (int i = 0; i < rsize; i++)
                              { arraymember =
m_rightsecondArray.GetAt(i);
                              if (arraymember < baseline &&
arraymember > current_head)
                                    {next_head_index = i;
                                    baseline = arraymember;
                              }
                        }
                  m_rightorderArray.Add(next_head_index);
                  TRACE("m_rightorderArray[%d] = %d\n",j+1,
next_head_index);

            }

      i = 0;
```

```
      baseline = 0xFFFFFFFF;
      }

      if (lsize)
            {while (i < lsize)
////////////////////////////////////////////finds lowest member
                  { arraymember = m_leftsecondArray.GetAt(i++);
                  if (arraymember < baseline) baseline =
arraymember;
            }
            m_leftbasetime = baseline;
            i = 0;
            while (i < lsize)
////////////////////////////////////////////zeros array and
adds first member (0) to m_leftorderArray
                  { arraymember = m_leftsecondArray.GetAt(i) -
baseline;

                        if (!arraymember)
                              {zero = (BYTE)i;
                              m_leftorderArray.Add(zero);
                        }
                  m_leftsecondArray.SetAt(i++,arraymember);
            }


                                    //lsize - 1 because zero already
done
            for ( int k = 0; k < lsize - 1; k++)
////////////////////////////////ranks: fills m_leftorderArray with
increasing indices
                  {baseline = 0xFFFFFFFF;
                  current_head =
m_leftsecondArray.GetAt(m_leftorderArray.GetAt(m_leftorderArray.Ge
tUpperBound()));

                        for (int i = 0; i < lsize; i++)
                              { arraymember =
m_leftsecondArray.GetAt(i);
                              if (arraymember < baseline &&
arraymember > current_head)
                                    {next_head_index = i;
                                    baseline = arraymember;
                              }
```

```
                        }
                m_leftorderArray.Add(next_head_index);

            }
        }



}

#ifndef _DEBUG

void CProwin01Doc::Regress()
{     char slope_string[50];
      CString slopeString;
      long double sensitivity;
      long double sigma_sensitivity;
      long double mean_sensitivity; ///for a single retinal
location
      long double residual;
      long double sigma_residual_square;
      long double slope;
      long double std_err_slope;
      long double time;
      long double mean_time;
      long double sigma_time;
      long double time_offset; //time - mean(time)
      long double sigma_time_offset_square;
      long double sigma_time_x_sens_offset;
      long double sigma_time_square;
      long double square_sigma_time;
      long double intercept;

      BYTE point;
      BYTE field;
      BYTE nFields;
      BYTE i;
      BYTE current_no_fields;
      BYTE valid_points;

      ////////////for output file////////////
      CStdioFile outfile;
      char buffer[1000]="";
```

```
      nFields = m_rightorderArray.GetSize();
      for ( point = 0; point < 76; point++)
            for ( field = 0; field < nFields; field++)
                  {if (field < 2)  //////////always label first 2
fields as flat////////////////////
                        {m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],5);
                        m_rightslopeArray.SetAtGrow(point +
76*m_rightorderArray[field],"");
                        continue;
                  }
                  current_no_fields = field + 1;
                  sigma_time = 0;
                  sigma_time_square = 0;
                  square_sigma_time = 0;
                  sigma_sensitivity = 0;
                  sigma_time_offset_square = 0;
                  sigma_time_x_sens_offset = 0;
                  sigma_residual_square = 0;
                  valid_points = 0;
                  for ( i = 0; i < current_no_fields; i++)
                        {if ((sensitivity =
m_rightpointArray.GetAt(point + 76*m_rightorderArray.GetAt(i))) !=
MISSING_VALUE)
                              {sigma_sensitivity += sensitivity;
                              time =
m_rightsecondArray.GetAt(m_rightorderArray.GetAt(i));
                              sigma_time += time;
                              sigma_time_square += time*time;
                              valid_points++;

                        }

                  }
                  if (valid_points < 3)  //////////show as flat
if 2 or fewer valid sensitivities////////////////////
                        {m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],5);
                        m_rightslopeArray.SetAtGrow(point +
76*m_rightorderArray[field],"");
                        continue;
                  }
```

```
                square_sigma_time = sigma_time*sigma_time;
              mean_time = sigma_time / current_no_fields;
                mean_sensitivity = sigma_sensitivity /
current_no_fields;



                for ( i = 0; i < current_no_fields; i++)
                      {if ((sensitivity =
m_rightpointArray.GetAt(point + 76*m_rightorderArray.GetAt(i))) !=
MISSING_VALUE)
                            {time_offset =
m_rightsecondArray.GetAt(m_rightorderArray.GetAt(i)) - mean_time;
                            sigma_time_offset_square +=
time_offset * time_offset;
                            sigma_time_x_sens_offset +=
time_offset*(sensitivity - mean_sensitivity);

                      }
                }
                slope = sigma_time_x_sens_offset /
sigma_time_offset_square;
                sprintf(slope_string,"%Le",slope *
SECONDS_PER_YEAR/10);////////////////converts to dB/yr for storage
                slopeString = slope_string;
////////////////as CString
                m_rightslopeArray.SetAtGrow(point +
76*m_rightorderArray[field],slopeString);
                intercept = mean_sensitivity - slope *
mean_time; //first test time = 0 for this

                /* true non zeroed intercepts stored for
projections*/
                m_rightinterceptArray.SetAtGrow(point +
76*m_rightorderArray[field],
                mean_sensitivity - slope * (mean_time +
m_rightbasetime));


                for ( i = 0; i < current_no_fields; i++)
                      if ((sensitivity =
m_rightpointArray.GetAt(point + 76*m_rightorderArray.GetAt(i))) !=
MISSING_VALUE)
```

```
                                {time =
m_rightsecondArray.GetAt(m_rightorderArray.GetAt(i));
                                residual = sensitivity - time *
slope - intercept;
                                sigma_residual_square += residual *
residual;
                        }
                    std_err_slope = sqrtl(sigma_residual_square /
((current_no_fields - 2) * (sigma_time_square - square_sigma_time
/ current_no_fields)));
                    if (!std_err_slope)std_err_slope = 1e-
100;/////////////////as cannot divide by 0

                    slope /= std_err_slope;/////*******  'slope' NOW
REPRESENTS T STATISTIC ******///////////
                    m_rightTArray.SetAtGrow(point +
76*m_rightorderArray[field],slope*1000);

                        // get t table value
with((CProwin01App*)AfxGetApp())->m_tvalue[row][column] NB zero
based;

                    if ((slope < 0) && (fabsl(slope) <=
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][1]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],6);         //p>=0.1   -ve slope
                                else if (slope <
0)m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],7);//p<0.1    -ve slope

                if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][2]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],8);         //p<0.05   -ve slope

                if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][4]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],9);         //p<0.01   -ve slope

                if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][5]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],11);         //p<0.001   -ve slope
```

```
                if ((slope >= 0) && (slope <=
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][1]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],4);          //p>=0.1   +ve slope
                                else if (slope >=
0)m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],3);//p<0.1    +ve slope

                if ((slope >= 0) && (slope >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][2]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],2);          //p<0.05   +ve slope

                if ((slope >= 0) && (slope >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][4]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],1);          //p<0.01   +ve slope


            }

      ////////////for output file////////////
      outfile.Open("output.txt", CFile::modeCreate |
CFile::modeReadWrite | CFile::typeText);
      outfile.WriteString("name,eye,point,field,date,seconds,sensi
tivity*10,slope,intercept,T*1000\n");
      for ( point = 0; point < 76; point++)
            for ( field = 0; field < nFields; field++)
                  {sprintf(buffer,"%s,R,%d,%d,%s,%ld,",(const
char*)m_name,point+1,field+1,
                  (const
char*)(m_righttimestringArray.GetAt(m_rightorderArray.GetAt(field)
).Right(8)),

      m_rightsecondArray.GetAt(m_rightorderArray.GetAt(field)));
                  outfile.WriteString(buffer);

                  if (m_rightpointArray.GetAt(point +
76*m_rightorderArray.GetAt(field)) != MISSING_VALUE)

      sprintf(buffer,"%d,",m_rightpointArray.GetAt(point +
76*m_rightorderArray.GetAt(field)));
                  else sprintf(buffer,",");
```

```
                outfile.WriteString(buffer);

                if (field > 1 && m_rightpointArray.GetAt(point +
76*m_rightorderArray.GetAt(field)) != MISSING_VALUE)
                sprintf(buffer,"%s,%d,%d\n",
                (const char*)(m_rightslopeArray.GetAt(point +
76*m_rightorderArray.GetAt(field))),
                m_rightinterceptArray.GetAt(point +
76*m_rightorderArray.GetAt(field)),
                m_rightTArray.GetAt(point +
76*m_rightorderArray.GetAt(field))
                  );
                else sprintf(buffer,",,\n");
                outfile.WriteString(buffer);
      }




      nFields = m_leftorderArray.GetSize();
      for ( point = 0; point < 76; point++)
            for ( field = 0; field < nFields; field++)
                {if (field < 2)  //////////always label first 2
fields as flat////////////////////
                      {m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],5);
                      m_leftslopeArray.SetAtGrow(point +
76*m_leftorderArray[field],"");
                      continue;
                }
                current_no_fields = field + 1;
                sigma_time = 0;
                sigma_time_square = 0;
                square_sigma_time = 0;
                sigma_sensitivity = 0;
                sigma_time_offset_square = 0;
                sigma_time_x_sens_offset = 0;
                sigma_residual_square = 0;
                valid_points = 0;
                for ( i = 0; i < current_no_fields; i++)
```

```
                        {if ((sensitivity =
m_leftpointArray.GetAt(point + 76*m_leftorderArray.GetAt(i))) !=
MISSING_VALUE)
                              {sigma_sensitivity += sensitivity;
                              time =
m_leftsecondArray.GetAt(m_leftorderArray.GetAt(i));
                              sigma_time += time;
                              sigma_time_square += time*time;
                              valid_points++;

                        }

                  }
                  if (valid_points < 3)  //////////show as flat
if 2 or fewer valid sensitivities//////////////////////
                        {m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],5);
                        m_leftslopeArray.SetAtGrow(point +
76*m_leftorderArray[field],"");
                        continue;
                  }
                  square_sigma_time = sigma_time*sigma_time;
               mean_time = sigma_time / current_no_fields;
                  mean_sensitivity = sigma_sensitivity /
current_no_fields;



                  for ( i = 0; i < current_no_fields; i++)
                        {if ((sensitivity =
m_leftpointArray.GetAt(point + 76*m_leftorderArray.GetAt(i))) !=
MISSING_VALUE)
                              {time_offset =
m_leftsecondArray.GetAt(m_leftorderArray.GetAt(i)) - mean_time;
                              sigma_time_offset_square +=
time_offset * time_offset;
                              sigma_time_x_sens_offset +=
time_offset*(sensitivity - mean_sensitivity);

                        }
                  }
                  slope = sigma_time_x_sens_offset /
sigma_time_offset_square;
```

```
                sprintf(slope_string,"%Le",slope *
SECONDS_PER_YEAR/10);////////////////converts to dB/yr for storage
                slopeString = slope_string;
////////////////as CString
                m_leftslopeArray.SetAtGrow(point +
76*m_leftorderArray[field],slopeString);
                intercept = mean_sensitivity - slope *
mean_time; //first test time = 0 for this

                /* true non zeroed intercepts stored for
projections*/
                m_leftinterceptArray.SetAtGrow(point +
76*m_leftorderArray[field],
                mean_sensitivity - slope * (mean_time +
m_leftbasetime));


                for ( i = 0; i < current_no_fields; i++)
                      if ((sensitivity =
m_leftpointArray.GetAt(point + 76*m_leftorderArray.GetAt(i))) !=
MISSING_VALUE)
                            {time =
m_leftsecondArray.GetAt(m_leftorderArray.GetAt(i));
                            residual = sensitivity - time *
slope - intercept;
                            sigma_residual_square += residual *
residual;
                      }
                std_err_slope = sqrtl(sigma_residual_square /
((current_no_fields - 2) * (sigma_time_square - square_sigma_time
/ current_no_fields)));
                if (!std_err_slope)std_err_slope = 1e-
100;////////////////as cannot divide by 0

                slope /= std_err_slope;/////*******  'slope' NOW
REPRESENTS T STATISTIC ******//////////
                m_leftTArray.SetAtGrow(point +
76*m_leftorderArray[field],slope);
                m_leftTArray.SetAtGrow(point +
76*m_leftorderArray[field],slope*1000);
```

```
                        // get t table value
with((CProwin01App*)AfxGetApp())->m_tvalue[row][column] NB zero
based;

                  if ((slope < 0) && (fabsl(slope) <=
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][1]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],6);         //p>=0.1   -ve slope
                              else if (slope <
0)m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],7);//p<0.1    -ve slope

                if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][2]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],8);         //p<0.05   -ve slope

                if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][4]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],9);         //p<0.01   -ve slope

                if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][5]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],11);         //p<0.001   -ve slope

                if ((slope >= 0) && (slope <=
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][1]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],4);         //p>=0.1   +ve slope
                              else if (slope >=
0)m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],3);//p<0.1    +ve slope

                if ((slope >= 0) && (slope >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][2]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],2);         //p<0.05   +ve slope

                if ((slope >= 0) && (slope >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][4]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],1);         //p<0.01   +ve slope
```

```
            }


      ////////////for output file////////////
      for ( point = 0; point < 76; point++)
            for ( field = 0; field < nFields; field++)
                  {sprintf(buffer,"%s,L,%d,%d,%s,%ld,",(const
char*)m_name,point+1,field+1,
                  (const
char*)(m_lefttimestringArray.GetAt(m_leftorderArray.GetAt(field)).
Right(8)),

      m_leftsecondArray.GetAt(m_leftorderArray.GetAt(field)));
                  outfile.WriteString(buffer);

                  if (m_leftpointArray.GetAt(point +
76*m_leftorderArray.GetAt(field)) != MISSING_VALUE)

      sprintf(buffer,"%d,",m_leftpointArray.GetAt(point +
76*m_leftorderArray.GetAt(field)));
                  else sprintf(buffer,",");
                  outfile.WriteString(buffer);

                  if (field > 1 && m_leftpointArray.GetAt(point +
76*m_leftorderArray.GetAt(field)) != MISSING_VALUE)
                  sprintf(buffer,"%s,%d,%d\n",
                  (const char*)(m_leftslopeArray.GetAt(point +
76*m_leftorderArray.GetAt(field))),
                  m_leftinterceptArray.GetAt(point +
76*m_leftorderArray.GetAt(field)),
                  m_leftTArray.GetAt(point +
76*m_leftorderArray.GetAt(field))
                    );
                  else sprintf(buffer,",,\n");
                  outfile.WriteString(buffer);
      }


}
#endif

#ifdef _DEBUG
```

```
void CProwin01Doc::Regress()  //Reverse version
{     char slope_string[50];
      CString slopeString;
      long double sensitivity;
      long double sigma_sensitivity;
      long double mean_sensitivity; ///for a single retinal
location
      long double residual;
      long double sigma_residual_square;
      long double slope;
      long double std_err_slope;
      long double time;
      long double mean_time;
      long double sigma_time;
      long double time_offset; //time - mean(time)
      long double sigma_time_offset_square;
      long double sigma_time_x_sens_offset;
      long double sigma_time_square;
      long double square_sigma_time;
      long double intercept;

      CByteArray reverseArray;//Reverse version

      BYTE point;
      BYTE field;
      BYTE nFields;
      BYTE i;
      BYTE current_no_fields;
      BYTE valid_points;

      reverseArray.SetSize(0, 20);//Reverse version


      nFields = m_rightorderArray.GetSize();

      for ( field = 0; field < nFields; field++)//Reverse version

      reverseArray.SetAtGrow(field,m_rightorderArray[nFields-
field-1]);//Reverse version

      for ( field = 0; field < nFields; field++)//Reverse version
```

```
      m_rightorderArray.SetAtGrow(field,reverseArray[field]);//Rev
erse version


      for ( point = 0; point < 76; point++)
            for ( field = 0; field < nFields; field++)
                  {if (field < 2)  //////////always label first 2
fields as flat//////////////////////
                        {m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],5);
                        m_rightslopeArray.SetAtGrow(point +
76*m_rightorderArray[field],"");
                        continue;
                  }
                  current_no_fields = field + 1;
                  sigma_time = 0;
                  sigma_time_square = 0;
                  square_sigma_time = 0;
                  sigma_sensitivity = 0;
                  sigma_time_offset_square = 0;
                  sigma_time_x_sens_offset = 0;
                  sigma_residual_square = 0;
                  valid_points = 0;
                  for ( i = 0; i < current_no_fields; i++)
                        {if ((sensitivity =
m_rightpointArray.GetAt(point + 76*m_rightorderArray.GetAt(i))) !=
MISSING_VALUE)
                              {sigma_sensitivity += sensitivity;
                              time =
m_rightsecondArray.GetAt(m_rightorderArray.GetAt(i));
                              sigma_time += time;
                              sigma_time_square += time*time;
                              valid_points++;

                        }

                  }
                  if (valid_points < 3)  //////////show as flat
if 2 or fewer valid sensitivities//////////////////////
                        {m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],5);
                        m_rightslopeArray.SetAtGrow(point +
76*m_rightorderArray[field],"");
```

```
                        continue;
                    }
                    square_sigma_time = sigma_time*sigma_time;
                mean_time = sigma_time / current_no_fields;
                    mean_sensitivity = sigma_sensitivity /
current_no_fields;



                    for ( i = 0; i < current_no_fields; i++)
                        {if ((sensitivity =
m_rightpointArray.GetAt(point + 76*m_rightorderArray.GetAt(i))) !=
MISSING_VALUE)
                                {time_offset =
m_rightsecondArray.GetAt(m_rightorderArray.GetAt(i)) - mean_time;
                                sigma_time_offset_square +=
time_offset * time_offset;
                                sigma_time_x_sens_offset +=
time_offset*(sensitivity - mean_sensitivity);

                        }
                    }
                    slope = sigma_time_x_sens_offset /
sigma_time_offset_square;
                    sprintf(slope_string,"%Le",slope *
SECONDS_PER_YEAR/10);////////////////converts to dB/yr for storage
                    slopeString = slope_string;
////////////////as CString
                    m_rightslopeArray.SetAtGrow(point +
76*m_rightorderArray[field],slopeString);
                    intercept = mean_sensitivity - slope *
mean_time; //first test time = 0 for this

                    /* true non zeroed intercepts stored for
projections*/
                    m_rightinterceptArray.SetAtGrow(point +
76*m_rightorderArray[field],
                    mean_sensitivity - slope * (mean_time +
m_rightbasetime));


                    for ( i = 0; i < current_no_fields; i++)
```

```
                        if ((sensitivity =
m_rightpointArray.GetAt(point + 76*m_rightorderArray.GetAt(i))) !=
MISSING_VALUE)
                                {time =
m_rightsecondArray.GetAt(m_rightorderArray.GetAt(i));
                                residual = sensitivity - time *
slope - intercept;
                                sigma_residual_square += residual *
residual;
                        }
                    std_err_slope = sqrtl(sigma_residual_square /
((current_no_fields - 2) * (sigma_time_square - square_sigma_time
/ current_no_fields)));
                    if (!std_err_slope)std_err_slope = 1e-
100;/////////////////as cannot divide by 0

                    slope /= std_err_slope;/////*******  'slope' NOW
REPRESENTS T STATISTIC ******//////////

                        // get t table value
with((CProwin01App*)AfxGetApp())->m_tvalue[row][column] NB zero
based;

                    if ((slope < 0) && (fabsl(slope) <=
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][1]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],6);         //p>=0.1   -ve slope
                                else if (slope <
0)m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],7);//p<0.1    -ve slope

                  if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][2]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],8);         //p<0.05   -ve slope

                    if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][4]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],9);         //p<0.01   -ve slope

                    if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][5]))
```

```
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],11);          //p<0.001   -ve slope

                if ((slope >= 0) && (slope <=
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][1]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],4);          //p>=0.1   +ve slope
                                else if (slope >=
0)m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],3);//p<0.1    +ve slope

                if ((slope >= 0) && (slope >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][2]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],2);          //p<0.05   +ve slope

                if ((slope >= 0) && (slope >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][4]))
                        m_rightpvalueArray.SetAtGrow(point +
76*m_rightorderArray[field],1);          //p<0.01   +ve slope


            }

      for ( field = 0; field < nFields; field++)//Reverse version

      reverseArray.SetAtGrow(field,m_rightorderArray[nFields-
field-1]);//Reverse version

      for ( field = 0; field < nFields; field++)//Reverse version

      m_rightorderArray.SetAtGrow(field,reverseArray[field]);//Rev
erse version







      nFields = m_leftorderArray.GetSize();

      for ( field = 0; field < nFields; field++)//Reverse version
```

```
            reverseArray.SetAtGrow(field,m_leftorderArray[nFields-
field-1]);//Reverse version

      for ( field = 0; field < nFields; field++)//Reverse version

      m_leftorderArray.SetAtGrow(field,reverseArray[field]);//Reve
rse version

      for ( point = 0; point < 76; point++)
            for ( field = 0; field < nFields; field++)
                  {if (field < 2)  //////////always label first 2
fields as flat////////////////////
                        {m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],5);
                        m_leftslopeArray.SetAtGrow(point +
76*m_leftorderArray[field],"");
                        continue;
                  }
                  current_no_fields = field + 1;
                  sigma_time = 0;
                  sigma_time_square = 0;
                  square_sigma_time = 0;
                  sigma_sensitivity = 0;
                  sigma_time_offset_square = 0;
                  sigma_time_x_sens_offset = 0;
                  sigma_residual_square = 0;
                  valid_points = 0;
                  for ( i = 0; i < current_no_fields; i++)
                        {if ((sensitivity =
m_leftpointArray.GetAt(point + 76*m_leftorderArray.GetAt(i))) !=
MISSING_VALUE)
                              {sigma_sensitivity += sensitivity;
                              time =
m_leftsecondArray.GetAt(m_leftorderArray.GetAt(i));
                              sigma_time += time;
                              sigma_time_square += time*time;
                              valid_points++;

                        }

                  }
                  if (valid_points < 3)  //////////show as flat
if 2 or fewer valid sensitivities////////////////////
```

```
                        {m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],5);
                        m_leftslopeArray.SetAtGrow(point +
76*m_leftorderArray[field],"");
                        continue;
                    }
                    square_sigma_time = sigma_time*sigma_time;
                mean_time = sigma_time / current_no_fields;
                    mean_sensitivity = sigma_sensitivity /
current_no_fields;



                    for ( i = 0; i < current_no_fields; i++)
                        {if ((sensitivity =
m_leftpointArray.GetAt(point + 76*m_leftorderArray.GetAt(i))) !=
MISSING_VALUE)
                                {time_offset =
m_leftsecondArray.GetAt(m_leftorderArray.GetAt(i)) - mean_time;
                                sigma_time_offset_square +=
time_offset * time_offset;
                                sigma_time_x_sens_offset +=
time_offset*(sensitivity - mean_sensitivity);

                        }
                    }
                    slope = sigma_time_x_sens_offset /
sigma_time_offset_square;
                    sprintf(slope_string,"%Le",slope *
SECONDS_PER_YEAR/10);////////////////converts to dB/yr for storage
                    slopeString = slope_string;
////////////////as CString
                    m_leftslopeArray.SetAtGrow(point +
76*m_leftorderArray[field],slopeString);
                    intercept = mean_sensitivity - slope *
mean_time; //first test time = 0 for this

                    /* true non zeroed intercepts stored for
projections*/
                    m_leftinterceptArray.SetAtGrow(point +
76*m_leftorderArray[field],
                    mean_sensitivity - slope * (mean_time +
m_leftbasetime));
```

```
                for ( i = 0; i < current_no_fields; i++)
                      if ((sensitivity =
m_leftpointArray.GetAt(point + 76*m_leftorderArray.GetAt(i))) !=
MISSING_VALUE)
                                {time =
m_leftsecondArray.GetAt(m_leftorderArray.GetAt(i));
                                residual = sensitivity - time *
slope - intercept;
                                sigma_residual_square += residual *
residual;
                      }
                std_err_slope = sqrtl(sigma_residual_square /
((current_no_fields - 2) * (sigma_time_square - square_sigma_time
/ current_no_fields)));
                if (!std_err_slope)std_err_slope = 1e-
100;/////////////////as cannot divide by 0

                slope /= std_err_slope;/////*******  'slope' NOW
REPRESENTS T STATISTIC ******////////////

                      // get t table value
with((CProwin01App*)AfxGetApp())->m_tvalue[row][column] NB zero
based;

                if ((slope < 0) && (fabsl(slope) <=
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][1]))
                      m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],6);         //p>=0.1   -ve slope
                                else if (slope <
0)m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],7);//p<0.1    -ve slope

              if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][2]))
                      m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],8);         //p<0.05   -ve slope

              if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][4]))
                      m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],9);         //p<0.01   -ve slope
```

```
                if ((slope < 0) && (fabsl(slope) >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][5]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],11);          //p<0.001   -ve slope

                if ((slope >= 0) && (slope <=
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][1]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],4);          //p>=0.1   +ve slope
                                else if (slope >=
0)m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],3);//p<0.1    +ve slope

                if ((slope >= 0) && (slope >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][2]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],2);          //p<0.05   +ve slope

                if ((slope >= 0) && (slope >
((CProwin01App*)AfxGetApp())->m_tvalue[current_no_fields - 3][4]))
                        m_leftpvalueArray.SetAtGrow(point +
76*m_leftorderArray[field],1);          //p<0.01   +ve slope


            }

      for ( field = 0; field < nFields; field++)//Reverse version
            reverseArray.SetAtGrow(field,m_leftorderArray[nFields-
field-1]);//Reverse version

      for ( field = 0; field < nFields; field++)//Reverse version

      m_leftorderArray.SetAtGrow(field,reverseArray[field]);//Reve
rse version

}
#endif


/////////////////////////////////////////////////////////////////////////
////////////
// CProwin01Doc serialization

void CProwin01Doc::Serialize(CArchive& ar)
```

```
{
      if (ar.IsStoring())
      {
      BeginWaitCursor();

      ar << m_rightReanalyzed;
            ar << m_leftReanalyzed;
            ar << m_rightGaussianFiltered;
            ar << m_leftGaussianFiltered;
            ar << m_name ;
            m_lefttimestringArray.Serialize(ar);
            m_leftstimsizeArray.Serialize(ar);
            m_leftpointArray.Serialize(ar);
            m_leftinterceptArray.Serialize(ar);
            m_leftUndoArray.Serialize(ar);
            m_leftslopeArray.Serialize(ar);
            m_leftpvalueArray.Serialize(ar);
            m_leftTArray.Serialize(ar);
            m_leftorderArray.Serialize(ar);
            m_leftsecondArray.Serialize(ar);

            m_righttimestringArray.Serialize(ar);
            m_rightstimsizeArray.Serialize(ar);
            m_rightpointArray.Serialize(ar);
            m_rightinterceptArray.Serialize(ar);
            m_rightUndoArray.Serialize(ar);
            m_rightslopeArray.Serialize(ar);
            m_rightpvalueArray.Serialize(ar);
            m_rightTArray.Serialize(ar);
            m_rightorderArray.Serialize(ar);
            m_rightsecondArray.Serialize(ar);


            EndWaitCursor();
      }
      else
      {
      BeginWaitCursor();

            ar >> m_rightReanalyzed;
            ar >> m_leftReanalyzed;
            ar >> m_rightGaussianFiltered;
            ar >> m_leftGaussianFiltered;
            ar >> m_name;
```

```
            m_lefttimestringArray.Serialize(ar);
            m_leftstimsizeArray.Serialize(ar);
            m_leftpointArray.Serialize(ar);
            m_leftinterceptArray.Serialize(ar);
            m_leftUndoArray.Serialize(ar);
            m_leftslopeArray.Serialize(ar);
            m_leftpvalueArray.Serialize(ar);
            m_leftTArray.Serialize(ar);
            m_leftorderArray.Serialize(ar);
            m_leftsecondArray.Serialize(ar);

            m_righttimestringArray.Serialize(ar);
            m_rightstimsizeArray.Serialize(ar);
            m_rightpointArray.Serialize(ar);
            m_rightinterceptArray.Serialize(ar);
            m_rightUndoArray.Serialize(ar);
            m_rightslopeArray.Serialize(ar);
            m_rightpvalueArray.Serialize(ar);
            m_rightTArray.Serialize(ar);
            m_rightorderArray.Serialize(ar);
            m_rightsecondArray.Serialize(ar);


            EndWaitCursor();
        }
}

////////////////////////////////////////////////////////////////
//////////
void CProwin01Doc::DeleteContents()
{
    m_righttimestringArray.RemoveAll();
    m_rightstimsizeArray.RemoveAll();
    m_rightpointArray.RemoveAll();
    m_rightinterceptArray.RemoveAll();
    m_rightUndoArray.RemoveAll();
    m_rightslopeArray.RemoveAll();
    m_rightTArray.RemoveAll();
    m_rightpvalueArray.RemoveAll();
    m_rightsecondArray.RemoveAll();
    m_rightorderArray.RemoveAll();

    m_lefttimestringArray.RemoveAll();
    m_leftstimsizeArray.RemoveAll();
```

```
    m_leftpointArray.RemoveAll();
    m_leftinterceptArray.RemoveAll();
    m_leftUndoArray.RemoveAll();
    m_leftslopeArray.RemoveAll();
    m_leftTArray.RemoveAll();
    m_leftpvalueArray.RemoveAll();
    m_leftsecondArray.RemoveAll();
    m_leftorderArray.RemoveAll();


}

/////////////////////////////////////////////////////////////////////////
///////////


/////////////////////////////////////////////////////////////////////////
///////////
// CProwin01Doc diagnostics

#ifdef _DEBUG
void CProwin01Doc::AssertValid() const
{
      CDocument::AssertValid();
}

void CProwin01Doc::Dump(CDumpContext& dc) const
{
      CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////
///////////
// CProwin01Doc commands
```

```
// prowidoc.h : interface of the CProwin01Doc class
//
/////////////////////////////////////////////////////////////////////////
///////////
#include <math.h>
#define MISSING_VALUE 990
#define SECONDS_PER_YEAR  31557600
#define MAX_FIELDS 20
class CProwin01Doc : public CDocument
{
      friend class CProwin01View;
      friend class CLeftView;
      friend class CDateDialog;
    friend class CProgIndDlg;

protected: // create from serialization only
      CProwin01Doc();
      DECLARE_SERIAL(CProwin01Doc)


private:                                  // accessible to friends
      BYTE m_rightReanalyzed;
      BYTE m_leftReanalyzed;
      BYTE m_rightGaussianFiltered;
      BYTE m_leftGaussianFiltered;
      int m_nLeftCount;
      int m_nRightCount;
      CString m_name;
      BOOL m_bDateDialogDisplayed;
      CDateDialog* m_pDateDialog;
      BOOL m_bProgIndDlgDisplayed;
      CProgIndDlg* m_pProgIndDlg;

      CStringArray m_lefttimestringArray;
      CDWordArray m_leftsecondArray;
      CByteArray m_leftstimsizeArray;
      CWordArray m_leftpointArray;
      CWordArray m_leftinterceptArray;
      CWordArray m_leftUndoArray;
      CStringArray m_leftslopeArray;
      CByteArray m_leftpvalueArray;
      CWordArray m_leftTArray;
      CByteArray m_leftorderArray;
      DWORD m_leftbasetime;
```

```
	CStringArray m_righttimestringArray;
	CDWordArray m_rightsecondArray;
	CByteArray m_rightstimsizeArray;
	CWordArray m_rightpointArray;
	CWordArray m_rightinterceptArray;
	CWordArray m_rightUndoArray;
	CStringArray m_rightslopeArray;
	CByteArray m_rightpvalueArray;
	CWordArray m_rightTArray;
	CByteArray m_rightorderArray;
	DWORD m_rightbasetime;


// Attributes
public:
// Operations
public:
	void GetDocumentData(char* buffer);
	void DoTimeOrder();
	void Regress();
// Implementation
public:
	virtual ~CProwin01Doc();
	virtual void Serialize(CArchive& ar);   // overridden for
document i/o
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:
	virtual BOOL OnNewDocument();
	virtual void DeleteContents();

// Generated message map functions
protected:
	//{{AFX_MSG(CProwin01Doc)
		// NOTE - the ClassWizard will add and remove member
functions here.
		//    DO NOT EDIT what you see in these blocks of
generated code !
	//}}AFX_MSG
```

```
    DECLARE_MESSAGE_MAP()
};

////////////////////////////////////////////////////////////////////
///////////
```

```
// prowin01.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "prowin01.h"

#include "mainfrm.h"
#include "splitfrm.h"
#include "prowidoc.h"
#include "prowivw.h"
#include "leftview.h"
#include "getptdlg.h"
#include "welcome.h"
#include "address.h"
#include "refdlg.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
///////////
// CProwin01App

BEGIN_MESSAGE_MAP(CProwin01App, CWinApp)
      //{{AFX_MSG_MAP(CProwin01App)
      ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
      ON_COMMAND(ID_FILE_NEW, OnFileNew)
      //}}AFX_MSG_MAP
      // Standard file based document commands
      ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
      ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
      // Standard print setup command
      ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////
///////////
// CProwin01App construction

CProwin01App::CProwin01App()
{   m_bLegendDialogDisplayed = FALSE;
```

```
      float tvalue[60][6]  =
/////these values from Altman 'Prac. Stats. for Med. Res. p521
            {3.078,     6.314,      12.706,     31.821,
      63.657,     636.619,
            1.886,      2.920,      4.303,      6.965,      9.925,
      31.599,
            1.638,      2.353,      3.182,      4.541,      5.841,
      12.924,
            1.533,      2.132,      2.776,      3.747,      4.604,
      8.610,
            1.476,      2.015,      2.571,      3.365,      4.032,
      6.869,
            1.440,      1.943,      2.447,      3.143,      3.707,
      5.959,
            1.415,      1.895,      2.365,      2.998,      3.499,
      5.408,
            1.397,      1.860,      2.306,      2.896,      3.355,
      5.041,
            1.383,      1.833,      2.262,      2.821,      3.250,
      4.781,
            1.372,      1.812,      2.228,      2.764,      3.169,
      4.587,
            1.363,      1.796,      2.201,      2.718,      3.106,
      4.437,
            1.356,      1.782,      2.179,      2.681,      3.055,
      4.318,
            1.350,      1.771,      2.160,      2.650,      3.012,
      4.221,
            1.345,      1.761,      2.145,      2.624,      2.977,
      4.140,
            1.341,      1.753,      2.131,      2.602,      2.947,
      4.073,
            1.337,      1.746,      2.120,      2.583,      2.921,
      4.015,
            1.333,      1.740,      2.110,      2.567,      2.898,
      3.965,
            1.330,      1.734,      2.101,      2.552,      2.878,
      3.922,
            1.328,      1.729,      2.093,      2.539,      2.861,
      3.883,
            1.325,      1.725,      2.086,      2.528,      2.845,
      3.850,
            1.323,      1.721,      2.080,      2.518,      2.831,
      3.819,
```

```
      1.321,      1.717,      2.074,      2.508,      2.819,
3.792,
      1.319,      1.714,      2.069,      2.500,      2.807,
3.768,
      1.318,      1.711,      2.064,      2.492,      2.797,
3.745,
      1.316,      1.708,      2.060,      2.485,      2.787,
3.725,
      1.315,      1.706,      2.056,      2.479,      2.779,
3.707,
      1.314,      1.703,      2.052,      2.473,      2.771,
3.690,
      1.313,      1.701,      2.048,      2.467,      2.763,
3.674,
      1.311,      1.699,      2.045,      2.462,      2.756,
3.659,
      1.310,      1.697,      2.042,      2.457,      2.750,
3.646,
      1.309,      1.696,      2.040,      2.453,      2.744,
3.633,
      1.309,      1.694,      2.037,      2.449,      2.738,
3.622,
      1.308,      1.692,      2.035,      2.445,      2.733,
3.611,
      1.307,      1.691,      2.032,      2.441,      2.728,
3.601,
      1.306,      1.690,      2.030,      2.438,      2.724,
3.591,
      1.306,      1.688,      2.028,      2.434,      2.719,
3.582,
      1.305,      1.687,      2.026,      2.431,      2.715,
3.574,
      1.304,      1.686,      2.024,      2.429,      2.712,
3.566,
      1.304,      1.685,      2.023,      2.426,      2.708,
3.558,
      1.303,      1.684,      2.021,      2.423,      2.704,
3.551,
      1.303,      1.683,      2.020,      2.421,      2.701,
3.544,
      1.302,      1.682,      2.018,      2.418,      2.698,
3.538,
      1.302,      1.681,      2.017,      2.416,      2.695,
3.532,
```

```
            1.301,      1.680,      2.015,      2.414,      2.692,
      3.526,
            1.301,      1.679,      2.014,      2.412,      2.690,
      3.520,
            1.300,      1.679,      2.013,      2.410,      2.687,
      3.515,
            1.300,      1.678,      2.012,      2.408,      2.685,
      3.510,
            1.299,      1.677,      2.011,      2.407,      2.682,
      3.505,
            1.299,      1.677,      2.010,      2.405,      2.680,
      3.500,
            1.299,      1.676,      2.009,      2.403,      2.678,
      3.496,
            1.298,      1.675,      2.008,      2.402,      2.676,
      3.492,
            1.298,      1.675,      2.007,      2.400,      2.674,
      3.488,
            1.298,      1.674,      2.006,      2.399,      2.672,
      3.484,
            1.297,      1.674,      2.005,      2.397,      2.670,
      3.480,
            1.297,      1.673,      2.004,      2.396,      2.668,
      3.476,
            1.297,      1.673,      2.003,      2.395,      2.667,
      3.473,
            1.297,      1.672,      2.002,      2.394,      2.665,
      3.470,
            1.296,      1.672,      2.002,      2.392,      2.663,
      3.466,
            1.296,      1.671,      2.001,      2.391,      2.662,
      3.463,
            1.296,      1.671,      2.000,      2.390,      2.660,
      3.460};
      for (int i = 0; i < 60; i++)
            for(int j = 0; j < 6; j++)
                  m_tvalue[i][j] = tvalue[i][j];
      // Place all significant initialization in InitInstance
}

////////////////////////////////////////////////////////////////////
///////////
// The one and only CProwin01App object
```

```
CProwin01App NEAR theApp;

/////////////////////////////////////////////////////////////////////
///////////
// CProwin01App initialization

BOOL CProwin01App::InitInstance()
{
      BeginWaitCursor();
      CWelcomeDialog dlg;

    m_pGetPtDialogArray.SetSize(0,5);
      m_nGetPtDialogIndex = 0;
      DoTitleWindow();
      // Progression criteria
      m_strMinInnerSlope = AfxGetApp() ->
GetProfileString("Progression criteria","InnerSlope","1.00");
      m_strMinEdgeSlope = AfxGetApp() ->
GetProfileString("Progression criteria","EdgeSlope","2.00");
      m_nInnerPValue = AfxGetApp() -> GetProfileInt("Progression
criteria","InnerP",1);
      m_nEdgePValue = AfxGetApp() -> GetProfileInt("Progression
criteria","EdgeP",1);

      // Standard initialization
      // If you are not using these features and wish to reduce
the size
      //  of your final executable, you should remove from the
following
      //  the specific initialization routines you do not need.

      SetDialogBkColor(RGB(0,0,255),RGB(255,255,255));        //
Set dialog background color to BLUE, CONTROLS WHITE
      LoadStdProfileSettings();  // Load standard INI file options
(including MRU)

      // Register the application's document templates.  Document
templates
      //  serve as the connection between documents, frame windows
and views.

      CMultiDocTemplate* pDocTemplate;
      pDocTemplate = new CMultiDocTemplate(
            IDR_PROWINTYPE,
```

```
            RUNTIME_CLASS(CProwin01Doc),
            RUNTIME_CLASS(CSplitFrame),
            //RUNTIME_CLASS(CMDIChildWnd),        // standard MDI
child frame
            //RUNTIME_CLASS(CLeftView),
            RUNTIME_CLASS(CProwin01View));
      AddDocTemplate(pDocTemplate);

      // create main MDI Frame window
      CMainFrame* pMainFrame = new CMainFrame;
      if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
            return FALSE;
      m_pMainWnd = pMainFrame;

    // Make popout status bar
      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);
    pStatus->SetPaneInfo(0,AFX_IDW_STATUS_BAR,SBPS_POPOUT |
SBPS_STRETCH,300);

      // enable file manager drag/drop and DDE Execute open
      EnableShellOpen();
      RegisterShellFileTypes();

      // simple command line parsing
      if (m_lpCmdLine[0] == '\0')
      {
            // create a new (empty) document
            //OnFileNew();
      }
      else
      {
            // open an existing document
            OpenDocumentFile(m_lpCmdLine);
      }

      m_pMainWnd->DragAcceptFiles(FALSE);
      // The main window has been initialized, so show and update
it.
      pMainFrame->ShowWindow(SW_SHOWMAXIMIZED);
      pMainFrame->UpdateWindow();

    /*dlg.m_nOpenDatabase = 0;                        //welcome
dialog
```

```
    if (dlg.DoModal()==IDOK)
      if (dlg.m_nOpenDatabase == 0)OnFileNew();
      else OnFileOpen();*/
      EndWaitCursor();
    // /*cripple*/AfxMessageBox("This is a demonstration version
only.\nYou can only open the database DEMO.TXT\n");
      OnFileNew();
      return TRUE;
}

void CProwin01App::DoTitleWindow()
{     CFrameWnd* pTitleWnd = new CFrameWnd;
      CTime start_time = CTime::GetCurrentTime();
      CTime end_time;
      CTimeSpan interval;
      LONG elapsed;
      int x = ::GetSystemMetrics(SM_CXSCREEN);
      int y = ::GetSystemMetrics(SM_CYSCREEN);
      pTitleWnd -> Create(AfxRegisterWndClass(CS_BYTEALIGNWINDOW
,0,0,0),"Title window", WS_BORDER | WS_POPUP | WS_VISIBLE |
WS_DISABLED ,
            CRect(x/2-260,y/2-158,x/2-260+518,y/2-
158+315),NULL,NULL,0,NULL);
      CClientDC TitleDC(pTitleWnd)  ;
      CDC* pDisplayMemDC = new CDC;
      CBitmap* pBitmap = new CBitmap;
      pBitmap -> LoadBitmap(IDB_LOGO_BITMAP);
      pDisplayMemDC -> CreateCompatibleDC(&TitleDC);
      pDisplayMemDC -> SelectObject(pBitmap);
      TitleDC.BitBlt(0,0,518,315,pDisplayMemDC,0,0,SRCCOPY);

/*    TitleDC.SetBkColor(RGB(192,192,192));
      TitleDC.SetTextColor(RGB(0,0,255));
      TitleDC.TextOut(290,50,"DEMONSTRATION COPY"); */

      pTitleWnd -> ShowWindow(SW_SHOW);
      pTitleWnd -> UpdateWindow();
      //::sndPlaySound("welcome.wav",SND_ASYNC);
      do
            {end_time = CTime::GetCurrentTime();
             interval = end_time - start_time;
             elapsed = interval.GetTotalSeconds();
      }
      while (elapsed < 5L);
```

```
    pTitleWnd -> ReleaseDC(pDisplayMemDC);
    pTitleWnd -> DestroyWindow();
    delete pDisplayMemDC;
    delete pBitmap;           //delete m_pTitleDC is in
CProwin01App::ExitInstance();

}


/////////////////////////////////////////////////////////////////////
///////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
      CAboutDlg();

// Dialog Data
      //{{AFX_DATA(CAboutDlg)
      enum { IDD = IDD_ABOUTBOX };
      //}}AFX_DATA

// Implementation
protected:
      virtual void DoDataExchange(CDataExchange* pDX);    //
DDX/DDV support
      //{{AFX_MSG(CAboutDlg)
      afx_msg void OnVisButton();
      afx_msg void OnFredButton();
      afx_msg void OnDaveButton();
      afx_msg void OnRefsButton();
      virtual BOOL OnInitDialog();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
      //{{AFX_DATA_INIT(CAboutDlg)
      //}}AFX_DATA_INIT
}
```

```
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
	ON_BN_CLICKED(IDC_VIS_BUTTON, OnVisButton)
	ON_BN_CLICKED(IDC_FRED_BUTTON, OnFredButton)
	ON_BN_CLICKED(IDC_DAVE_BUTTON, OnDaveButton)
	ON_BN_CLICKED(IDC_REFS_BUTTON, OnRefsButton)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CProwin01App::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////
////////////
// VB-Event registration
// (calls to AfxRegisterVBEvent will be placed here by
ClassWizard)

//{{AFX_VBX_REGISTER_MAP()
//}}AFX_VBX_REGISTER_MAP

/////////////////////////////////////////////////////////////////////////
////////////
// CProwin01App commands

void CProwin01App::OnFileNew()
{
	char buffer[MAX_INFILE_LENGTH]="";
	CGetPtDialog* p_GetPtDialog;
	CStdioFile* p_infile = new CStdioFile;
	CString strInfile;
```

```
      CFileDialog select_database_dlg(TRUE,"txt",NULL,OFN_READONLY
| OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST ,"ProWin databases
(*.txt)|*.txt||",NULL);
      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);

                  select_database_dlg.m_ofn.lpstrTitle = "Open
Database";
                  if (select_database_dlg.DoModal()==IDOK)
                        {if (p_infile ->
Open(select_database_dlg.m_ofn.lpstrFile, CFile::modeRead |
CFile::typeText))
                              {p_infile ->
ReadString(buffer,MAX_INFILE_LENGTH);
                                    if
(!_fstrcmp(buffer,COLUMN_NAMES_STRING) /*cripple && p_infile ->
GetLength() == 141903*/)
                                          {strInfile =
select_database_dlg.m_ofn.lpstrFile;
                                          pStatus-
>SetPaneText(0,"Opening database " + strInfile,TRUE);
                                          pStatus->UpdateWindow();
                                          p_GetPtDialog = new
CGetPtDialog;

      m_pGetPtDialogArray.Add(p_GetPtDialog);
                                          p_GetPtDialog ->
m_nDatabaseIndex = m_nGetPtDialogIndex++;
                                          p_GetPtDialog -> m_pFile
= p_infile;
                                          p_GetPtDialog ->
GetNames();
                                          p_GetPtDialog ->
Create();
                                          p_GetPtDialog ->
SetWindowText("DATABASE: " + strInfile);
                                          pStatus-
>SetPaneText(0,"For Help, press F1",TRUE);
                                          pStatus->UpdateWindow();

                                    }
                                    else
{//::sndPlaySound("incorrec.wav",SND_ASYNC);
```

```
      sprintf(buffer,"\"%s\"\nSorry, this file is not a ProWin
database",select_database_dlg.m_ofn.lpstrFile);

      AfxMessageBox(buffer,MB_ICONSTOP | MB_OK);
                                                p_infile ->
Close();
                                                delete p_infile;
                                    }

                          }
                          else AfxMessageBox("Cannot open
database",MB_ICONSTOP | MB_OK);

                    }

}

void CProwin01App::DoNewDocument()
{
      CWinApp::OnFileNew();
}

int CProwin01App::ExitInstance()
{
      CGetPtDialog* pGetPtDialog;
      for (int i = 0; i < m_pGetPtDialogArray.GetSize(); i++)
            {if (pGetPtDialog =
(CGetPtDialog*)m_pGetPtDialogArray[i])
                  pGetPtDialog -> Destroy();
      }
      m_pGetPtDialogArray.RemoveAll();
      return CWinApp::ExitInstance();
}

void CAboutDlg::OnVisButton()
{
      CAddressDialog dlg;
      dlg.m_Address = "A.C.Viswanathan FRCOphth\nInternational
Glaucoma Association Research Fellow\n\
Department of Visual Science\n11 - 43 Bath Street\nLondon EC1V
9EL\n\nTelephone 00 44 171 608 6833\nFax 00 44 171 608 6834\
\nemail a.viswanathan@ucl.ac.uk";
      dlg.DoModal();
```

```
}

void CAboutDlg::OnFredButton()
{
      CAddressDialog dlg;
      dlg.m_Address = "F.W.Fitzke PhD\nHead of Department of
Physiological Optics\n\
Department of Visual Science\n11 - 43 Bath Street\nLondon EC1V
9EL\n\nTelephone/Fax 00 44 171 608 6834\n\
\nemail smgx510@ucl.ac.uk";
      dlg.DoModal();

}

void CAboutDlg::OnDaveButton()
{
      CAddressDialog dlg;
      dlg.m_Address = "D.P.Crabb MSc\nResearch Fellow\n\
Department of Visual Science\n11 - 43 Bath Street\nLondon EC1V
9EL\n\nTelephone 00 44 171 608 6833\nFax 00 44 171 608 6834\
\nemail d.crabb@ucl.ac.uk";
      dlg.DoModal();

}

void CAboutDlg::OnRefsButton()
{
      CString string("References");
      AfxGetApp() -> WinHelp((DWORD) (LPCSTR) string, HELP_KEY);
      OnOK();
      /*CRefDialog dlg;
      dlg.m_strRefs = "1. Fitzke FW, Crabb DP, McNaught AI, Edgar
DF, Hitchings RA. Image processing of computerised visual field
data. Br J Ophthalmol. 1995 ; 79 : 207-212.\

                  \
2. Crabb DP, Fitzke FW, Edgar DF, McNaught AI, Wynn HP. New
approach to estimating the variability in visual field data using
an image processing technique. Br J Ophthalmol. 1995 ; 79 : 213-
217.\

                  \
```

```
3. Crabb DP, McNaught AI, Fitzke FW, Hitchings RA. Spatially
enhanced modelling of sensitivity decay in low tension glaucoma.
In: Mills RP and Wall M (eds). Perimetry Update 1994/95.
Amsterdam: Kugler , 1995 ;73-81.\

                \
4. McNaught AI, Crabb DP, Fitzke FW, Hitchings RA. Modelling
series of visual fields to detect progression in normal tension
glaucoma. Graefes Arch Clin Exp Ophthalmol. 1995; 233 : 750-755.\

                \
5. Fitzke FW, Hitchings RA, Poinoosawmy D, McNaught AI, Crabb DP.
Analysis of visual field progression in glaucoma. Br J Ophthalmol.
1996; 80 : 40-48.\

                \
6. McNaught AI, Crabb DP, Fitzke FW, Hitchings RA. Visual field
progression: comparison of Humphrey Statpac2 and pointwise linear
regression. Graefes Arch Clin Exp Ophthalmol. 1996 (in press).\
";
      dlg.DoModal(); */

}

BOOL CAboutDlg::OnInitDialog()
{
      int x = ::GetSystemMetrics(SM_CXSCREEN);
      int y = ::GetSystemMetrics(SM_CYSCREEN);

      CRect dlgRect;
      GetWindowRect(&dlgRect);
      CDialog::OnInitDialog();

      SetWindowPos(&CWnd::wndTopMost,x/2-dlgRect.Width()/2,y/2-
dlgRect.Height()/2,0,0,SWP_NOSIZE );

      return TRUE;  // return TRUE  unless you set the focus to a
control
}
```

```
; prowin01.def : Declares the module parameters for the
application.

NAME          PROWIN01
DESCRIPTION   'PROGRESSOR for Windows'
EXETYPE       WINDOWS

CODE          PRELOAD MOVEABLE DISCARDABLE
DATA          PRELOAD MOVEABLE MULTIPLE

HEAPSIZE      1024    ; initial heap size
; Stack size is passed as argument to linker's /STACK option
```

```
// prowin01.h : main header file for the PROWIN01 application
//

#ifndef __AFXWIN_H__
      #error include 'stdafx.h' before including this file for PCH
#endif
#include "mmsystem.h"          // for sound
#include "resource.h"       // main symbols
////////////////////////////////////////////////////////////////////
///////////
// CProwin01App:
// See prowin01.cpp for the implementation of this class
//

class CProwin01App : public CWinApp
{     friend class CGetPtDialog;
public:
      CProwin01App();

// Overrides
      virtual BOOL InitInstance();
      virtual int ExitInstance();

// Data members
      BOOL m_bLegendDialogDisplayed;
      float m_tvalue[60][6];
      CString m_strMinInnerSlope;
      CString m_strMinEdgeSlope;
      int m_nInnerPValue;
      int m_nEdgePValue;

      CObArray m_pGetPtDialogArray;
      int m_nGetPtDialogIndex;
      int m_nActiveDatabaseIndex;


// Implementation
      void DoTitleWindow();
      void DoNewDocument();

      //{{AFX_MSG(CProwin01App)
      afx_msg void OnAppAbout();
      afx_msg void OnFileNew();
      //}}AFX_MSG
```

```
      DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////
///////////
// VB-Event extern declarations

//{{AFX_VBX_REGISTER()
//}}AFX_VBX_REGISTER

/////////////////////////////////////////////////////////////////////////
///////////
```

```
// prowivw.cpp : implementation of the CProwin01View class
//

#include "stdafx.h"
#include "prowin01.h"
#include "prowidoc.h"
#include "prowivw.h"
#include "datedlg.h"
#include "legdlg.h"
#include "progcrit.h"
#include "progind.h"
#include "mainfrm.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

#define BOX_DIMENSION 70
#define BOX_MARGIN 10
#define GRAY_BOX_DIMENSION 30

/////////////////////////////////////////////////////////////////////////
///////////
// CProwin01View

IMPLEMENT_DYNCREATE(CProwin01View, CScrollView)

BEGIN_MESSAGE_MAP(CProwin01View, CScrollView)
      //{{AFX_MSG_MAP(CProwin01View)
      ON_COMMAND(ID_EYE_RIGHT, OnEyeRight)
      ON_UPDATE_COMMAND_UI(ID_EYE_RIGHT, OnUpdateEyeRight)
      ON_COMMAND(ID_EYE_LEFT, OnEyeLeft)
      ON_UPDATE_COMMAND_UI(ID_EYE_LEFT, OnUpdateEyeLeft)
      ON_COMMAND(ID_VIEW_DATES, OnViewDates)
      ON_COMMAND(ID_VIEW_LEGEND, OnViewLegend)
      ON_COMMAND(ID_VIEW_PROGRESSING_POINTS,
OnViewProgressingPoints)
      ON_UPDATE_COMMAND_UI(ID_VIEW_PROGRESSING_POINTS,
OnUpdateViewProgressingPoints)
      ON_COMMAND(ID_VIEW_PROG_CRIT, OnViewProgCrit)
      ON_UPDATE_COMMAND_UI(ID_VIEW_LEGEND, OnUpdateViewLegend)
      ON_UPDATE_COMMAND_UI(ID_VIEW_DATES, OnUpdateViewDates)
      ON_WM_LBUTTONDOWN()
```

```
	ON_WM_SETCURSOR()
	ON_WM_MOUSEMOVE()
	ON_COMMAND(ID_FILTER_GAUSSIAN, OnFilterGaussian)
	ON_UPDATE_COMMAND_UI(ID_FILTER_GAUSSIAN,
OnUpdateFilterGaussian)
	ON_COMMAND(ID_WINDOW_CLOSEALL, OnWindowCloseall)
	ON_COMMAND(ID_VIEW_PROGRESSION_INDICES,
OnViewProgressionIndices)
	ON_UPDATE_COMMAND_UI(ID_VIEW_PROGRESSION_INDICES,
OnUpdateViewProgressionIndices)
	ON_COMMAND(ID_VIEW_GRAYSCALE, OnViewGrayscale)
	ON_UPDATE_COMMAND_UI(ID_VIEW_GRAYSCALE,
OnUpdateViewGrayscale)
	ON_COMMAND(ID_EYE_BINOCULAR, OnEyeBinocular)
	ON_UPDATE_COMMAND_UI(ID_EYE_BINOCULAR, OnUpdateEyeBinocular)
	ON_COMMAND(ID_VIEW_ESTERMAN_DEFECTS, OnViewEstermanDefects)
	ON_UPDATE_COMMAND_UI(ID_VIEW_ESTERMAN_DEFECTS,
OnUpdateViewEstermanDefects)
	ON_COMMAND(ID_EDIT_COPY, OnEditCopy)
	ON_COMMAND(ID_EDIT_COPY_SCREEN, OnEditCopyScreen)
	//}}AFX_MSG_MAP
	// Standard printing commands
	ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_PREVIEW,
CScrollView::OnFilePrintPreview)
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////
///////////
// CProwin01View construction/destruction

CProwin01View::CProwin01View()
{
	m_bRightEye = TRUE;
	m_bBinocular = FALSE;
	m_bShowProgressingPoints = FALSE;
	m_bShowGrayscale = FALSE;
	m_bEsterman = FALSE;
	m_LightBlueBrush.CreateSolidBrush(RGB(0,0,255));
	m_LightRedBrush.CreateSolidBrush(RGB(255,0,0));
	m_DarkRedBrush.CreateSolidBrush(RGB(128,0,0));
	m_MagentaBrush.CreateSolidBrush(RGB(255,0,255));
	m_YellowBrush.CreateSolidBrush(RGB(255,255,0));
	m_GrayBrush.CreateSolidBrush(RGB(128,128,128));
```

```
	m_BlueGreenBrush.CreateSolidBrush(RGB(0,128,128));
	m_OliveBrush.CreateSolidBrush(RGB(128,128,0));
	m_DarkGreenBrush.CreateSolidBrush(RGB(0,128,0));
	m_LightGreenBrush.CreateSolidBrush(RGB(0,255,0));
	m_RedHatchBrush.CreateHatchBrush(HS_DIAGCROSS,RGB(255,0,0));
	m_DarkBlueBrush.CreateSolidBrush(RGB(0,0,128));
	m_WhiteBrush.CreateSolidBrush(RGB(255,255,255));
	m_NameColor = RGB(128,128,128);
	m_tracker.SetRectEmpty();

	for (int i = 0; i < 10; i++)
		{sprintf(m_strGrayBrushName,"GRAYBRUSH%d",i);
		m_GrayscaleBitmap[i].LoadBitmap(m_strGrayBrushName);
		TRACE("\tLoading %s\n",m_strGrayBrushName);

	m_GrayscaleBrush[i].CreatePatternBrush(&m_GrayscaleBitmap[i]
);
	}
}

CProwin01View::~CProwin01View()
{
}

////////////////////////////////////////////////////////////////////
///////////
// CProwin01View drawing

void CProwin01View::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo /* =
NULL*/)
{	CRect ClientRect;
	GetClientRect(&ClientRect);
	if (!pDC -> IsPrinting())
		{pDC -> SetMapMode(MM_ISOTROPIC);
		pDC -> SetWindowExt(1000,1000);
		pDC ->
SetViewportExt(ClientRect.right,ClientRect.bottom);
		pDC ->
SetViewportOrg(ClientRect.right/2,ClientRect.bottom/2);
	}

}

void CProwin01View::OnDraw(CDC* pDC)
```

```
{
      CProwin01Doc* pDoc = GetDocument();
      ASSERT_VALID(pDoc);
      int start_field;
      int bar_width;
      int sensitivity, l_sensitivity, r_sensitivity;
      int i,j,x,y;
      CFont NameFont;
      CFont EyeFont;
      CPen AxisPen(PS_DOT,1,RGB(255,255,255)) ;
      CPen Central20Pen(PS_INSIDEFRAME,1,RGB(0,0,255)) ;
      CString strInnerSlope;
      CString strInnerP;
      CString strEdgeSlope;
      CString strEdgeP;
      CRect ClientRect;
      int nInnerP;
      int nEdgeP;
      int lbsel;
      int nBinocLeftCount, nBinocRightCount;
      CBrush* pOldBrush = pDC -> SelectObject(&m_LightRedBrush);

      if(!m_tracker.IsRectNull())
            {pDC -> LPtoDP(m_tracker);
            InvalidateRect(CRect(m_tracker.TopLeft()-
CSize(10,10),m_tracker.Size()+CSize(20,20)),
                  m_bShowGrayscale | m_bBinocular ? TRUE : FALSE);
            m_tracker.SetRectEmpty();
      }

      NameFont.CreateFont(50,0,0,0,400,FALSE,FALSE,0,ANSI_CHARSET,
OUT_DEFAULT_PRECIS,CLIP_DEFAULT_PRECIS,DEFAULT_QUALITY,DEFAULT_PIT
CH | FF_ROMAN, NULL);
      EyeFont.CreateFont(40,0,0,0,400,FALSE,FALSE,0,ANSI_CHARSET,O
UT_DEFAULT_PRECIS,CLIP_DEFAULT_PRECIS,DEFAULT_QUALITY,DEFAULT_PITC
H | FF_ROMAN, NULL);
      CFont* pOldFont = pDC -> SelectObject(&NameFont);
      GetClientRect(&ClientRect);
      pDC -> DPtoLP(ClientRect);

      if ( !(m_bShowGrayscale | m_bBinocular) )
            {if (!pDC -> IsPrinting())pDC ->
FillRect(ClientRect,&m_DarkBlueBrush);
            pDC -> SetBkColor (RGB(0,0,128));
```

```
            pDC -> SelectObject(&AxisPen);
            pDC -> MoveTo(0,450);
            pDC -> LineTo(0,-450);
            pDC -> MoveTo(-450,0);
            pDC -> LineTo(450,0);
      }

      if (pDC -> IsPrinting()&&(m_bShowGrayscale |
m_bBinocular))pDC->SetTextColor (RGB(0,0,0));
      else pDC->SetTextColor (m_NameColor);
      pDC -> TextOut(-500,-500,pDoc -> m_name);
      pDC->SetTextColor (m_bShowGrayscale | m_bBinocular ?
RGB(0,0,0) : RGB(255,255,255));
      pDC -> SelectObject(&EyeFont);

            if (!m_bBinocular)
                  {if (m_bRightEye)
                        {pDC -> TextOut(-500,-450,"Right Eye");
                        if (pDoc -> m_rightReanalyzed)
                              {if ( !m_bShowGrayscale )pDC ->
SelectObject(&NameFont);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,0));
                              pDC -> TextOut(-500,-
400,"REANALYZED");
                              if ( m_bShowGrayscale )pDC ->
ExcludeClipRect(-500,-400,-270,-360);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,255));
                        }
                        if (pDoc -> m_rightGaussianFiltered)
                              {if ( !m_bShowGrayscale )pDC ->
SelectObject(&NameFont);
                              if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(0,255,255));
                              pDC -> TextOut(-500,-
350,"FILTERED");
                              if ( m_bShowGrayscale )pDC ->
ExcludeClipRect(-500,-350,-330,-310);
                              pDC -> SelectObject(&EyeFont);
                              pDC -> TextOut(-500,-
310,"(Gaussian)");
                              if ( m_bShowGrayscale )pDC ->
ExcludeClipRect(-500,-310,-345,-270);
```

```
                        if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,255));
                  }
            }
            else
                  {pDC -> TextOut(-500,-450,"Left Eye  ");
                  if (pDoc -> m_leftReanalyzed)
                        {if ( !m_bShowGrayscale )pDC ->
SelectObject(&NameFont);
                        if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,0));
                        pDC -> TextOut(-500,-
400,"REANALYZED");
                        if (m_bShowGrayscale)pDC ->
ExcludeClipRect(-500,-400,-270,-360);
                        if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,255));
                  }
                  if (pDoc -> m_leftGaussianFiltered)
                        {if ( !m_bShowGrayscale )pDC ->
SelectObject(&NameFont);
                        if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(0,255,255));
                        pDC -> TextOut(-500,-
350,"FILTERED");
                        if ( m_bShowGrayscale )pDC ->
ExcludeClipRect(-500,-350,-330,-310);
                        pDC -> SelectObject(&EyeFont);
                        pDC -> TextOut(-500,-
310,"(Gaussian)");
                        if ( m_bShowGrayscale )pDC ->
ExcludeClipRect(-500,-310,-345,-270);
                        if ( !m_bShowGrayscale )pDC-
>SetTextColor (RGB(255,255,255));
                  }
            }
      }
      else
            {pDC -> TextOut(-500,-450,"Binocular");
            if (pDoc -> m_rightReanalyzed || pDoc ->
m_leftReanalyzed)
                  {pDC -> TextOut(-500,-400,"REANALYZED");
                  pDC -> ExcludeClipRect(-500,-400,-270,-
360);
```

```
                }
                if (pDoc -> m_rightGaussianFiltered || pDoc ->
m_leftGaussianFiltered)
                      {pDC -> TextOut(-500,-350,"FILTERED");
                      pDC -> ExcludeClipRect(-500,-350,-330,-
310);
                      pDC -> TextOut(-500,-310,"(Gaussian)");
                      pDC -> ExcludeClipRect(-500,-310,-345,-
270);
                }
          }

     for ( int row = 0; row < 12; row++)
          for ( int col = 0; col < 12; col++)
m_nMatrix[row][col] = MISSING_VALUE;

     x = -2*BOX_DIMENSION -3*BOX_MARGIN;
     y = -5*BOX_DIMENSION -9*BOX_MARGIN;

     strInnerSlope = ((CProwin01App*)AfxGetApp()) ->
m_strMinInnerSlope;
     strEdgeSlope = ((CProwin01App*)AfxGetApp()) ->
m_strMinEdgeSlope;
     nInnerP = ((CProwin01App*)AfxGetApp()) -> m_nInnerPValue;
     nEdgeP = ((CProwin01App*)AfxGetApp()) -> m_nEdgePValue;

     if (nInnerP == 0)strInnerP = "(p >= 0.1)";
     if (nInnerP == 1)strInnerP = "(p < 0.1 )";
     if (nInnerP == 2)strInnerP = "(p < 0.05)";
     if (nInnerP == 3)strInnerP = "(p < 0.01)";
     if (nInnerP == 4)strInnerP = "(p < 0.001)";
     if (nEdgeP == 0)strEdgeP = "(p >= 0.1)";
     if (nEdgeP == 1)strEdgeP = "(p < 0.1 )";
     if (nEdgeP == 2)strEdgeP = "(p < 0.05)";
     if (nEdgeP == 3)strEdgeP = "(p < 0.01)";
     if (nEdgeP == 4)strEdgeP = "(p < 0.001)";

     for ( i = 0; i < 76; i++)
          {if ( i == 4 )
               {x = -3*BOX_DIMENSION -5*BOX_MARGIN;
               y = -4*BOX_DIMENSION -7*BOX_MARGIN;
          }
          if ( i == 10 )
               {x = -4*BOX_DIMENSION -7*BOX_MARGIN;
```

```
                y = -3*BOX_DIMENSION -5*BOX_MARGIN;
          }
          if ( i == 18 )
                {x = -5*BOX_DIMENSION -9*BOX_MARGIN;
                y = -2*BOX_DIMENSION -3*BOX_MARGIN;
          }
          if ( i == 28 )
                {x = -5*BOX_DIMENSION -9*BOX_MARGIN;
                y = -BOX_DIMENSION -BOX_MARGIN;
          }
          if ( i == 38 )
                {x = -5*BOX_DIMENSION -9*BOX_MARGIN;
                y = BOX_MARGIN;
          }
          if ( i == 48 )
                {x = -5*BOX_DIMENSION -9*BOX_MARGIN;
                y = BOX_DIMENSION +3*BOX_MARGIN;
          }
          if ( i == 58 )
                {x = -4*BOX_DIMENSION -7*BOX_MARGIN;
                y = 2*BOX_DIMENSION +5*BOX_MARGIN;
          }
          if ( i == 66 )
                {x = -3*BOX_DIMENSION -5*BOX_MARGIN;
                y = 3*BOX_DIMENSION +7*BOX_MARGIN;
          }
          if ( i == 72 )
                {x = -2*BOX_DIMENSION -3*BOX_MARGIN;
                y = 4*BOX_DIMENSION +9*BOX_MARGIN;
          }

          m_BoxArray[i] = CRect(x - BOX_MARGIN/2, y -
BOX_MARGIN/2 , x + BOX_DIMENSION + BOX_MARGIN/2, y + BOX_DIMENSION
+ BOX_MARGIN/2);

          if(!m_bBinocular)
                {if (m_bRightEye)
                      {if (m_nCount = pDoc ->
m_rightorderArray.GetSize())
                            {if (pDoc -> m_bDateDialogDisplayed)
                                  {lbsel = (int)pDoc ->
m_pDateDialog -> m_nRightSelCount;
                                  if (lbsel == 1)
```

```
                                        {m_nCount = (int)pDoc ->
m_pDateDialog -> m_nRightRestrictedDates[0]  + 1;

      OnMouseMove(m_nFlags,m_mousePos);
                                    }
                               }
                               start_field = 0;
                               if (m_nCount > MAX_FIELDS)
start_field = m_nCount - MAX_FIELDS;  // shows the last MAX_FIELDS
fields
                               bar_width =
BOX_DIMENSION/MAX_FIELDS;                              //bar_width
= BOX_DIMENSION/m_nCount; for scaling to fit
                               for ( j = start_field; j < m_nCount;
j++)
                                    if( (sensitivity = (WORD)pDoc
-> m_rightpointArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(j)))
                                        != MISSING_VALUE)
                                        {if (m_bShowGrayscale
&&( j == m_nCount - 1 /*the last field of the series*/) )

      AddToMatrix(&i,&sensitivity);

                                        if
(m_bShowProgressingPoints && ( j == m_nCount - 1 /*the last field
of the series*/) )
                                            if
(IsProgressing(&i,&j,pDoc,&m_bRightEye))
                                                {pDC ->
FillRect( m_BoxArray[i], &m_RedHatchBrush);
                                                if
(m_bShowGrayscale)pDC -> ExcludeClipRect(m_BoxArray[i]);
                                            }

                                        if
(!m_bShowProgressingPoints && !m_bShowGrayscale)
                                            pDC->
FillRect(CRect(x + (j-start_field)*bar_width,y,x + (j-
start_field+1)*bar_width,
                                            y + (300 -
sensitivity)*BOX_DIMENSION/300),GetBrush(&i, &j,pDoc));

                                    }
```

```
                        }
                        else if (!i)
                              {pDC -> SelectObject(&NameFont);
                              pDC->SetTextColor (RGB(255,0,0));
                              pDC -> TextOut(-200,-50,"NO VALID
FIELDS" );
                        }

                  }
                  else
                        {if (m_nCount = pDoc ->
m_leftorderArray.GetSize())
                              {if (pDoc -> m_bDateDialogDisplayed)
                                    {lbsel = (int)pDoc ->
m_pDateDialog -> m_nLeftSelCount;
                                    if (lbsel == 1)
                                          {m_nCount = (int)pDoc ->
m_pDateDialog -> m_nLeftRestrictedDates[0]  + 1;

      OnMouseMove(m_nFlags,m_mousePos);
                                    }
                              }
                              start_field = 0;
                              if (m_nCount > MAX_FIELDS)
start_field = m_nCount - MAX_FIELDS;  // shows the last MAX_FIELDS
fields
                              bar_width =
BOX_DIMENSION/MAX_FIELDS;                              //bar_width
= BOX_DIMENSION/m_nCount; for scaling to fit
                              for ( j = start_field; j < m_nCount;
j++)
                                    if( (sensitivity = (WORD)pDoc
-> m_leftpointArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(j)))
                                          != MISSING_VALUE)
                                          {if (m_bShowGrayscale
&&( j == m_nCount - 1 /*the last field of the series*/) )

      AddToMatrix(&i,&sensitivity);

                                          if
(m_bShowProgressingPoints && ( j == m_nCount - 1 /*the last field
of the series*/) )
```

```
                                                    if
(IsProgressing(&i,&j,pDoc,&m_bRightEye))
                                                          {pDC ->
FillRect( m_BoxArray[i], &m_RedHatchBrush);
                                                          if
(m_bShowGrayscale)pDC -> ExcludeClipRect(m_BoxArray[i]);
                                                    }

                                               if
(!m_bShowProgressingPoints && !m_bShowGrayscale)
                                                    pDC->
FillRect(CRect(x + (j-start_field)*bar_width,y,x + (j-
start_field+1)*bar_width,
                                                    y + (300 -
sensitivity)*BOX_DIMENSION/300),GetBrush(&i, &j,pDoc));

                                         }

                        }
                        else if (!i)
                              {pDC -> SelectObject(&NameFont);
                              pDC->SetTextColor (RGB(255,0,0));
                              pDC -> TextOut(-200,-50,"NO VALID
FIELDS" );
                        }

                  }

            }
            else
                  {if ((nBinocLeftCount = pDoc ->
m_leftorderArray.GetSize()) &&
                        (nBinocRightCount = pDoc ->
m_rightorderArray.GetSize()))
                        {if (pDoc -> m_bDateDialogDisplayed)
                              {lbsel = (int)pDoc -> m_pDateDialog
-> m_nLeftSelCount;
                              if (lbsel == 1)
                                    {nBinocLeftCount = (int)pDoc -
> m_pDateDialog -> m_nLeftRestrictedDates[0]  + 1;
                              }
                              lbsel = (int)pDoc -> m_pDateDialog -
> m_nRightSelCount;
                              if (lbsel == 1)
```

```
                                    {nBinocRightCount = (int)pDoc
-> m_pDateDialog -> m_nRightRestrictedDates[0]  + 1;
                                    }
                              }

                              l_sensitivity = (WORD)pDoc ->
m_leftpointArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(nBinocLeftCount - 1));
                              r_sensitivity = (WORD)pDoc ->
m_rightpointArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(nBinocRightCount - 1));
                              sensitivity = l_sensitivity >
r_sensitivity ? l_sensitivity : r_sensitivity;
                              AddToMatrix(&i,&sensitivity);
                              if (m_bEsterman && sensitivity < 100 )
                                    {pDC -> FillRect( m_BoxArray[i],
&m_DarkBlueBrush);
                                    pDC -> Ellipse( m_BoxArray[i]);
                                    pDC -> SelectObject(&EyeFont);
                                    pDC->SetTextColor (RGB(0,0,0));
                                    pDC -> TextOut(x+7,y+15,"<10");
                                    pDC ->
ExcludeClipRect(m_BoxArray[i]);
                              }
                        }
                  }

                  x += BOX_DIMENSION + 2*BOX_MARGIN;

            }

            if (m_bShowGrayscale | m_bBinocular)Interpolate(pDC);

            if (m_bBinocular && m_bEsterman)
                  {pDC -> SelectStockObject(HOLLOW_BRUSH);
                  pDC -> SelectObject(&Central20Pen);
                  pDC -> Ellipse(-300,-300,300,300);
            }

            if (m_bShowProgressingPoints && !m_bBinocular)
                  {pDC -> SelectObject(&EyeFont);
                  pDC->SetTextColor (m_bShowGrayscale ? RGB(0,0,0) :
RGB(255,255,255));
```

```
            pDC -> TextOut(-500,460,"Inner: < -" + strInnerSlope +
" dB / yr " + strInnerP);
            pDC -> TextOut(0,460,"Edge: < -" + strEdgeSlope + " dB
/ yr " + strEdgeP );
      }

      if ( !pDoc -> m_bDateDialogDisplayed )
//for progression indices dialog
            {pDoc -> m_nLeftCount = pDoc ->
m_leftorderArray.GetSize(); //for progression indices dialog
            pDoc -> m_nRightCount = pDoc ->
m_rightorderArray.GetSize();//for progression indices dialog
      }
      pDC -> SelectStockObject(NULL_PEN);
      pDC -> SelectObject(pOldFont);
      pDC -> SelectObject(pOldBrush);
}

#pragma optimize ("",off)
CBrush* CProwin01View::GetBrush(int* point, int* field,
CProwin01Doc* pDoc)
{
      BYTE pvalue;
      CString slopeString;
      long double slope;

      if (m_bRightEye)pvalue = (BYTE)pDoc ->
m_rightpvalueArray.GetAt(*point + 76*pDoc ->
m_rightorderArray.GetAt(*field));
            else pvalue = (BYTE)pDoc ->
m_leftpvalueArray.GetAt(*point + 76*pDoc ->
m_leftorderArray.GetAt(*field));

      if (m_bRightEye)slopeString = (CString)pDoc ->
m_rightslopeArray.GetAt(*point + 76*pDoc ->
m_rightorderArray.GetAt(*field));
            else slopeString = (CString)pDoc ->
m_leftslopeArray.GetAt(*point + 76*pDoc ->
m_leftorderArray.GetAt(*field));

      slope = _atold((const char*)slopeString);
      if ( (fabsl(slope) <= 1) && (pvalue != 10)/* ie excluded in
Doc*/ ) return &m_GrayBrush; /////////cutoff for 'flat'
```

```
      switch (pvalue)
            {case 11: return &m_WhiteBrush;
            case 10: return &m_LightBlueBrush;
            case 9: return &m_LightRedBrush;
            case 8: return &m_DarkRedBrush;
            case 7: return &m_MagentaBrush;
            case 6: return &m_YellowBrush;
            case 5: return &m_GrayBrush;
            case 4: return &m_BlueGreenBrush;
            case 3: return &m_OliveBrush;
            case 2: return &m_DarkGreenBrush;
            case 1: return &m_LightGreenBrush;
            default: return &m_LightBlueBrush;
      }

}
#pragma optimize ("",on)

BOOL CProwin01View::IsProgressing(int* point,int*
last_field,CProwin01Doc* pDoc,BOOL* pbRightEye)
{     // characteristics of the point from the Doc
      BOOL retval = FALSE;
      BYTE pvalue;
      CString slopeString;
      long double slope;

      // progression criteria from the App
      int p_crit;
      CString slope_critString;
      long double slope_crit;

      if (*pbRightEye)pvalue = (BYTE)pDoc ->
m_rightpvalueArray.GetAt(*point + 76*pDoc ->
m_rightorderArray.GetAt(*last_field));
            else pvalue = (BYTE)pDoc ->
m_leftpvalueArray.GetAt(*point + 76*pDoc ->
m_leftorderArray.GetAt(*last_field));

      if (*pbRightEye)slopeString = (CString)pDoc ->
m_rightslopeArray.GetAt(*point + 76*pDoc ->
m_rightorderArray.GetAt(*last_field));
            else slopeString = (CString)pDoc ->
m_leftslopeArray.GetAt(*point + 76*pDoc ->
m_leftorderArray.GetAt(*last_field));
```

```
      slope = _atold((const char*)slopeString);
      if (slope < 0)
            {// Get appropriate progression criteria
            if ( *point < 5 || *point == 9 || *point == 10 ||
*point == 17 || *point == 18 || *point == 27 || *point == 28
            || *point == 37|| *point == 38 || *point == 47 ||
*point == 48 || *point == 57 || *point == 58 || *point == 65
            || *point == 66|| *point > 70 )
                  {p_crit = ((CProwin01App*)AfxGetApp()) ->
m_nEdgePValue;
                  slope_critString = ((CProwin01App*)AfxGetApp())
-> m_strMinEdgeSlope;
            }
            else
                  {p_crit = ((CProwin01App*)AfxGetApp()) ->
m_nInnerPValue;
                  slope_critString = ((CProwin01App*)AfxGetApp())
-> m_strMinInnerSlope;

            }
            slope_crit = _atold((const char*)slope_critString);

            // Compare
            if (fabsl(slope) > slope_crit )
                  {if (p_crit == 0 && (pvalue == 6 || pvalue == 7
|| pvalue == 8 || pvalue == 9 || pvalue == 11) )retval = TRUE;
                  if (p_crit == 1 && (pvalue == 7 || pvalue == 8
|| pvalue == 9 || pvalue == 11) )retval = TRUE;
                  if (p_crit == 2 && (pvalue == 8 || pvalue == 9
|| pvalue == 11) )retval = TRUE;
                  if (p_crit == 3 && (pvalue == 9 || pvalue ==
11))retval = TRUE;
                  if (p_crit == 4 && pvalue == 11)retval = TRUE;
            }
      }

    return retval;
}

void CProwin01View::OnActivateView(BOOL bActivate, CView*
pActivateView, CView* pDeactiveView)
{
      CClientDC dc(this);
```

```
	CRect InvalidRect(-500,-500,500,-450);
	OnPrepareDC(&dc);
	dc.LPtoDP(InvalidRect);
	if (bActivate)
		{if ( !(m_bShowGrayscale || m_bBinocular) )m_NameColor
= RGB(255,255,255);
		else m_NameColor = RGB(0,0,0);
	}
	else
		{m_NameColor = RGB(128,128,128);
		if(!m_tracker.IsRectNull())
			{dc.LPtoDP(m_tracker);
			InvalidateRect(CRect(m_tracker.TopLeft()-
CSize(10,10),m_tracker.Size()+CSize(20,20)),
				(m_bShowGrayscale) ? TRUE : FALSE);
			m_tracker.SetRectEmpty();
		}
	}

	InvalidateRect(InvalidRect,FALSE);
}

void CProwin01View::OnInitialUpdate()
{
	CScrollView::OnInitialUpdate();

	CSize sizeTotal;
	// TODO: calculate the total size of this view
	sizeTotal.cx = sizeTotal.cy = 100;
	SetScrollSizes(MM_TEXT, sizeTotal);
}

void CProwin01View::OnUpdate(CView* pView, LPARAM lHint, CObject*
pHint)
{
	if(!m_bBinocular)
		{if (lHint == 1 && m_bRightEye)Invalidate(FALSE);
		if (lHint == 2 && !m_bRightEye)Invalidate(FALSE);
		if (lHint == 3)Invalidate(FALSE);
	}
	else Invalidate(FALSE);
}
```

```
/////////////////////////////////////////////////////////////////////////
///////////
// CProwin01View printing

BOOL CProwin01View::OnPreparePrinting(CPrintInfo* pInfo)
{
      if (m_bBinocular) pInfo -> SetMaxPage(1);
      else pInfo -> SetMaxPage(2);
      BOOL bRet = DoPreparePrinting(pInfo);
      if (m_bBinocular)pInfo -> m_nNumPreviewPages = 1;
      else pInfo -> m_nNumPreviewPages = 2;
      return bRet;
}

void CProwin01View::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo*
/*pInfo*/)
{
      // TODO: add extra initialization before printing
}

void CProwin01View::OnEndPrinting(CDC* /*pDC*/, CPrintInfo*
/*pInfo*/)
{
      // TODO: add cleanup after printing
}

/////////////////////////////////////////////////////////////////////////
///////////
// CProwin01View diagnostics

#ifdef _DEBUG
void CProwin01View::AssertValid() const
{
      CScrollView::AssertValid();
}

void CProwin01View::Dump(CDumpContext& dc) const
{
      CScrollView::Dump(dc);
}

CProwin01Doc* CProwin01View::GetDocument() // non-debug version is
inline
{
```

```
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CProwin01Doc)));
	return (CProwin01Doc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////
////////////
// CProwin01View message handlers


void CProwin01View::OnEyeRight()
{
	m_bRightEye = TRUE;
	Invalidate(FALSE);
}

void CProwin01View::OnUpdateEyeRight(CCmdUI* pCmdUI)
{
	if(m_bBinocular)pCmdUI -> Enable(FALSE);
	else pCmdUI -> Enable(!m_bRightEye);
}

void CProwin01View::OnEyeLeft()
{
	m_bRightEye = FALSE;
	Invalidate(FALSE);
}

void CProwin01View::OnUpdateEyeLeft(CCmdUI* pCmdUI)
{
	if(m_bBinocular)pCmdUI -> Enable(FALSE);
	else pCmdUI -> Enable(m_bRightEye);
}

void CProwin01View::OnViewDates()
{
	CProwin01Doc* pDoc = GetDocument();
	if (!(pDoc -> m_bDateDialogDisplayed))
		{pDoc -> m_pDateDialog -> m_pDoc = pDoc;
		pDoc -> m_pDateDialog -> Create();
		pDoc -> m_bDateDialogDisplayed = TRUE;
	}
}
```

```
void CProwin01View::OnViewLegend()
{
      if (!((CProwin01App*)AfxGetApp()) ->
m_bLegendDialogDisplayed)
            {CLegendDialog* pLegendDialog = new CLegendDialog;
            pLegendDialog -> Create();
            ((CProwin01App*)AfxGetApp()) ->
m_bLegendDialogDisplayed = TRUE;
      }

      //delete pLegendDialog is done in CLegendDialog::OnCancel()
as: delete this;
}

void CProwin01View::OnViewProgressingPoints()
{
      m_bShowProgressingPoints = m_bShowProgressingPoints ^ 1;
      Invalidate(FALSE);
}

void CProwin01View::OnUpdateViewProgressingPoints(CCmdUI* pCmdUI)
{
      if(m_bBinocular)pCmdUI -> Enable(FALSE);
      else pCmdUI -> SetCheck(m_bShowProgressingPoints);
}


void CProwin01View::OnViewProgCrit()
{   CProwin01Doc* pDoc = GetDocument();
      CProgCritDialog* pDlg = new CProgCritDialog;
      if( pDlg -> DoModal() == IDOK )
            {pDoc -> UpdateAllViews(NULL,3L,NULL);
            if (pDoc -> m_bProgIndDlgDisplayed) pDoc ->
m_pProgIndDlg -> OnInitDialog();
      }
      delete pDlg;
}


void CProwin01View::OnUpdateViewLegend(CCmdUI* pCmdUI)
{
      pCmdUI -> Enable(!((CProwin01App*)AfxGetApp()) ->
m_bLegendDialogDisplayed);
}
```

```
void CProwin01View::OnUpdateViewDates(CCmdUI* pCmdUI)
{
      CProwin01Doc* pDoc = GetDocument();
      pCmdUI -> Enable(!(pDoc -> m_bDateDialogDisplayed));
}


void CProwin01View::OnLButtonDown(UINT nFlags, CPoint point)
{
      CProwin01Doc* pDoc = GetDocument();

      CClientDC dc(this);
      CRect rect,ClientRect,dummyRect;
      CSquareTracker recttracker;

      GetClientRect(&ClientRect);
      OnPrepareDC(&dc);

      int bar_width,bar_length,j;
      WORD sensitivity;

      if (!m_bBinocular)
            {if(m_tracker.IsRectNull())
                  {for(int i = 0; i < 76; i++)
                        {rect = m_BoxArray[i];
                        dc.LPtoDP(rect);
                        if (rect.PtInRect(point))
                              {::SetCursor((HCURSOR)AfxGetApp() ->
LoadStandardCursor(IDC_CROSS));
                              CPen
FramePen(PS_SOLID,0,RGB(0,0,255)) ;
                              recttracker.m_rect = rect;

      recttracker.TrackRubberBand(this,rect.TopLeft(),FALSE);
                              recttracker.m_rect.NormalizeRect();
                              if (recttracker.m_rect.Width() >
rect.Width() &&
dummyRect.IntersectRect(ClientRect,recttracker.m_rect))
                                    {dc.DPtoLP(rect);
                                    dc.DPtoLP(recttracker.m_rect);
                                    m_tracker =
recttracker.m_rect;
```

```
      recttracker.m_rect.InflateRect(-m_tracker.Width()*0.1,-
m_tracker.Height()*0.1);
                                    if(m_nCount > MAX_FIELDS)
bar_width = recttracker.m_rect.Width()/m_nCount;
                                          else bar_width =
recttracker.m_rect.Width()/MAX_FIELDS;
                                    if (m_bRightEye)

      {dc.FillRect(CRect(m_tracker.left,m_tracker.top,m_tracker.ri
ght,m_tracker.bottom),&m_DarkBlueBrush);
                                          for ( j = 0; j <
m_nCount; j++)
                                                {sensitivity =
(WORD)pDoc -> m_rightpointArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(j));
                                                if (sensitivity !=
MISSING_VALUE )
                                                      {bar_length
= recttracker.m_rect.Height() * (300 - (float) sensitivity) / 300;
                                                      if
(bar_length < m_tracker.top - recttracker.m_rect.top)

      bar_length = m_tracker.top - recttracker.m_rect.top;

      dc.FillRect(CRect(recttracker.m_rect.left + (j)*bar_width,

      recttracker.m_rect.top,recttracker.m_rect.left +
(j+1)*bar_width,

      recttracker.m_rect.top + bar_length),GetBrush(&i, &j,pDoc));
                                                }
                                          }
                                    }
                                    else

      {dc.FillRect(CRect(m_tracker.left,m_tracker.top,m_tracker.ri
ght,m_tracker.bottom),&m_DarkBlueBrush);
                                          for ( j = 0; j <
m_nCount; j++)
                                                {sensitivity =
(WORD)pDoc -> m_leftpointArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(j));
```

```
                                                    if (sensitivity !=
MISSING_VALUE )
                                                          {bar_length
= recttracker.m_rect.Height() * (300 - (float) sensitivity) / 300;
                                                          if
(bar_length < m_tracker.top - recttracker.m_rect.top)

      bar_length = m_tracker.top - recttracker.m_rect.top;

      dc.FillRect(CRect(recttracker.m_rect.left + (j)*bar_width,

      recttracker.m_rect.top,recttracker.m_rect.left +
(j+1)*bar_width,

      recttracker.m_rect.top + bar_length),GetBrush(&i, &j,pDoc));
                                                    }
                                              }
                                        }
                                        CPen* pOldPen =
dc.SelectObject(&FramePen);

      dc.MoveTo(m_tracker.BottomRight());

      dc.LineTo(m_tracker.left,m_tracker.bottom);

      dc.LineTo(m_tracker.TopLeft());

      dc.LineTo(m_tracker.BottomRight().x,m_tracker.TopLeft().y);

      dc.LineTo(m_tracker.BottomRight());
                                        dc.SelectObject(pOldPen);
                                  }
                            }
                      }
                }
                else
                      {rect = m_tracker;
                      dc.LPtoDP(rect);
                      if (rect.PtInRect(point))
                            {InvalidateRect(CRect(rect.TopLeft()-
CSize(10,10),rect.Size()+CSize(20,20)),
                                  (m_bShowGrayscale) ? TRUE : FALSE);
                            m_tracker.SetRectEmpty();
                      }
```

```
            }
      }

}

BOOL CProwin01View::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT
message)
{
      if (!m_bBinocular)
            {if(m_tracker.IsRectNull())

      {if(m_bCursorInRect)::SetCursor((HCURSOR)AfxGetApp() ->
LoadCursor(IDC_MAGNIFY));
                  else ::SetCursor((HCURSOR)AfxGetApp() ->
LoadStandardCursor(IDC_ARROW));
            }
            else

      {if(m_bCursorInRect)::SetCursor((HCURSOR)AfxGetApp() ->
LoadCursor(IDC_MINIFY));
                  else ::SetCursor((HCURSOR)AfxGetApp() ->
LoadStandardCursor(IDC_ARROW));
            }
      }
      else ::SetCursor((HCURSOR)AfxGetApp() ->
LoadStandardCursor(IDC_ARROW));

      return TRUE;
}

void CSquareTracker::AdjustRect(int nHandle, LPRECT lpRect)
{   CRect rect = *lpRect;
    int width = rect.Width();
      int height = rect.Height();
      if (width > height)rect.SetRect(rect.TopLeft().x,
rect.BottomRight().y -
width,rect.BottomRight().x,rect.BottomRight().y);
      if (height > width)rect.SetRect(rect.BottomRight().x -
height,
rect.TopLeft().y,rect.BottomRight().x,rect.BottomRight().y);
      *lpRect = rect;
}

void CProwin01View::OnMouseMove(UINT nFlags, CPoint point)
```

```
{
	CClientDC dc(this);
	CRect rect;
	CProwin01Doc* pDoc = GetDocument();
	CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);

	float sensitivity;
	BYTE pvalue;
	CString pstring;
	CString slopeString;
	long double slope;
	char message[150];

	m_nFlags = nFlags;
	m_mousePos = point;

	OnPrepareDC(&dc);

	if(!m_bBinocular)
		{if(m_tracker.IsRectNull())
			{for(int i = 0; i < 76; i++)
				{rect = m_BoxArray[i];
				dc.LPtoDP(rect);
				if (rect.PtInRect(point))
					{if (m_bRightEye)
						{if (pDoc ->
m_rightorderArray.GetSize())
							{sensitivity =
(WORD)pDoc -> m_rightpointArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(m_nCount - 1));
							if (sensitivity ==
MISSING_VALUE)sprintf(message,"30 - 2 location: not tested on this
date");
							else
								{pvalue =
(BYTE)pDoc -> m_rightpvalueArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(m_nCount - 1));
								if ((pvalue ==
4)||(pvalue == 6))pstring = "p >= 0.1";
								if ((pvalue ==
3)||(pvalue == 7))pstring = "p < 0.1";
								if ((pvalue ==
2)||(pvalue == 8))pstring = "p < 0.05";
```

```
                                                    if ((pvalue ==
1)||(pvalue == 9))pstring = "p < 0.01";
                                                    if ((pvalue ==
11))pstring = "p < 0.001";
                                                    slopeString =
(CString)pDoc -> m_rightslopeArray.GetAt(i + 76*pDoc ->
m_rightorderArray.GetAt(m_nCount - 1));
                                                    if
(slopeString.IsEmpty())

      {sprintf(message,"Sensitivity: %2.1f dB  Slope is undefined:
too few fields for analysis",sensitivity/10);
                                                    }
                                                    else
                                                          {slope =
_atold((const char*)slopeString);

      sprintf(message,"Sensitivity: %2.1f dB    Slope: %.2Lf dB /
yr    %s",sensitivity/10,slope,pstring);
                                                    }
                                              }
                                              pStatus-
>SetPaneText(0,message,TRUE);
                                              pStatus->UpdateWindow();
                                              }
                                        }
                                  else
                                        {if (pDoc ->
m_leftorderArray.GetSize())
                                              {sensitivity =
(WORD)pDoc -> m_leftpointArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(m_nCount - 1));
                                              if (sensitivity ==
MISSING_VALUE)sprintf(message,"30 - 2 location: not tested on this
date");
                                              else
                                                    {pvalue =
(BYTE)pDoc -> m_leftpvalueArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(m_nCount - 1));
                                                    if ((pvalue ==
4)||(pvalue == 6))pstring = "p >= 0.1";
                                                    if ((pvalue ==
3)||(pvalue == 7))pstring = "p < 0.1";
```

```
                                                    if ((pvalue ==
2)||(pvalue == 8))pstring = "p < 0.05";
                                                    if ((pvalue ==
1)||(pvalue == 9))pstring = "p < 0.01";
                                                    if ((pvalue ==
11))pstring = "p < 0.001";
                                                    slopeString =
(CString)pDoc -> m_leftslopeArray.GetAt(i + 76*pDoc ->
m_leftorderArray.GetAt(m_nCount - 1));
                                                    if
(slopeString.IsEmpty())

      {sprintf(message,"Sensitivity: %2.1f dB  Slope is undefined:
too few fields for analysis",sensitivity/10);
                                                    }
                                                    else
                                                          {slope =
_atold((const char*)slopeString);

      sprintf(message,"Sensitivity: %2.1f dB    Slope: %.2Lf dB /
yr    %s",sensitivity/10,slope,pstring);
                                                    }
                                              }
                                              pStatus-
>SetPaneText(0,message,TRUE);
                                              pStatus->UpdateWindow();
                                              }
                                        }
                                  m_bCursorInRect = TRUE;
                                  break;
                            }
                            else
                                  {//pStatus->SetPaneText(0,"For Help,
press F1",TRUE);
                                  //pStatus->UpdateWindow();
                                  m_bCursorInRect = FALSE;
                            }
                      }
                }
                else
                      {rect = m_tracker;
                      dc.LPtoDP(rect);
                      if (rect.PtInRect(point))m_bCursorInRect = TRUE;
                      else m_bCursorInRect = FALSE;
```

```
            }

      }
      else
            {sprintf(message,"Binocular simulation");
            pStatus->SetPaneText(0,message,TRUE);
            pStatus->UpdateWindow();
      }

      CScrollView::OnMouseMove(nFlags, point);
}

void CProwin01View::OnFilterGaussian()
{
      BYTE filter[3][3] ={{1,2,1},
                                    {2,4,2},
                                    {1,2,1}};
      int fx,fy;
      BYTE divisor;
      WORD filtered;

      int current_field, no_of_fields, pointno;

      int row,col;
      WORD matrix[12][12];

      CProwin01Doc* pDoc = GetDocument();

      CStatusBar* pStatus = (CStatusBar*) AfxGetApp()->m_pMainWnd-
>GetDescendantWindow(AFX_IDW_STATUS_BAR);

      if ( m_bRightEye )
            {no_of_fields = pDoc -> m_rightpointArray.GetSize() /
76;
            if (!no_of_fields)return;
                  BeginWaitCursor();
                  if (!(pDoc -> m_rightGaussianFiltered))
                        {pStatus->SetPaneText(0,"Applying Gaussian
filter...",TRUE);
                        pStatus->UpdateWindow();

                        for (int i = 0; i < pDoc ->
m_rightpointArray.GetSize(); i++)
```

```
                                pDoc ->
m_rightUndoArray.SetAtGrow(i, pDoc -> m_rightpointArray[i]);

                  for ( current_field = 0; current_field <
no_of_fields; current_field++ )
                        {for ( row = 0; row < 12; row++)
                              for ( col = 0; col < 12;
col++)matrix[row][col] = MISSING_VALUE;

                        for ( pointno = 0; pointno < 76;
pointno++)
                              { if ( pointno < 4 )
matrix[1][pointno + 4] = pDoc -> m_rightpointArray.GetAt( pointno
+ current_field * 76);
                              else if ( pointno > 3 && pointno <
10 ) matrix[2][pointno - 1] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                              else if ( pointno > 9 && pointno <
18 ) matrix[3][pointno - 8] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 17 && pointno < 28
) matrix[4][pointno - 17] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 27 && pointno < 38
) matrix[5][pointno - 27] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 37 && pointno < 48
) matrix[6][pointno - 37] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 47 && pointno < 58
) matrix[7][pointno - 47] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 57 && pointno < 66
) matrix[8][pointno - 56] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 65 && pointno < 72
) matrix[9][pointno - 63] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                            else if ( pointno > 71 )
matrix[10][pointno - 68] = pDoc -> m_rightpointArray.GetAt(
pointno + current_field * 76);
                        }
                        matrix[5][8] = matrix[6][8] =
MISSING_VALUE;///////blind spot excluded
```

```
                        for ( row = 1; row < 11; row++)
                              for ( col = 1; col < 11; col++)
                                    {if ( matrix[row][col] ==
MISSING_VALUE )continue;
                                    divisor = 0;
                                    filtered = 0;
                                    for ( fx = 0; fx < 3; fx++)
                                          for ( fy = 0; fy < 3;
fy++)
                                                {if
(matrix[row+fx-1][col+fy-1] != MISSING_VALUE)
                                                      {filtered +=
matrix[row+fx-1][col+fy-1] * filter[fx][fy];
                                                      divisor +=
filter[fx][fy];
                                                }
                                          }
                                    filtered /= divisor;
                                    if ( row == 1 && col > 3 &&
col < 8 )pDoc -> m_rightpointArray.SetAt( col - 4 + current_field
* 76, filtered);
                                    else if ( row == 2 && col > 2
&& col < 9 )pDoc -> m_rightpointArray.SetAt( col + 1 +
current_field * 76, filtered);
                                    else if ( row == 3 && col > 1
&& col < 10 )pDoc -> m_rightpointArray.SetAt( col + 8 +
current_field * 76, filtered);
                                    else if ( row == 4 && col > 0
&& col < 11 )pDoc -> m_rightpointArray.SetAt( col + 17 +
current_field * 76, filtered);
                                    else if ( row == 5 && col > 0
&& col < 11 )pDoc -> m_rightpointArray.SetAt( col + 27 +
current_field * 76, filtered);
                                    else if ( row == 6 && col > 0
&& col < 11 )pDoc -> m_rightpointArray.SetAt( col + 37 +
current_field * 76, filtered);
                                    else if ( row == 7 && col > 0
&& col < 11 )pDoc -> m_rightpointArray.SetAt( col + 47 +
current_field * 76, filtered);
                                    else if ( row == 8 && col > 1
&& col < 10 )pDoc -> m_rightpointArray.SetAt( col + 56 +
current_field * 76, filtered);
```

```
                                    else if ( row == 9 && col > 2
&& col < 9 )pDoc -> m_rightpointArray.SetAt( col + 63 +
current_field * 76, filtered);
                                    else if ( row == 10 && col > 3
&& col < 8 )pDoc -> m_rightpointArray.SetAt( col + 68 +
current_field * 76, filtered);
                                }
                }
                pDoc -> Regress();
                //pDoc -> SetModifiedFlag();
                pDoc -> m_rightGaussianFiltered = 1;
                pDoc -> UpdateAllViews(NULL,1L,NULL);

          }
          else
                {pStatus->SetPaneText(0,"Removing Gaussian
filter...",TRUE);
                pStatus->UpdateWindow();

                for (int i = 0; i < pDoc ->
m_rightpointArray.GetSize(); i++)
                      pDoc -> m_rightpointArray[i] =  pDoc ->
m_rightUndoArray[i];

                pDoc -> m_rightUndoArray.RemoveAll();
                pDoc -> Regress();
                //pDoc -> SetModifiedFlag();
                pDoc -> m_rightGaussianFiltered = 0;
                pDoc -> UpdateAllViews(NULL,1L,NULL);
          }
     }
     else
          {no_of_fields = pDoc -> m_leftpointArray.GetSize() /
76;
          if (!no_of_fields)return;
                BeginWaitCursor();
                if (!(pDoc -> m_leftGaussianFiltered))
                      {pStatus->SetPaneText(0,"Applying Gaussian
filter...",TRUE);
                      pStatus->UpdateWindow();

                      for (int i = 0; i < pDoc ->
m_leftpointArray.GetSize(); i++)
```

```
                                pDoc ->
m_leftUndoArray.SetAtGrow(i, pDoc -> m_leftpointArray[i]);

                  for ( current_field = 0; current_field <
no_of_fields; current_field++ )
                        {for ( row = 0; row < 12; row++)
                              for ( col = 0; col < 12;
col++)matrix[row][col] = MISSING_VALUE;

                        for ( pointno = 0; pointno < 76;
pointno++)
                              { if ( pointno < 4 )
matrix[1][pointno + 4] = pDoc -> m_leftpointArray.GetAt( pointno +
current_field * 76);
                              else if ( pointno > 3 && pointno <
10 ) matrix[2][pointno - 1] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                              else if ( pointno > 9 && pointno <
18 ) matrix[3][pointno - 8] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                                 else if ( pointno > 17 && pointno < 28
) matrix[4][pointno - 17] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                                 else if ( pointno > 27 && pointno < 38
) matrix[5][pointno - 27] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                                 else if ( pointno > 37 && pointno < 48
) matrix[6][pointno - 37] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                                 else if ( pointno > 47 && pointno < 58
) matrix[7][pointno - 47] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                                 else if ( pointno > 57 && pointno < 66
) matrix[8][pointno - 56] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                                 else if ( pointno > 65 && pointno < 72
) matrix[9][pointno - 63] = pDoc -> m_leftpointArray.GetAt(
pointno + current_field * 76);
                                 else if ( pointno > 71 )
matrix[10][pointno - 68] = pDoc -> m_leftpointArray.GetAt( pointno
+ current_field * 76);
                        }
                        matrix[5][8] = matrix[6][8] =
MISSING_VALUE;///////blind spot excluded
```

```
                        for ( row = 1; row < 11; row++)
                              for ( col = 1; col < 11; col++)
                                    {if ( matrix[row][col] ==
MISSING_VALUE )continue;
                                    divisor = 0;
                                    filtered = 0;
                                    for ( fx = 0; fx < 3; fx++)
                                          for ( fy = 0; fy < 3;
fy++)
                                                {if
(matrix[row+fx-1][col+fy-1] != MISSING_VALUE)
                                                      {filtered +=
matrix[row+fx-1][col+fy-1] * filter[fx][fy];
                                                      divisor +=
filter[fx][fy];
                                                }
                                          }
                                    filtered /= divisor;
                                    if ( row == 1 && col > 3 &&
col < 8 )pDoc -> m_leftpointArray.SetAt( col - 4 + current_field *
76, filtered);
                                    else if ( row == 2 && col > 2
&& col < 9 )pDoc -> m_leftpointArray.SetAt( col + 1 +
current_field * 76, filtered);
                                    else if ( row == 3 && col > 1
&& col < 10 )pDoc -> m_leftpointArray.SetAt( col + 8 +
current_field * 76, filtered);
                                    else if ( row == 4 && col > 0
&& col < 11 )pDoc -> m_leftpointArray.SetAt( col + 17 +
current_field * 76, filtered);
                                    else if ( row == 5 && col > 0
&& col < 11 )pDoc -> m_leftpointArray.SetAt( col + 27 +
current_field * 76, filtered);
                                    else if ( row == 6 && col > 0
&& col < 11 )pDoc -> m_leftpointArray.SetAt( col + 37 +
current_field * 76, filtered);
                                    else if ( row == 7 && col > 0
&& col < 11 )pDoc -> m_leftpointArray.SetAt( col + 47 +
current_field * 76, filtered);
                                    else if ( row == 8 && col > 1
&& col < 10 )pDoc -> m_leftpointArray.SetAt( col + 56 +
current_field * 76, filtered);
```

```
                                    else if ( row == 9 && col > 2
&& col < 9 )pDoc -> m_leftpointArray.SetAt( col + 63 +
current_field * 76, filtered);
                                    else if ( row == 10 && col > 3
&& col < 8 )pDoc -> m_leftpointArray.SetAt( col + 68 +
current_field * 76, filtered);
                                }
                }
                pDoc -> Regress();
                //pDoc -> SetModifiedFlag();
                pDoc -> m_leftGaussianFiltered = 1;
                pDoc -> UpdateAllViews(NULL,2L,NULL);

            }
            else
                {pStatus->SetPaneText(0,"Removing Gaussian
filter...",TRUE);
                pStatus->UpdateWindow();

                for (int i = 0; i < pDoc ->
m_leftpointArray.GetSize(); i++)
                        pDoc -> m_leftpointArray[i] =  pDoc ->
m_leftUndoArray[i];

                pDoc -> m_leftUndoArray.RemoveAll();
                pDoc -> Regress();
                //pDoc -> SetModifiedFlag();
                pDoc -> m_leftGaussianFiltered = 0;
                pDoc -> UpdateAllViews(NULL,2L,NULL);
            }
      }
      if (pDoc -> m_bProgIndDlgDisplayed) pDoc -> m_pProgIndDlg ->
OnInitDialog();
      if ((m_bRightEye && pDoc -> m_rightGaussianFiltered) ||
(!m_bRightEye && pDoc -> m_leftGaussianFiltered))
            pStatus->SetPaneText(0,"Gaussian filter applied. For
Help, press F1",TRUE);
      else pStatus->SetPaneText(0,"Gaussian filter removed. For
Help, press F1",TRUE);
      pStatus->UpdateWindow();
      EndWaitCursor();

}
```

```
void CProwin01View::OnUpdateFilterGaussian(CCmdUI* pCmdUI)
{
	CProwin01Doc* pDoc = GetDocument();
	if(m_bBinocular)pCmdUI -> Enable(FALSE);
	else
		{if (m_bRightEye)pCmdUI -> SetCheck(pDoc ->
m_rightGaussianFiltered);
		else pCmdUI -> SetCheck(pDoc ->
m_leftGaussianFiltered);
	}

}


void CProwin01View::OnWindowCloseall()
{
	//if (AfxMessageBox("All windows will be closed and unsaved
data will be lost.\n\n\tDo you wish to continue?",
	//MB_YESNO | MB_DEFBUTTON2 | MB_ICONQUESTION) == IDYES)
		((CProwin01App*)AfxGetApp()) ->
CloseAllDocuments(FALSE);

}

void CProwin01View::OnViewProgressionIndices()
{
	CProwin01Doc* pDoc = GetDocument();
	CProwin01View* pProwin01View = this;

	if (!(pDoc -> m_bProgIndDlgDisplayed))
		{pDoc -> m_pProgIndDlg -> m_pDoc = pDoc;
		pDoc -> m_pProgIndDlg -> m_pProwin01View =
pProwin01View;
		pDoc -> m_pProgIndDlg -> m_pLeftView = NULL;
		pDoc -> m_pProgIndDlg -> Create();
		pDoc -> m_bProgIndDlgDisplayed = TRUE;
	}

}

void CProwin01View::OnUpdateViewProgressionIndices(CCmdUI* pCmdUI)
{
	CProwin01Doc* pDoc = GetDocument();
	pCmdUI -> Enable(!(pDoc -> m_bProgIndDlgDisplayed));
```

```
}


void CProwin01View::OnViewGrayscale()
{
	m_bShowGrayscale = m_bShowGrayscale ^ 1;
	m_NameColor = m_bShowGrayscale ? RGB(0,0,0) :
RGB(255,255,255);
	Invalidate();

}

void CProwin01View::OnUpdateViewGrayscale(CCmdUI* pCmdUI)
{
	if(m_bBinocular)pCmdUI -> Enable(FALSE);
	else pCmdUI -> SetCheck(m_bShowGrayscale);

}

CBrush* CProwin01View::GetGrayscaleBrush(int* sens)
{
	int i = 0;

	if (*sens ==0) i = 9;
	else if (*sens > 0 && *sens < 51) i = 8;
	else if (*sens > 50 && *sens < 101) i = 7;
	else if (*sens > 100 && *sens < 151) i = 6;
	else if (*sens > 150 && *sens < 201) i = 5;
	else if (*sens > 200 && *sens < 251) i = 4;
	else if (*sens > 250 && *sens < 301) i = 3;
	else if (*sens > 300 && *sens < 351) i = 2;
	else if (*sens > 350 && *sens < 401) i = 1;
	else if (*sens > 400) i = 0;

	return &m_GrayscaleBrush[i];

}


void CProwin01View::AddToMatrix(int* point,int* sensitivity)
{
	if ( *point < 4 ) m_nMatrix[1][*point + 4] = *sensitivity;
```

```
            else if ( *point > 3 && *point < 10 )
m_nMatrix[2][*point - 1] = *sensitivity;
            else if ( *point > 9 && *point < 18 )
m_nMatrix[3][*point - 8] = *sensitivity;
            else if ( *point > 17 && *point < 28 )
m_nMatrix[4][*point - 17] = *sensitivity;
            else if ( *point > 27 && *point < 38 )
m_nMatrix[5][*point - 27] = *sensitivity;
            else if ( *point > 37 && *point < 48 )
m_nMatrix[6][*point - 37] = *sensitivity;
            else if ( *point > 47 && *point < 58 )
m_nMatrix[7][*point - 47] = *sensitivity;
            else if ( *point > 57 && *point < 66 )
m_nMatrix[8][*point - 56] = *sensitivity;
            else if ( *point > 65 && *point < 72 )
m_nMatrix[9][*point - 63] = *sensitivity;
          else if ( *point > 71 ) m_nMatrix[10][*point - 68] =
*sensitivity;

}

void CProwin01View::Interpolate(CDC* pDC)
{
      BOOL bBigCircle = FALSE;
      CRgn boundcircle;
      CRgn nasal_24_2;
      CPen* pOldPen;
      CPen GrayAxisPen(PS_SOLID,1,RGB(0,0,0));
      CPen WhiteAxisPen(PS_SOLID,9,RGB(255,255,255));
      CDC* pDisplayMemDC = new CDC;
      CBitmap* pBitmap = new CBitmap;
      CRect clientRect;
      CRect clipRect;
      int row,col,x,y,sens,nsens;
      int matrix[36][36];

      GetClientRect(clientRect);
      pDisplayMemDC -> CreateCompatibleDC(pDC);
      OnPrepareDC(pDisplayMemDC);
      pBitmap -> CreateCompatibleBitmap(pDC,clientRect.right -
clientRect.left,clientRect.bottom - clientRect.top);
      CBitmap* pOldBitmap = (CBitmap*)(pDisplayMemDC ->
SelectObject(pBitmap));
```

```
      pDisplayMemDC -> FillRect(CRect(-500,-
500,500,500),&m_WhiteBrush);

      for (row = 0; row < 36; row++)
            for (col = 0; col < 36; col++)
                  matrix[row][col] = MISSING_VALUE;

      for (row = 0; row < 12; row++)
            for (col = 0; col < 12; col++)
                  {sens = m_nMatrix[row][col];
                  matrix[row*3+1][col*3+1]  = sens;
                  if (row == 1 && sens != MISSING_VALUE)bBigCircle
= TRUE;
            }

      for (row = 4; row < 34; row+=3)
            for (col = 4; col < 34; col+=3)
                  {sens = matrix[row][col];
                  nsens = matrix[row-3][col];
                  if (sens != MISSING_VALUE)
                        {if (nsens == MISSING_VALUE)
                              {matrix[row-1][col] = sens;
                              matrix[row-2][col] = sens;
                        }
                        else matrix[row-1][col] = sens*2/3 +
nsens*1/3;
                  }

                  nsens = matrix[row+3][col];
                  if (sens != MISSING_VALUE )
                        {if (nsens == MISSING_VALUE)
                              {matrix[row+1][col] = sens;
                              matrix[row+2][col] = sens;
                        }
                        else matrix[row+1][col] = sens*2/3 +
nsens*1/3;
                  }

                  nsens = matrix[row][col-3];
                  if (sens != MISSING_VALUE )
                        {if (nsens == MISSING_VALUE)
                              {matrix[row][col-1] = sens;
                              matrix[row][col-2] = sens;
                        }
```

```
                        else matrix[row][col-1] = sens*2/3 +
nsens*1/3;
                  }

                  nsens = matrix[row][col+3];
                  if (sens != MISSING_VALUE )
                        {if (nsens == MISSING_VALUE)
                              {matrix[row][col+1] = sens;
                              matrix[row][col+2] = sens;
                        }
                        else matrix[row][col+1] = sens*2/3 +
nsens*1/3;
                  }

            }

      for (row = 3; row < 34; row++)
            for (col = 4; col < 34; col+=3)
                  {if (row%3 == 1)break;
                  sens = matrix[row][col];
                  nsens = matrix[row][col-3];
                  if (sens != MISSING_VALUE )
                        {if (nsens == MISSING_VALUE)
                              {matrix[row][col-1] = sens;
                              matrix[row][col-2] = sens;
                        }
                        else matrix[row][col-1] = sens*2/3 +
nsens*1/3;
                  }

                  nsens = matrix[row][col+3];
                  if (sens != MISSING_VALUE )
                        {if (nsens == MISSING_VALUE)
                              {matrix[row][col+1] = sens;
                              matrix[row][col+2] = sens;
                        }
                        else matrix[row][col+1] = sens*2/3 +
nsens*1/3;
                  }

            }
```

```
      if (bBigCircle)boundcircle.CreateEllipticRgn(-450,-
450,450,450);
            else
                  {boundcircle.CreateEllipticRgn(-390,-
390,390,400);
                  nasal_24_2.CreateEllipticRgn(-450,-290,450,295);

      boundcircle.CombineRgn(&boundcircle,&nasal_24_2,RGN_OR);
            }


      for (row = 3; row < 33; row++)
            for (col = 3; col < 33; col++)
                  {sens = matrix[row][col];
                  if (sens == MISSING_VALUE)continue;
                  x = -540 + col*(GRAY_BOX_DIMENSION);
                  y = -540 + row*(GRAY_BOX_DIMENSION);
                  if
(boundcircle.PtInRegion(x+GRAY_BOX_DIMENSION/2,y+GRAY_BOX_DIMENSIO
N/2))
                  pDisplayMemDC -> FillRect( CRect(CPoint(x,y),

      CSize(GRAY_BOX_DIMENSION,GRAY_BOX_DIMENSION)),

      GetGrayscaleBrush(&sens));
            }


      pOldPen = pDisplayMemDC -> SelectObject(&WhiteAxisPen);
      pDisplayMemDC -> MoveTo(0,450);
      pDisplayMemDC -> LineTo(0,-450);
      pDisplayMemDC -> MoveTo(-450,0);
      pDisplayMemDC -> LineTo(450,0);
      pDisplayMemDC -> SelectObject(&GrayAxisPen);
      pDisplayMemDC -> MoveTo(0,450);
      pDisplayMemDC -> LineTo(0,-450);
      pDisplayMemDC -> MoveTo(-450,0);
      pDisplayMemDC -> LineTo(450,0);
      pDisplayMemDC -> SelectObject(pOldPen);

      pDC -> ExcludeClipRect(-500,-500,500,-445);          //name
      pDC -> ExcludeClipRect(-500,-450,-350,-410);     //eye
      pDC -> BitBlt(-500,-500,1000,1000,pDisplayMemDC,-500,-
500,SRCCOPY);
```

```
    pDisplayMemDC -> SelectObject(pOldBitmap);
      boundcircle.DeleteObject();
      nasal_24_2.DeleteObject();
      pDC -> SelectClipRgn(NULL);
      delete pDisplayMemDC;
    delete pBitmap;
}

void CProwin01View::OnEyeBinocular()
{
      m_bBinocular = m_bBinocular ^ 1;
      m_NameColor = (m_bBinocular | m_bShowGrayscale) ? RGB(0,0,0)
: RGB(255,255,255);
      Invalidate();

}

void CProwin01View::OnUpdateEyeBinocular(CCmdUI* pCmdUI)
{
      pCmdUI -> SetCheck(m_bBinocular);

}

void CProwin01View::OnViewEstermanDefects()
{
      m_bEsterman = m_bEsterman ^ 1;
      Invalidate(FALSE);

}

void CProwin01View::OnUpdateViewEstermanDefects(CCmdUI* pCmdUI)
{
      pCmdUI -> Enable(m_bBinocular);
      pCmdUI -> SetCheck(m_bEsterman);
}

void CProwin01View::OnEditCopy()
{
      CDC* pDC = new CDC;
      CClientDC dc(this);
      CBitmap* pOldBitmap;
      CBitmap* pBitmap = new CBitmap;
      CRect clientRect;
      GetClientRect(clientRect);
```

```
	pDC -> CreateCompatibleDC(&dc);
	pBitmap -> CreateCompatibleBitmap(&dc,clientRect.right -
clientRect.left,clientRect.bottom - clientRect.top);
	pOldBitmap = pDC -> SelectObject(pBitmap);
	pDC -> BitBlt(0,0,1100,1100,&dc,0,0,SRCCOPY);
	pDC -> SelectObject(pOldBitmap);

	OpenClipboard();
	::EmptyClipboard();
	::SetClipboardData(CF_BITMAP,pBitmap -> Detach());

	CloseClipboard();
	pBitmap -> DeleteObject();
	delete pBitmap;
	delete pDC;
}

void CProwin01View::OnEditCopyScreen()
{
	CDC* pDC = new CDC;
	CWindowDC dc(AfxGetApp() -> m_pMainWnd);
	CBitmap* pOldBitmap;
	CBitmap* pBitmap = new CBitmap;
	CRect MainWndRect;
	AfxGetApp() -> m_pMainWnd ->GetWindowRect(MainWndRect);

	pDC -> CreateCompatibleDC(&dc);
	pBitmap -> CreateCompatibleBitmap(&dc,MainWndRect.right -
MainWndRect.left,MainWndRect.bottom - MainWndRect.top);
	pOldBitmap = pDC -> SelectObject(pBitmap);
	pDC -> BitBlt(0,0,MainWndRect.right -
MainWndRect.left,MainWndRect.bottom -
MainWndRect.top,&dc,0,0,SRCCOPY);
	pDC -> SelectObject(pOldBitmap);

	OpenClipboard();
	::EmptyClipboard();
	::SetClipboardData(CF_BITMAP,pBitmap -> Detach());

	CloseClipboard();
	pBitmap -> DeleteObject();
	delete pBitmap;
	delete pDC;
```

```
}

void CProwin01View::OnPrint(CDC* pDC, CPrintInfo* pInfo)
{
      CRect printRect = pInfo -> m_rectDraw;
      pDC -> LPtoDP(printRect);
      pDC -> SetMapMode(MM_ISOTROPIC);
      pDC -> SetWindowExt(1000,1000);
      pDC -> SetViewportExt(printRect.right,printRect.bottom);
      pDC -> SetViewportOrg(printRect.right/2,printRect.bottom/2);
      pDC -> DPtoLP(printRect);
      if ( !(m_bShowGrayscale | m_bBinocular) )
            {pDC -> FillRect(printRect,&m_DarkBlueBrush);
      }
      if (pInfo -> m_nCurPage == 1)
            {OnDraw(pDC);
      }
      else
            {m_bRightEye = m_bRightEye ^ 1;
            OnDraw(pDC);
            m_bRightEye = m_bRightEye ^ 1;
      }
}
```

```
// prowivw.h : interface of the CProwin01View class
//
/////////////////////////////////////////////////////////////////////////
///////////

class CProwin01View : public CScrollView
{   class CProwin01Doc;
      friend class CProgIndDlg;
private:
      BOOL m_bRightEye;
      BOOL m_bBinocular;
      BOOL m_bEsterman;
      BOOL m_bShowProgressingPoints;
      BOOL m_bShowGrayscale;
      BOOL m_bCursorInRect;
      UINT m_nFlags;
      CPoint m_mousePos;
      int m_nCount;
protected: // create from serialization only
      CProwin01View();
      DECLARE_DYNCREATE(CProwin01View)
// Data members
public:
      CBrush m_LightBlueBrush;
      CBrush m_LightRedBrush;
      CBrush m_DarkRedBrush;
      CBrush m_MagentaBrush;
      CBrush m_YellowBrush;
      CBrush m_GrayBrush;
      CBrush m_BlueGreenBrush;
      CBrush m_OliveBrush;
      CBrush m_DarkGreenBrush;
      CBrush m_LightGreenBrush;
      CBrush m_RedHatchBrush;
      CBrush m_DarkBlueBrush;
      CBrush m_WhiteBrush;
      COLORREF m_NameColor;
      CRect m_BoxArray[76];
      CRect m_tracker;

      char m_strGrayBrushName[11];
      CBrush m_GrayscaleBrush[10];
      CBitmap m_GrayscaleBitmap[10];
```

```
      int m_nMatrix[12][12];
// Attributes
public:
      CProwin01Doc* GetDocument();

// Operations
public:
      CBrush* GetBrush(int* point, int* field, CProwin01Doc*
pDoc);
      CBrush* GetGrayscaleBrush(int* sens);
      BOOL IsProgressing(int* point,int* last_field,CProwin01Doc*
pDoc,BOOL* pbRightEye);
      void AddToMatrix(int* point,int* sensitivity);
      void Interpolate(CDC* pDC);
// Implementation
public:
      virtual ~CProwin01View();
      virtual void OnDraw(CDC* pDC);  // overridden to draw this
view
#ifdef _DEBUG
      virtual void AssertValid() const;
      virtual void Dump(CDumpContext& dc) const;
#endif

protected:
      virtual void OnInitialUpdate(); // called first time after
construct
      virtual void OnActivateView(BOOL bActivate, CView*
pActivateView, CView* pDeactiveView);
      virtual void OnUpdate(CView* pView, LPARAM lHint, CObject*
pHint);
    void OnPrepareDC(CDC* pDC, CPrintInfo* pInfo = NULL);
      // Printing support
      virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
      virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
      virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
      virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);

// Generated message map functions
protected:
      //{{AFX_MSG(CProwin01View)
      afx_msg void OnEyeRight();
      afx_msg void OnUpdateEyeRight(CCmdUI* pCmdUI);
      afx_msg void OnEyeLeft();
```

```
      afx_msg void OnUpdateEyeLeft(CCmdUI* pCmdUI);
      afx_msg void OnViewDates();
      afx_msg void OnViewLegend();
      afx_msg void OnViewProgressingPoints();
      afx_msg void OnUpdateViewProgressingPoints(CCmdUI* pCmdUI);
      afx_msg void OnViewProgCrit();
      afx_msg void OnUpdateViewLegend(CCmdUI* pCmdUI);
      afx_msg void OnUpdateViewDates(CCmdUI* pCmdUI);
      afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
      afx_msg BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT
message);
      afx_msg void OnMouseMove(UINT nFlags, CPoint point);
      afx_msg void OnFilterGaussian();
      afx_msg void OnUpdateFilterGaussian(CCmdUI* pCmdUI);
      afx_msg void OnWindowCloseall();
      afx_msg void OnViewProgressionIndices();
      afx_msg void OnUpdateViewProgressionIndices(CCmdUI* pCmdUI);
      afx_msg void OnViewGrayscale();
      afx_msg void OnUpdateViewGrayscale(CCmdUI* pCmdUI);
      afx_msg void OnEyeBinocular();
      afx_msg void OnUpdateEyeBinocular(CCmdUI* pCmdUI);
      afx_msg void OnViewEstermanDefects();
      afx_msg void OnUpdateViewEstermanDefects(CCmdUI* pCmdUI);
      afx_msg void OnEditCopy();
      afx_msg void OnEditCopyScreen();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in prowivw.cpp
inline CProwin01Doc* CProwin01View::GetDocument()
   { return (CProwin01Doc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////
///////////
// CSquareTracker class

class CSquareTracker : public CRectTracker
{
public:
      void AdjustRect(int nHandle, LPRECT lpRect);
};
```

```
// refdlg.cpp : implementation file
//

#include "stdafx.h"
#include "prowin01.h"
#include "refdlg.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
///////////
// CRefDialog dialog


CRefDialog::CRefDialog(CWnd* pParent /*=NULL*/)
      : CDialog(CRefDialog::IDD, pParent)
{
      //{{AFX_DATA_INIT(CRefDialog)
      m_strRefs = "";
      //}}AFX_DATA_INIT
}

void CRefDialog::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CRefDialog)
      DDX_Text(pDX, IDC_REFS_EDIT, m_strRefs);
      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CRefDialog, CDialog)
      //{{AFX_MSG_MAP(CRefDialog)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////
///////////
// CRefDialog message handlers

BOOL CRefDialog::OnInitDialog()
```

```
{
      int x = ::GetSystemMetrics(SM_CXSCREEN);
      int y = ::GetSystemMetrics(SM_CYSCREEN);

      CRect dlgRect;
      GetWindowRect(&dlgRect);
      CDialog::OnInitDialog();

      SetWindowPos(&CWnd::wndTopMost,x/2-dlgRect.Width()/2,y/2-
dlgRect.Height()/2,0,0,SWP_NOSIZE );

      return TRUE;  // return TRUE  unless you set the focus to a
control
}
```

```
// refdlg.h : header file
//

/////////////////////////////////////////////////////////////////////////
///////////
// CRefDialog dialog

class CRefDialog : public CDialog
{
// Construction
public:
      CRefDialog(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
      //{{AFX_DATA(CRefDialog)
      enum { IDD = IDD_REFERENCE_DIALOG };
      CString     m_strRefs;
      //}}AFX_DATA

// Implementation
protected:
      virtual void DoDataExchange(CDataExchange* pDX);      //
DDX/DDV support

      // Generated message map functions
      //{{AFX_MSG(CRefDialog)
      virtual BOOL OnInitDialog();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
```

```
// splitfrm.cpp : implementation file
//

#include "stdafx.h"
#include "prowin01.h"
#include "splitfrm.h"
#include "prowivw.h"
#include "leftview.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
///////////
// CSplitFrame

IMPLEMENT_DYNCREATE(CSplitFrame, CMDIChildWnd)

CSplitFrame::CSplitFrame()
{
}

CSplitFrame::~CSplitFrame()
{
}

BOOL CSplitFrame::OnCreateClient(LPCREATESTRUCT /*lpcs*/,
CCreateContext* pContext)
{   int x = ::GetSystemMetrics(SM_CXSCREEN);
      int y = ::GetSystemMetrics(SM_CYSCREEN);

      BOOL rtn = m_wndSplitter.CreateStatic(this,1,2);
      rtn |=
m_wndSplitter.CreateView(0,0,RUNTIME_CLASS(CLeftView),CSize(0.386*
x,y/3),pContext);
      rtn |=
m_wndSplitter.CreateView(0,1,RUNTIME_CLASS(CProwin01View),CSize(0.
386*x,y/3),pContext);
      return rtn;
      /*return m_wndSplitter.Create(this,
            2, 2,       // TODO: adjust the number of rows,
columns
```

```
            CSize(10, 10),    // TODO: adjust the minimum pane
size
            pContext);*/
}

BEGIN_MESSAGE_MAP(CSplitFrame, CMDIChildWnd)
      //{{AFX_MSG_MAP(CSplitFrame)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////
///////////
// CSplitFrame message handlers
```

```
// splitfrm.h : header file
//

/////////////////////////////////////////////////////////////////////////
////////////
// CSplitFrame frame with splitter

#ifndef __AFXEXT_H__
#include <afxext.h>
#endif

class CSplitFrame : public CMDIChildWnd
{    class CProwin01View;
       class CLeftView;
      DECLARE_DYNCREATE(CSplitFrame)
protected:
      CSplitFrame();                // protected constructor used
by dynamic creation

// Attributes
protected:
      CSplitterWnd      m_wndSplitter;
public:

// Operations
public:

// Implementation
public:
      virtual ~CSplitFrame();
      virtual BOOL OnCreateClient(LPCREATESTRUCT lpcs,
CCreateContext* pContext);

      // Generated message map functions
      //{{AFX_MSG(CSplitFrame)
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////
////////////
```

APPENDIX B. SUPPORTING PUBLICATIONS

Peer reviewed publications

1. Viswanathan AC, Hitchings RA, Fitzke FW. How often do patients need visual field tests? *Graefes Arch Clin Exp Ophthalmol* 1997;235:563-568.

2. Viswanathan AC, Fitzke FW, Hitchings RA. Early Detection of Visual Field Progression in Glaucoma: A Comparison of PROGRESSOR and Statpac 2. *Br J Ophthalmol* 1997;81(12):1037-1042.

3. Viswanathan AC, McNaught AI, Poinoosawmy D, Fontana L, Crabb DP, Fitzke FW, Hitchings RA. Severity and stability of glaucoma: patient perception compared with objective measurement. *Arch Ophthalmol* 1999;117(4):450-4.

4. Viswanathan AC, Hitchings RA, Fitzke FW. Spatial Filtering of Glaucomatous Visual Fields using PROGRESSOR for Windows. In: Wall M, Heijl A, editors. *Perimetry Update*. Amsterdam / New York: Kugler Publications, 1996/1997:311-319.

Papers under review

Viswanathan AC, Crabb DP, McNaught AI, Westcott MC, Kamal D, Garway-Heath DF, Fitzke FW, Hitchings RA. Inter-Observer Agreement on Visual Field Progression in Glaucoma: a Comparison of Methods. *Invest Ophthalmol Vis Sci* 1999, under review.

Book Chapters

Visual field analysis. A. C. Viswanathan and F. W. Fitzke. In: The Oxford Textbook of Ophthalmology. Oxford University Press, Oxford, UK. Eds. D. L. Easty and J. M. Sparrow. 1st Edition 1999, Vol.1 p. 185-192.

Change in visual field. A. C. Viswanathan. In: Fundamentals of Clinical Ophthalmology: Glaucoma. BMJ Books: Editor R. A. Hitchings. Series editor S. Lightman (in press).

SPATIAL FILTERING OF GLAUCOMATOUS VISUAL FIELDS USING PROGRESSOR FOR WINDOWS

ANANTH C. VISWANATHAN[1], FREDERICK W. FITZKE[1] and ROGER A. HITCHINGS[2]

[1]*Institute of Ophthalmology, London;* [2]*Moorfields Eye Hospital, London; UK*

Abstract

Background: PROGRESSOR for Windows is a computerized system for the analysis of glaucomatous field progression incorporating a graphical user interface. The software package includes spatial filtering, which has been shown to reduce long-term fluctuation.[1] However, some spatial processing, such as gaussian filtering, may 'blur' early focal defects.
Purpose: To determine the role of gaussian filtering in the early detection of glaucomatous loss.
Methods: Nineteen field series from untreated normal-tension glaucoma patients, which an experienced observer judged as deteriorating, were studied. The time taken from the start of each series until progression criteria (slope worse than -1 dB/year for inner points, -2 dB/year for edge points, $p < 0.05$) were satisfied by at least one retinal location was calculated with and without gaussian filtering.
Results: The unfiltered fields detected progression earlier than the filtered fields in three of the 19 field series (mean 1.18 years, SD 0.30 years). The filtered fields detected progression earlier in five series (mean 1.04 years, SD 1.48 years). Both filtered and unfiltered fields detected progression at the same test in 11 field series. There was no predominance of focal defects in the series where progression was detected earlier by the unfiltered fields.
Conclusions: PROGRESSOR for Windows is a convenient software tool for analyzing glaucomatous field decay. Gaussian filtering reduces long-term fluctuation without delaying the detection of early loss in normal-tension glaucoma.

Introduction

Visual field series obtained by automated perimetry provide both spatial and temporal information about disease progression in glaucoma patients. It is crucial that the rate of progression is accurately quantified since this provides a measure of the need for, and the efficacy of, treatment. This task is hampered by the inherent variability between field tests (long-term fluctuation[1]) which is known to be greater in glaucoma.[2] A promising avenue of research is the application of digital image processing techniques, such as that used to process images obtained by magnetic resonance imaging (MRI) to reduce the 'noise' in the results of computerized perimetry. These spatial filtering techniques take account of the interdependence of neighboring retinal

Address for correspondence: F.W. Fitzke, PhD, Institute of Ophthalmology, 11-43 Bath Street, London EC1V 9EL, UK

Perimetry Update 1996/1997, pp. 311–319
Proceedings of the XIIth International Perimetric Society Meeting
Würzburg, Germany, June 4–8, 1996
edited by M. Wall and A. Heijl

test locations and have been shown to improve the repeatability of automated perimetry by a factor of two[3] and to improve the predictability of glaucomatous decay.[4] However, it is possible that spatial filtering will smooth areas of the field representative of early focal decay, rather than simply noise, and valuable information will be lost.

The PROGRESSOR software[5] performs pointwise linear regression of sensitivity on time and presents the results as a cumulative graphical display (Fig. 1a). Each test location is represented by a bar graph in which each bar represents a successive field test. The length of the bar relates to the depth of defect, and the color of the bar represents the *p* value of the regression slope (Fig. 1b). The pointwise linear model has been demonstrated to provide a valid framework for detecting and forecasting glaucomatous loss[6] and PROGRESSOR has been found to compare favorably with other methods of glaucoma change analysis such as STATPAC-2.[7] PROGRESSOR has recently been updated: the program now runs in a Windows environment and incorporates many new methods of analysis, one of which is the ability to apply spatial filtering to a series of fields (Fig. 2).

This study was designed to investigate, using PROGRESSOR for Windows, whether spatial filtering leads to a clinically significant delay in the detection of glaucomatous field progression.

Methods

Subjects

Patients with untreated normal-tension glaucoma which had been confirmed by phasing were chosen for study as it was felt to be important to analyze the natural history of glaucomatous visual damage in the absence of the potentially confounding effects of therapy. All subjects had visual acuity of 20/40 or better. None had significant ocular pathology apart from normal-tension glaucoma. All subjects were experienced in Humphrey 30-2 tests and able to produce reliable computerized visual fields (less than 30% fixation losses and false negatives and less than 15% false positives). Each had had at least two tests over four months prior to the observation period: this is sufficient to obviate any learning effects[8,9] which may delay the diagnosis of progression.

Patients were chosen whose visual fields appeared to be progressively deteriorating in a typically glaucomatous manner, since it is in these patients that the 'blurring' associated with spatial filtering is most likely to cause delay in the detection of progression. This was done by inspection of STATPAC overview printouts by an experienced observer.

On the basis of the foregoing criteria, 19 eyes from 13 subjects were selected. An indication of the degree of glaucomatous damage in the selected group is given by the following summary measures of the mean deviation (MD) of the initial field in each test series: the mean of the MDs was -6.81 dB (SD 6.01 dB), the median was -5.43 dB and the range was -22.40 dB to +1.07 dB.

Fig. 1a. Cumulative graphical display from PROGRESSOR for Windows for a progressing normal-tension glaucoma patient.

Testing strategy

All tests were performed on a standard Humphrey automated perimeter. The 30-2 Full Threshold Program with standard 4-2 dB double reversal strategy was used throughout. Tests were performed at intervals of four months.

Progression criteria

PROGRESSOR performs linear regression of sensitivity on time for each test location. For each field in the series, each location is ascribed a change slope (in decibels per year) and a p value for a two tailed t test of the slope against zero (*i.e.*, the null hypothesis of no deterioration). A field series was regarded as progressing if any non-edge location showed a deterioration of 1 dB per year or worse with $p < 0.05$. A

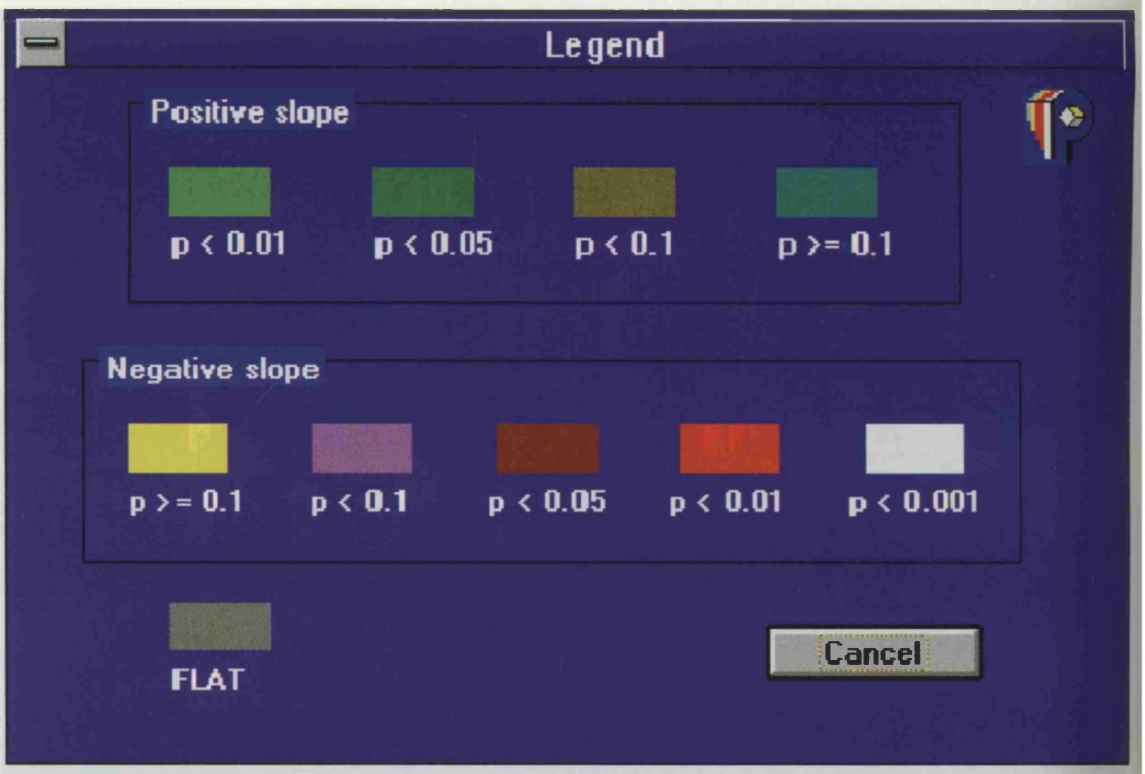


Fig. 1b. PROGRESSOR for Windows legend showing the colors which correspond to each p value of the regression slope.

more stringent slope criterion of 2 dB per year was applied to the edge points of the 30-2 test grid. These slope criteria have been demonstrated to compare closely with the Humphrey STATPAC-2 Glaucoma Change Probability analysis, and the *p* value criterion is stricter than that required to emulate STATPAC-2.[5]

Detection time

The detection time for a field series was defined as the time interval between the initial field and the field when the progression criteria (*vide supra*) were first satisfied.

Spatial filtering

Gaussian low-pass filtering is widely used to remove noise from pixel-based images.[10,11] In order to apply the filter to a 30-2 field, each test location is regarded as the center of a 3x3 neighborhood to which the filter (convolution mask) is applied (Fig. 3). Thus, each point is influenced by the weighted sensitivity of its contiguous neighbors. At the edges of the field the mask is only partial since edge points do not have a full complement of neighbors.

Statistical analysis

For each field series, detection time (*vide supra*) was calculated for both filtered and unfiltered data. Detection times for the unfiltered data were compared with their

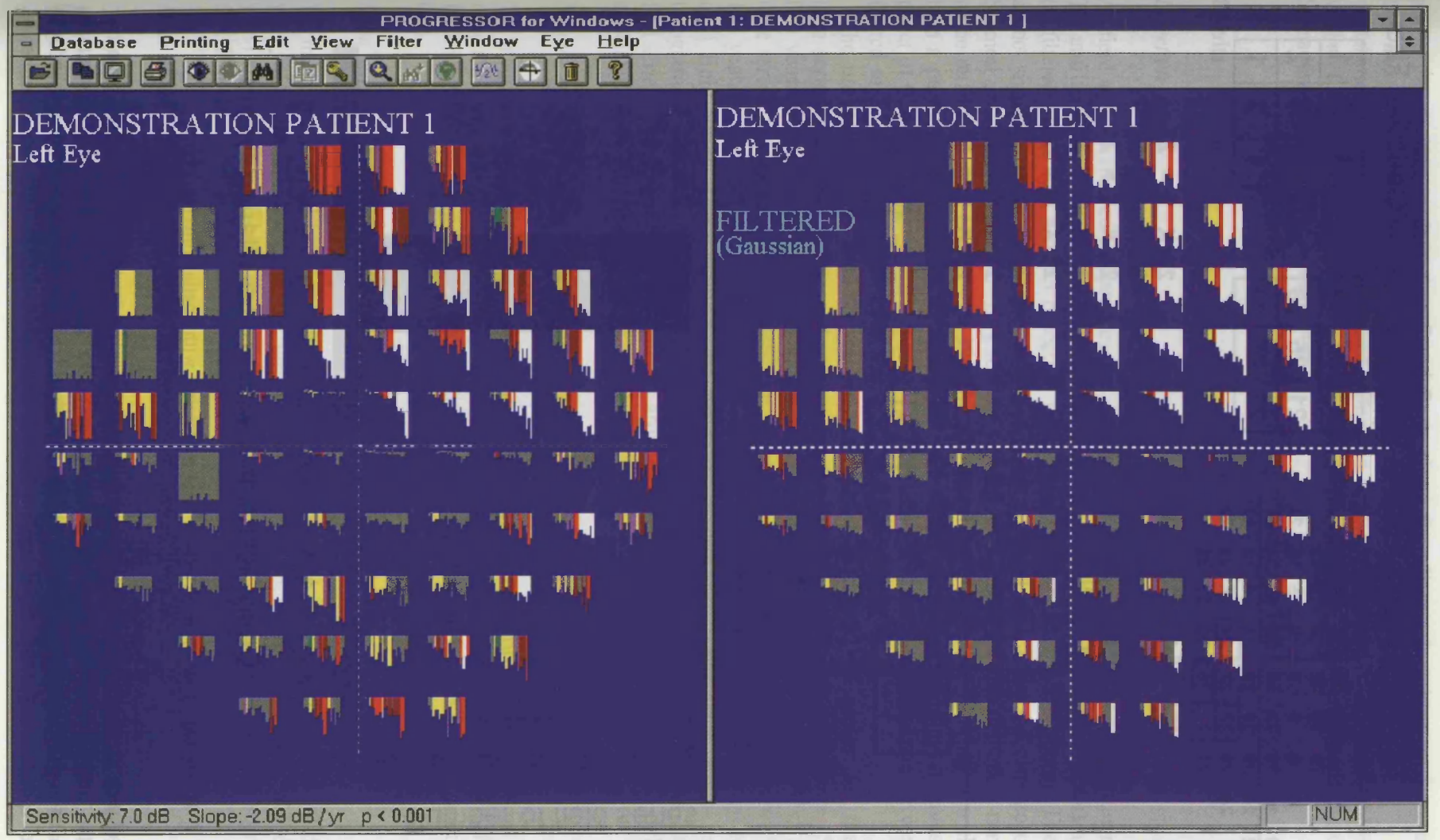


Fig. 2. PROGRESSOR for Windows display showing the results of gaussian filtering.

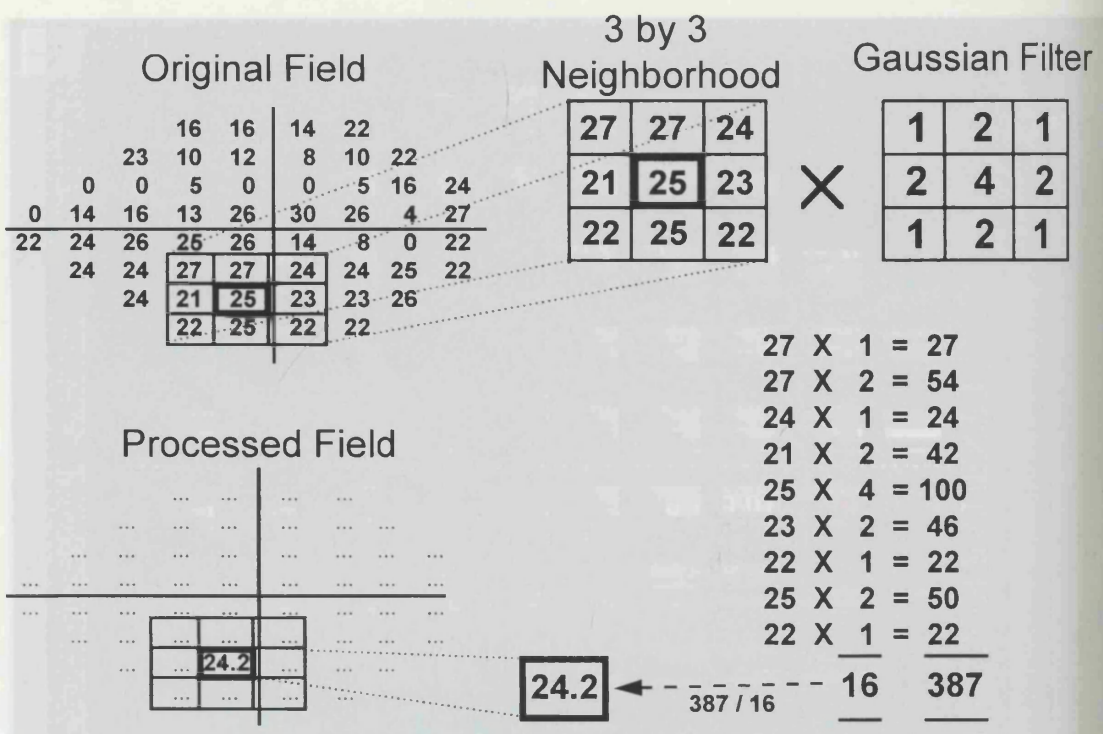


Fig. 3. Gaussian filtering process.

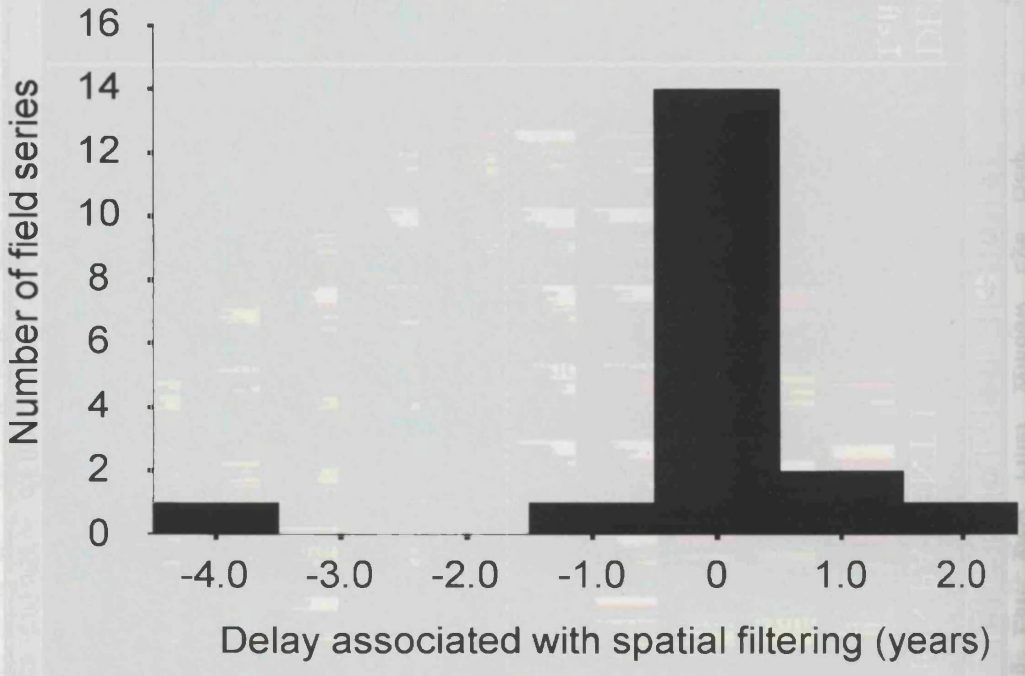


Fig. 4. Histogram of delay in detection time associated with spatial filtering.

correlates for the filtered data using a non-parametric test for paired data from two related samples (Wilcoxon signed rank Z test). In addition, the following summary measures of progression, automatically calculated by PROGRESSOR for Windows, were compared for all field series before and after spatial filtering: number of

progressing points at detection time, mean slope for all test locations at detection time and mean slope for progressing points at detection time.

Statistical analysis was performed using the software package SPSS for Windows version 6.0, except for the power calculation which was performed with Jandel SigmaStat for Windows version 1.0.

Results

Detection times

All field series satisfied the progression criteria both before and after spatial filtering. The unfiltered fields had a mean detection time of 1.077 years (SD 0.985 years) and the filtered fields had a mean detection time of 0.989 years (SD 0.639 years). These are not significantly different ($p = 0.779$, Wilcoxon signed rank Z test). Kolmogorov-Smirnov goodness of fit normality testing yielded $p = 0.260$ for the detection times of the unfiltered data and $p = 0.394$ for the detection times after filtering. On this basis a paired t test is justified and gives $p = 0.709$: a power calculation gives a power of 0.797 to detect a difference of eight months in mean detection time between filtered and unfiltered fields ($\alpha = 0.05$, SD = 0.985 years).

The unfiltered fields detected progression earlier in three field series: for these three series, the mean delay in detection for the filtered fields was 1.18 years (SD 0.30 years). The filtered fields detected progression earlier in five field series: for these series, the mean delay in detection for the unfiltered fields was 1.04 years (SD 1.48 years). Both filtered and unfiltered fields detected progression at the same test in 11 field series. These findings are displayed graphically in Figure 4, which is a histogram of the differences between filtered and unfiltered detection times for each field series. The distribution is centered around zero, which suggests no overall delay associated with filtering.

Number of progressing points at detection time

The mean number of progressing points which satisfied the progression criteria at detection time was 2.95 (SD 1.81) for the unfiltered fields and 3.00 (SD 2.49) after filtering. This difference was not significant ($p = 0.784$, Wilcoxon signed rank Z test).

Mean slope (whole field) at detection time

The mean slope of all test locations at detection time had a mean value of -3.99 dB/years (SD 9.94 dB/years) for the unfiltered fields and -3.76 dB/years (SD 9.84 dB/years) after filtering. This difference was not significant ($p = 0.334$, Wilcoxon signed rank Z test).

Mean slope (progressing points) at detection time

The mean slope of the locations which satisfied the progression criteria at detection time had a mean value of -22.47 dB/years (SD 56.44 dB/years) for the unfiltered fields and -9.15 dB/years (SD 14.14 dB/years) after filtering. This difference was significant at the 5% level ($p = 0.030$, Wilcoxon signed rank Z test).

Discussion

Fluctuation owing to variable patient response and other factors is the main obstacle to the accurate quantification of the results of automated perimetry. Any technique which may reduce this fluctuation is therefore worthy of study. Spatial filtering has been shown to be effective in reducing noise in other two-dimensional digital image processing applications[10,11] and seems to be of benefit in the analysis of glaucomatous visual field progression.[3,4] However, by its very nature, gaussian filtering obscures elements of a digital image which have high spatial frequency. If the digital image concerned is a computerized visual field, these elements are small isolated scotomas and the edges of existing scotomas: these are precisely the areas where progression is most likely to occur first. Thus, it is important to ascertain whether the process of spatial filtering leads to a clinically significant delay in detecting progression, since this would be a high price to pay for improved repeatability and predictability.

This study suggests that gaussian filtering is unlikely to entail a significant delay in the detection of progression.

This study was performed on a highly selected group of patients: the selection criteria were necessarily strict in order to isolate cases in which spatial filtering was most likely to cause a delay in detection. Since detection time was not affected in this group it is likely that other, more stable visual field series will be similarly unaffected. However, further work is required in order to determine whether these results may be generalized to patients with other forms of glaucoma and to those who have received or are receiving medical treatment.

The gaussian filter used in the study was chosen because it is a standard image-processing convolution mask, and has been used in previous work on computerized visual fields.[3,4] However, it is unlikely to be the ideal mask for the special environment of automated perimetry. Firstly, the mask is usually intended for use with pixels which have a range of values (bandwidth) of between 0 and 255: the range of sensitivity values produced by automated perimetry is almost an order of magnitude smaller than this. Secondly, the mask is limited to contiguous neighbors and is symmetrical. This does not conform to our knowledge of the patterns of glaucomatous field loss seen in clinical practice. For example, it seems likely that test locations will be more influenced by the behavior of other locations within the same retinal nerve fiber bundle distribution than by those which are outside it: the closest topological neighbors may not be the closest 'functional' neighbors. It is possible that each test location will require its own unique convolution mask for optimum performance. Research is currently being conducted to examine this hypothesis.[12]

The difference between the filtered and unfiltered data with regard to the mean slope of progressing points at detection time reflects the fact that gaussian filtering tends to lessen the slopes of the locations which are progressing at the greatest rate. Notwithstanding this, the fact that both the number of progressing points and the mean slope for the whole field at detection time are not affected by filtering suggests that, for these progression criteria, the overall 'progression status' of the field series are retained after filtering.

In summary, previous work has shown spatial filtering to be of potential benefit in the analysis of glaucomatous visual field progression.[3,4] This study has examined the combined use of spatial filtering and pointwise linear regression in the context of a

software package, PROGRESSOR for Windows. The study suggests that the 'blurring' effect inherent in the gaussian filtering process is not clinically significant.

Acknowledgments

Supported by the International Glaucoma Association, London, UK and the Medical Research Council, London, UK.

References

1. Boeglin RJ, Caprioli J, Zulauf M: Long-term fluctuation of the visual field in glaucoma. Am J Ophthalmol 113(4):396-400, 1992
2. Flammer J, Drance SM, Zulauf M: Differential light threshold: short- and long-term fluctuation in patients with glaucoma, normal controls, and patients with suspected glaucoma. Arch Ophthalmol 102(5):704-706, 1984
3. Fitzke FW, Crabb DP, McNaught AI, Edgar DF, Hitchings RA: Image processing of computerised visual field data. Br J Ophthalmol 79:207-212, 1995
4. Crabb DP, McNaught AI, Fitzke FW, Hitchings RA: Spatially enhanced modelling of sensitivity decay in low-tension glaucoma. In: Mills RP, Wall M (eds) Perimetry Update 1994/1995, pp 73-81. Amsterdam/New York: Kugler Publ 1995
5. Fitzke FW, Hitchings RA, Poinoosawmy D, McNaught AI, Crabb DP: Analysis of visual field progression in glaucoma. Br J Ophthalmol 80:40-48, 1996
6. McNaught AI, Crabb DP, Fitzke FW, Hitchings RA: Modelling series of visual fields to detect progression in normal tension glaucoma. Graefes Arch Clin Exp Ophthalmol 233:750-755, 1995
7. McNaught AI, Crabb DP, Fitzke FW, Hitchings RA: Visual field progression: comparison of Humphrey Statpac 2 and pointwise linear regression. Graefes Arch Clin Exp Ophthalmol 1996 (in press)
8. Werner EB, Adelson A, Krupin T: Effect of patient experience on the results of automated perimetry in clinically stable glaucoma patients. Ophthalmology 95:764-767, 1988
9. Werner EB, Krupin T, Adelson A, Feitl ME: Effect of patient experience on the results of automated perimetry in glaucoma suspect patients. Ophthalmology 97:44-48, 1990
10. Gonzales RC, Wintz P: Digital Image Processing, 2nd Edn. Massachusetts: Addison-Wesley 1987
11. Phillips D: Image Processing in C, 1st Edn. Kansas: R&D Publications Inc 1994
12. Crabb DP, Fitzke FW, McNaught AI, Hitchings RA: A profile of the spatial dependence of pointwise sensitivity across the glaucomatous visual field. This Volume

CLINICAL SCIENCES

Severity and Stability of Glaucoma

Patient Perception Compared With Objective Measurement

Ananth C. Viswanathan, FRCOphth; Andrew I. McNaught, FRCOphth; Darmalingun Poinoosawmy; Luigi Fontana, MD; David P. Crabb, PhD; Fred W. Fitzke, PhD; Roger A. Hitchings, FRCOphth

Objective: To elucidate the relationship between the subjective assessment in patients with glaucoma of (1) the severity of their visual loss, and (2) any deterioration in their visual function and their objective visual fields as measured by computed perimetry.

Design: First, patients completed a questionnaire relating to perceived visual disability and underwent binocular visual field testing. Second, a separate group of patients answered a question about perceived visual deterioration: their monocular visual field tests were analyzed retrospectively by pointwise linear regression to establish stability or deterioration.

Setting: The Glaucoma Service of a specialist eye hospital, which is a tertiary referral center and serves the local community.

Subjects: One hundred twenty-three patients with glaucoma including 62 for the severity arm of the study and 61 for the progression arm.

Main Outcome Measures: Questionnaire responses, Esterman binocular disability score, and objective visual field deterioration.

Results: Questions strongly associated with Esterman binocular disability scores related to bumping into things, problems with stairs, and finding things that have been dropped. There was a strong association between perceived visual deterioration and measured bilateral visual field deterioration ($P<.01$).

Conclusions: There is a strong association between some types of perceived visual disability and the severity of binocular field loss. A patient who notices gradual visual deterioration is twice as likely to have bilateral visual field deterioration as not. The findings in this sample of patients with mild-to-moderate glaucoma challenge the belief that glaucoma is an insidious process in which the symptoms do not appear until the end stage of the disease.

Arch Ophthalmol. 1999;117:450-454

From the Institute of Ophthalmology (Drs Viswanathan, McNaught, Crabb, and Fitzke) and Moorfields Eye Hospital (Mr Poinoosawmy and Drs Fontana and Hitchings).

AT PRESENT, glaucoma is the third most common cause of blindness in the world.[1] Based on evidence of increasing access to cataract surgery and increasing success in prevention and treatment of nutritive and infectious cause of blindness, glaucoma has been projected to become the most common cause of blindness in the year 2000.[2] It is estimated that approximately 5.2 million people are bilaterally blind from glaucoma: this represents 15% of the total burden of world blindness.[3] There are approximately 250 000 known individuals with glaucoma in the United Kingdom, and it is likely that an equal number of cases are undiagnosed.[4] With the shift toward an older population, the medical, social, and economic burdens imposed by glaucoma will increase.

Two of the most important facets of the disease status of patients with glaucoma are the extent of established optic nerve damage and the rate of progression of this damage. The combination of these static and dynamic components has a critical bearing on the patient's present and future visual function and on the clinical management used. This study was designed to investigate the relationship between objective measurements and subjective patient perception of these 2 components.

VISUAL FIELD DAMAGE

Visual field damage in glaucoma is conventionally measured using automated perimetry. Monocular testing is used to tailor therapy to each eye. However, the patient's visual disability through field damage is better assessed using binocular testing: binocular visual field tests are routinely used to measure whether a patient's visual field is legally adequate for driving.[5] The most widely used strategies for testing and scoring the binocular visual field are based on the system devised by Esterman, which was originally used with monocular fields[6] and was then

PATIENTS AND METHODS

SEVERITY OF VISUAL FIELD DAMAGE

To compare severity of visual field damage with perceived visual disability, patients consecutively attending the Glaucoma Service at Moorfields Eye Hospital, London, England, for visual field testing were studied. All patients gave informed consent for the study; there were no refusals to participate. Visual acuity in each eye was required to be better than 20/40 and eligible patients had no significant ocular pathological conditions apart from primary open-angle glaucoma. Sixty-two patients met these inclusion criteria. One of the investigators (D.P.) administered a brief binary forced-choice questionnaire to each patient. The questionnaire consisted of the following 10 questions related to visual disability derived from those validated in previous published research in this area.[8]

1. Do you ever notice that parts of your field of vision are missing?
2. Have you noticed any deterioration in your sight over the last few years?
3. Do you ever have trouble following a line of print or finding the next line when reading?
4. Do you notice variation in color intensity?
5. Do you bump into things sometimes?
6. Do you trip on things or have difficulty with stairs?
7. Have you had to give up activities because of your sight?
8. Do you have difficulty finding things that you have dropped?
9. Are you troubled by glare or dazzled on sunny days or in bright lighting?
10. Do you have particular difficulty seeing after moving from a light to a dark room?

Patients then underwent a binocular visual field test with a perimeter (Humphrey Field Analyzer, model 630; Humphrey Instruments Inc) using the 120-point Esterman strategy program. The Esterman disability score represents the proportion of points missed during the test. Thus, as the Esterman disability score increases, disability increases. The strength of association between responses to the questionnaire and Esterman disability scores was investigated using stepwise multiple regression and correlation analysis. In these analyses, an attempt was made to ascertain which of the questions are good predictors of the Esterman disability score. However, it was also possible to regard each of the questions as a possible outcome of the Esterman disability score, since poor binocular visual field may lead to problems with the performance of visual tasks. For this reason, the data were also analyzed using logistic regression: in this analysis, the Esterman disability score was regarded as the independent variable and each question as a separate dependent variable.

RATE OF VISUAL FIELD DETERIORATION

Sixty-one consecutive untreated patients attending the normal-tension glaucoma clinic at Moorfields Eye Hospital were studied. All patients gave informed consent for the study; there were no refusals to participate. Untreated patients were chosen because we believed it was important to analyze the natural history of glaucomatous visual damage and the concomitant perception of that damage in the absence of the potentially confounding effects of therapy. Patients were untreated if no evidence of progressive disease using global indices could be detected in the presence of open-angle glaucoma with normal intraocular pressure. Subsequent detection of progressive disease using pointwise analysis, combined with the demonstration that a 25% to 30% lowering of intraocular pressure reduces the rate of deterioration,[31,32] led to treatment being recommended. Patients had no significant ocular pathologic findings apart from normal-tension glaucoma. All patients underwent a series of at least 5 visual field tests in each eye during 16 months. The maximum number of visual field tests performed during follow-up was 23 and the maximum length of follow-up was 9.6 years.

Patients were asked the single question, "Have you noticed any deterioration in your sight over the last few years?" and were allowed a choice of yes or no. The results of the series of visual field tests for each eye were then assessed for progression or stability using PROGRESSOR. Patients were unaware of the results of the PROGRESSOR analysis at the time that the question was asked. A field series was regarded as progressing if PROGRESSOR identified at least 1 test location with a rate of decay of 1 dB per year or worse associated with $P<.01$ for a 2-tailed t test of the slope against 0 (ie, the null hypothesis of no deterioration). The slope criterion of 1 dB per year represents a rate of sensitivity loss approximately 10 times greater than the normal age-related decline.[33] Edge points are known to be more subject to fluctuation,[33] so a stricter slope criterion of 2 dB per year (also $P<.01$) was introduced for them. These slope criteria, in combination with a less stringent slope significance criterion of $P<.10$, have been demonstrated to compare closely with the analyses (Statpac 2 Glaucoma Change Probability; Humphrey Instruments, Inc) used in other studies.[26,30] The association between measured monocular or binocular visual field deterioration and subjective perception of deterioration was examined with the χ^2 statistic.

STATISTICAL ANALYSIS

Statistical analysis was performed using the Statistical Package for the Social Sciences (SPSS Inc, Chicago, Ill) for Windows (version 6.0; Microsoft Corporation, Redmond, Wash).

adapted for use with binocular fields.[7] Test points are more closely spaced in areas of the field considered more functionally important (**Figure 1**). This gives additional weight to these areas when the Esterman disability score (the proportion of points missed) is calculated. In a study[8] of patients with severe glaucomatous visual field loss, a moderate association was found between some questions about visual disability and the degree of binocular visual field damage as measured by the Esterman disability score. The modest ability of the questions as predictors of the Esterman disability score was ascribed to denial of visual disability and the use of adaptive strategies. One of the aims of the present study was to examine the strength of association between subjective patient perception of visual disability (using questions similar to those used by previous workers[8] to enable comparison) and objective measurements of functional vision (Esterman disability score).

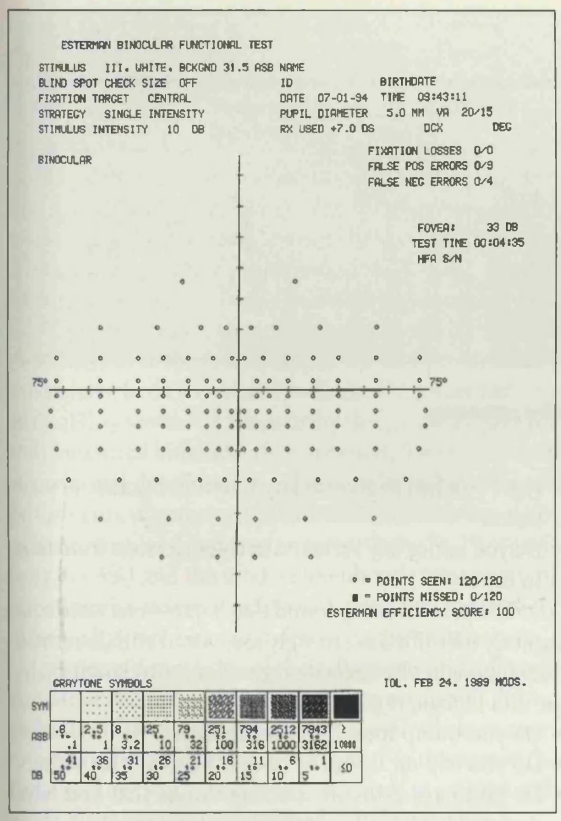


Figure 1. *Example of a printout of an Esterman strategy test (Humphrey Field Analyzer; Humphrey Instruments Inc, San Leandro, Calif).*

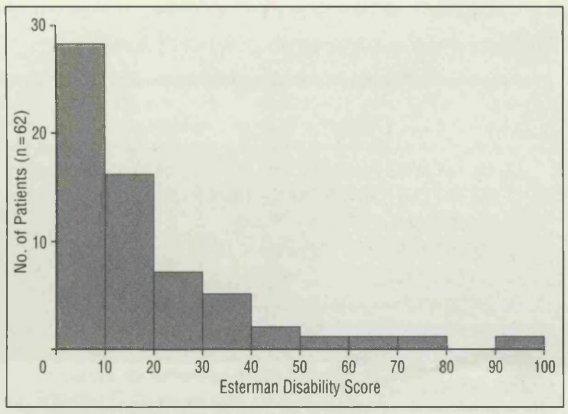


Figure 2. *Esterman disability scores.*

RATE OF VISUAL FIELD DETERIORATION

Although the rate at which the visual field deteriorates is one of the defining factors of the severity of glaucoma, there is no clear consensus on the criteria by which deterioration should be measured.[9-15] The use of clinical judgment alone has been found to be unreliable, even when performed by experienced observers,[16] so a variety of statistical techniques have been applied to estimate change in glaucomatous visual fields.

Methods based on estimates of change in summary measures of the field, such as regression analysis of the mean defect value,[17] mean deviation,[15] other global measures,[15] measurement of whole-field and quadrantic sensitivity losses,[18] and trend and regression analysis of various estimates of the sensitivity of the whole field or parts of it,[19-21] largely or completely ignore the detailed spatial information contained within computed visual field tests and are insensitive to early localized change.[22] Furthermore, different regions of the visual field may deteriorate at different rates.[10,21,23] The analysis of summary measures, whether based on the whole field or clusters of points within it, have been found to be inadequate at detecting glaucomatous change.[24,25]

We used a pointwise linear regression (PROGRESSOR; OBF Laboratories, Malmesbury, England)[26] that analyzes visual field deterioration using pointwise linear regression of sensitivity on time. This technique has been used for several years to investigate glaucomatous visual field change[27,28] and has recently been reexamined.[15] PROGRESSOR produces a cumulative graphical output. The pointwise linear model has been demonstrated to provide a valid framework for detecting and forecasting glaucomatous loss[29] and has been found to compare favorably with other pointwise methods of analysis (Statpac 2 Glaucoma Change Probability; Humphrey Instruments Inc, San Leandro, Calif).[30]

This study investigates the relationship between perceived visual field deterioration (assessed by questionnaire) and objectively measured field deterioration (using PROGRESSOR).

RESULTS

SEVERITY OF VISUAL FIELD DAMAGE

Esterman binocular disability scores ranged from 0 to 91.7. The mean (SD) was 12.85 (18.12) and the median was 7.08. The distribution of Esterman binocular disability scores is shown in **Figure 2**.

Stepwise multiple regression identified 3 questions as predictors of the Esterman binocular disability scores: question 5, question 6, and question 8. The adjusted R^2 resulting from the analysis was 0.34.

For comparison with previous studies,[8] a correlation coefficient value of 0.4 or greater indicated a noteworthy association between the question concerned and the Esterman binocular disability score. Questions satisfying this criterion were the same as those identified by the multiple regression analysis (questions 5, 6, and 8) with the addition of question 4.

Logistic regression identified the same questions as correlation analysis (questions 4, 5, 6, and 8). It also identified question 1, question 3, and question 10. These results are summarized in the **Table**.

RATE OF VISUAL FIELD DETERIORATION

Of the 61 patients questioned, 29 reported a subjective perception of gradual deterioration of vision and 32 reported no such perception. Of the 29 patients reporting visual deterioration, 22 had field deterioration in at least 1 eye as shown by the results of PROGRESSOR: 16 of these patients had deterioration in both eyes. Of the 32 patients reporting no perceived deterioration in vision, 19 were found in the results of PROGRESSOR to have field deterioration in at least 1 eye and 5 had field dete-

Questions Associated With Esterman Disability Score*

Question	Multiple Regression T Ratio	Correlation Coefficient	Logistic Regression P
1. Do you ever notice that parts of your field of vision are missing?	...	...	.01
3. Do you ever have trouble following a line of print or finding the next line when reading?	...	...	.01
4. Do you notice variation in color intensity?	...	0.41	.009
5. Do you bump into things sometimes?	2.10	0.51	.002†
6. Do you trip on things or have difficulty with stairs?	2.02	0.51	.002†
8. Do you have difficulty finding things that you have dropped?	1.93	0.55	.001†
10. Do you have particular difficulty seeing after moving from a light to a dark room?	...	...	.01

*Ellipses indicate not applicable.
†These questions are significant at the 5% level after Bonferroni correction[34] for multiple comparisons.

rioration in both eyes. The χ^2 statistic indicates a significant association ($P<.01$) between subjectively perceived visual deterioration and measured binocular visual field deterioration. If a subjective report of deterioration is regarded as a test for binocular deterioration, the sensitivity is 76.2% (16 of 21 patients) and the specificity is 67.5% (27 of 40 patients). The corresponding figures for monocular deterioration in either eye are a sensitivity of 53.7% (22 of 41 patients) and a specificity of 65% (13 of 20 patients). The likelihood ratio (the odds that a patient reporting visual deterioration has measurable binocular deterioration) is 2.34 (76.2/[100-67.5]). Of the 16 patients with measured binocular deterioration who were aware of visual field deterioration, 15 showed either deterioration at the same location in each eye or deterioration in 1 eye corresponding to a previously established scotoma in the other eye. Of the 5 patients with binocular deterioration who were unaware of visual field deterioration, 3 showed this type of behavior but 2 did not.

COMMENT

As mentioned previously, the present study was partly inspired by work carried out by Mills and Drance[8] in 1986. They selected 42 patients with severe visual loss due to glaucoma and examined the association between questionnaire responses and Esterman disability scores. Although the methods used in the present study and those of Mills and Drance are similar, the aims and patient selection criteria of the studies are different: Mills and Drance selected patients on the basis of severe visual loss (in 20 of the 42 patients, glaucoma was not the primary reason for impaired vision), whereas the present study concentrated on patients with good visual acuity in whom glaucoma was the primary reason for visual loss. Given these differences, it is interesting that 3 of the 5 questions that Mills and Drance found to have a correlation coefficient of greater than 0.4 were also identified as such in the present study: these were questions related to color intensity (question 4), bumping into things (question 5), and tripping (question 6). Mills and Drance described 4 questions as important according to both stepwise regression and correlation analysis:

- Do you have trouble following a line of print or finding the next line?
- Do you bump into things?
- Have you had to give up any activities because of your vision?
- Do you notice any variation in color richness from time to time?

The present study found that 3 questions were consistently identified as strongly associated with Esterman disability scores by stepwise regression, correlation analysis, and logistic regression:

- Do you bump into things sometimes?
- Do you trip on things or have difficulty with stairs?
- Do you have difficulty finding things that you have dropped?

These findings suggest that trouble following print and having to give up activities are a feature of the severe visual disability more prevalent in the patient group studied by Mills and Drance, whereas bumping into things is related to the level of visual disability for patients with glaucoma with mild-to-moderate visual field damage as well as those whose vision is more severely impaired. The question about tripping was excluded from the multiple regression analysis in the study by Mills and Drance because it was found to be closely related to the question about bumping into things. This was not the case in the present study, perhaps because the addition of the phrase "or have difficulty with stairs" was sufficient to differentiate it from the question about bumping into things. The question "Do you have difficulty finding things that you have dropped?"—which the present study found to have an important association with binocular visual disability—was derived from the question "Do you have trouble locating things?"—which Mills and Drance did not find to be associated with the Esterman disability score. This difference may be a result of the difference between the patient groups in the 2 studies or it may be because the former question is related to a specific visual task: patients may find it easier to remember difficulty, or the lack of it, in this area. The fact that the 3 questions identified by this study as strongly associated with the Esterman disability score all relate to visual tasks primarily involving the inferior visual field may reflect the fact that the Esterman test grid is biased toward the inferior visual field

Although the 3 questions (questions 5, 6, and 8) mentioned earlier were found to have a strong association with Esterman disability score by all 3 methods of analysis (stepwise regression, correlation analysis, and logistic regression) the adjusted R^2 of the multiple regression analysis is 0.34: the questions only explain 34% of the variance of

the Esterman disability score, so they are not viable as predictors of it. (Mills and Drance found an adjusted R^2 of 48% for their explanatory questions.)

The findings of an association between visual field deterioration in both eyes and the subjective perception of visual deterioration, but no association between visual field deterioration in at least 1 eye and subjective deterioration, suggests that a stable visual field in 1 eye tends to mask the perception of deterioration in the other. It is unlikely that the degree of initial visual field damage is a confounding variable for the perception of visual deterioration: if this were the case, question 2 of the questionnaire would have been associated with the Esterman binocular disability score. It is noteworthy that, of the 21 patients with measured binocular deterioration, 5 were unaware of any visual deterioration. This is reflected in the poor performance of perceived deterioration as a marker of objective binocular deterioration (sensitivity, 76.2%; specificity, 67.5%) and the modest likelihood ratio: a patient reporting visual deterioration was 2.34 times more likely to have objective binocular deterioration than not.

Patients' knowledge of their diagnosis and information contained about the status of their eye health during previous visits would be likely to influence their answers to questions relating to visual function. For this reason the findings of our study cannot be generalized to patients with newly diagnosed or undiagnosed glaucoma.

Previous studies have demonstrated moderate correlation between the perceived visual disability and visual field damage as measured by automated perimetry in severe glaucoma[35] and moderate or severe glaucoma.[36,37] Taken together, the results of this study suggest that, in mild-to-moderate glaucoma, both the existing severity of visual field damage in glaucoma and its rate of deterioration are associated with corresponding subjective perceptions of visual disability and deterioration. This challenges the belief that glaucoma is an insidious process in which the symptoms do not appear until the end stage of the disease. However, patient perception is a poor predictor of objective measurement of the visual field: the latter will remain paramount as means of measuring the visual consequences of the disease or of any therapy.

Accepted for publication November 17, 1998.

Supported in part by grants from the International Glaucoma Association, the Royal National Institute for the Blind, and the Medical Research Council, London, England.

Presented in part at the Fifth Congress of the European Glaucoma Society, Paris, France, June 20, 1996.

The Institute of Ophthalmology retains the intellectual property rights to the PROGRESSOR software used in this study.

Reprints: Fred W. Fitzke, PhD, Institute of Ophthalmology, 11-43 Bath St, London EC1V 9EL, England (e-mail: f.fitzke@ucl.ac.uk).

REFERENCES

1. Foster A, Johnson GJ. Magnitude and causes of blindness in the developing world. *Int Ophthalmol.* 1990;14:135-140.
2. Quigley HA. Number of people with glaucoma worldwide. *Br J Ophthalmol.* 1996; 80:389-393.
3. Thylefors B, Negrel AD. The global impact of glaucoma. *Bull World Health Organ.* 1994;72:323-326.
4. Coffey M, Reidy A, Wormald R, Xian WX, Wright L, Courtney P. Prevalence of glaucoma in the west of Ireland. *Br J Ophthalmol.* 1993;77:17-21.
5. Munton G. Vision. In: Taylor JF, ed. *Medical Aspects of Fitness to Drive.* 5th ed. London, England: Medical Commission on Accident Prevention; 1995:118-132.
6. Esterman B. Grid for scoring visual fields, II: perimeter. *Arch Ophthalmol.* 1968; 79:400-406.
7. Esterman B. Functional scoring of the binocular field. *Ophthalmology.* 1982;89: 1226-1234.
8. Mills RP, Drance SM. Esterman disability rating in severe glaucoma. *Ophthalmology.* 1986;93:371-378.
9. Zulauf M, Caprioli J. What constitutes progression of visual field defects? *Semin Ophthalmol.* 1992;7:130-146.
10. Hoskins HD, Jensvold N, Zaretsky M, Hetherington J. Rate of progression of discrete areas of the visual field. In: Heijl A, ed. *Perimetry Update 1988/1989.* Amsterdam, the Netherlands: Kugler & Ghedini; 1989:173-176.
11. Mikelberg FS. Do computerised visual fields and automated optic disc analysis assist in the choice of therapy in glaucoma? *Eye.* 1992;6(pt 1):47-49.
12. Hitchings K. Psychophysical testing in glaucoma. *Br J Ophthalmol.* 1993;77: 471-472.
13. Johnson CA. Modern developments in clinical perimetry. *Curr Opin Ophthalmol.* 1993;4:7-13.
14. Fitzke FW, McNaught AI. The diagnosis of visual field progression in glaucoma. *Curr Opin Ophthalmol.* 1994;5:110-115.
15. Smith SD, Katz J, Quigley HA. Analysis of progressive change in automated visual fields in glaucoma. *Invest Ophthalmol Vis Sci.* 1996;37:1419-1428.
16. Werner EB, Bishop KI, Koelle J, et al. A comparison of experienced clinical observers and statistical tests in detection of progressive visual field loss in glaucoma using automated perimetry. *Arch Ophthalmol.* 1988;106:619-623.
17. Weber J, Koll W, Krieglstein GK. Intraocular pressure and visual field decay in chronic glaucoma. *Ger J Ophthalmol.* 1993;2:165-169.
18. Wegner A, Ugi I, Hofman A. A long-term visual field evaluation of glaucoma patients treated topically with timolol or carteolol. In: Mills RP, ed. *Perimetry Update 1992/1993.* Amsterdam, the Netherlands: Kugler & Ghedini; 1993: 143-145.
19. Holmin C, Krakau CE. Regression analysis of the central visual field in chronic glaucoma cases. *Acta Ophthalmol (Copenh).* 1982;60:267-274.
20. Wu D, Schwartz B, Nagin P. Trend analyses of automated visual fields. *Doc Ophthalmol Proc Ser.* 1987;49:175-189.
21. O'Brien C, Schwartz B, Takamoto T, Wu DC. Intraocular pressure and the rate of visual field loss in chronic open-angle glaucoma. *Am J Ophthalmol.* 1991;111: 491-500.
22. Wild JM, Hussey MK, Flanagan JG, Trope GE. Pointwise topographical and longitudinal modeling of the visual field in glaucoma. *Invest Ophthalmol Vis Sci.* 1993;34:1907-1916.
23. O'Brien C, Schwartz B. The visual field in chronic open-angle glaucoma. *Eye.* 1990; 4(pt 4):557-562.
24. Chauhan BC, Drance SM, Douglas GR. The use of visual field indices in detecting changes in the visual field in glaucoma. *Invest Ophthalmol Vis Sci.* 1990;31: 512-520.
25. Birch MK, Wishart PK, O'Donnell N. Determining progressive field loss. In: Mills RP, Wall M, eds. *Perimetry Update 1994/1995.* Amsterdam, the Netherlands: Kugler & Ghedini; 1995:31-36.
26. Fitzke FW, Hitchings RA, Poinoosawmy D, McNaught AI, Crabb DP. Analysis of visual field progression in glaucoma. *Br J Ophthalmol.* 1996;80:40-48.
27. Noureddin BN, Poinoosawmy D, Fitzke FW, Hitchings RA. Regression analysis of visual field progression in low tension glaucoma. *Br J Ophthalmol.* 1991;75: 493-495.
28. Poinoosawmy D, Wu J, Fitzke FW, Hitchings RA. Discrimination between progression and non-progression visual field loss in low tension glaucoma using MDT. In: Mills-RP, ed. *Perimetry Update 1992/1993.* Amsterdam, the Netherlands: Kugler & Ghedini; 1993:109-114.
29. McNaught AI, Crabb DP, Fitzke FW, Hitchings RA. Modelling series of visual fields to detect progression in normal tension glaucoma. *Graefes Arch Clin Exp Ophthalmol.* 1995;233:750-755.
30. McNaught AI, Crabb DP, Fitzke FW, Hitchings RA. Visual field progression. *Graefes Arch Clin Exp Ophthalmol.* 1996;234:411-418.
31. Bhandari A, Crabb DP, Poinoosawmy D, Fitzke FW, Hitchings RA, Noureddin BN. Effect of surgery on visual field progression in normal-tension glaucoma. *Ophthalmology.* 1997;104:1131-1137.
32. Koseki N, Araie M, Shirato S, Yamamoto S. Effect of trabeculectomy on visual field performance in central 30 degrees field in progressive normal-tension glaucoma. *Ophthalmology.* 1997;104:197-201.
33. Heijl A, Lindgren G, Olsson J. Normal variability of static perimetric threshold values across the central visual field. *Arch Ophthalmol.* 1987;105:1544-1549.
34. Bland JM, Altman DG. Multiple significance tests: the Bonferroni method. *BMJ.* 1995;310:170.
35. Choy ES, Mills RP, Drance SM. Automated Esterman testing of disability in glaucoma. In: Greve E, Heijl A, eds. *Seventh International Visual Field Symposium.* Amsterdam, the Netherlands: Junk Publishers; 1986:527-535.
36. Parrish RK II. Visual impairment, visual functioning, and quality of life assessments in patients with glaucoma. *Trans Am Ophthalmol Soc.* 1996;94:919-1028.
37. Parrish RK II, Gedde SJ, Scott IU, et al. Visual function and quality of life among patients with glaucoma. *Arch Ophthalmol.* 1997;115:1447-1455.

Printed and Published in the United States of America

Graefe's Arch Clin Exp Ophthalmol (1997)
235:563–568

CLINICAL INVESTIGATION

Ananth C. Viswanathan
Roger A. Hitchings
Fred W. Fitzke

How often do patients need visual field tests?

Received: 21 February 1997
Accepted: 5 March 1997

Presented in part at the Association for Research in Vision and Ophthalmology Annual Meeting, May 1995 and at the American Academy of Ophthalmology Meeting, October 1995.

The authors have no proprietary interest in any of the materials used in this study.

A.C. Viswanathan · F.W. Fitzke (✉)
Institute of Ophthalmology,
11-43 Bath Street, London EC1V 9EL, UK

R.A. Hitchings
Moorfields Eye Hospital, City Road,
London EC1V 2PD, UK

Abstract ● Background: This study was undertaken to determine whether the interval between visual field tests affects the ability to detect progressive glaucomatous field loss.
● Methods: One hundred and nineteen retinal locations which were deteriorating significantly by ≥1 dB/year (untreated normal tension glaucoma patients: 6 eyes) were studied. Analysis was repeated using 'thinned' visual field tests: one test per year instead of the complete three per year over a period of 4 years.
● Results: The 'thinned' tests identified only 45.4% of the deteriorating points over the 4-year period. Furthermore, there was a mean delay of 1.10 years in detection ($P<0.01$).
● Conclusions: Less frequent visual field testing detects fewer progressing locations and detects them later.

Introduction

When considering how often to measure the visual field of a glaucoma patient, factors such as the type of glaucoma, the number and results of previous tests, the age of the patient, recent changes in glaucoma therapy and the ability of the patient to produce reliable test results must be borne in mind. However, one of the commonest and most important reasons for repeatedly measuring visual fields is in order to detect progressive damage as early as possible. It is to this aim that the present study is directed.

The use of computerized perimetry allows each retinal location tested to be ascribed a number representing the luminance sensitivity at that point. Thus a series of visual fields is represented by a series of grids of numbers. The interpretation of these grids in order to diagnose stable or progressive disease is not straightforward: there is inherent variability in the data obtained from automated perimetry, and the criteria by which progression should be meaured is a continuing subject of debate [4, 5, 10, 13, 16, 29].

Several approaches have been taken to attempt to distinguish true progression obscured by 'noise' owing to the variability of patient response. These apply a wide spectrum of statistical techniques to the analysis of consecutive visual fields. The Statpac 2 [9] and Delta [1] programs include estimates of the probability of change of individual test locations compared with normal reference data and with a baseline obtained from the patient's initial visual field tests. These methods rely on the database of normal values being a representative sample of the population from which the patient under test is drawn, and are subject to the criticism that information from tests after the baseline is ignored in the analysis. Other methods which use information from all the tests in a series include regression analysis of the mean defect value [24], measurement of whole-field and quadrantic sensitivity losses [25], the rate of sensitivity loss with the number of deteriorating locations [22], the visual field coefficient [3] and trend and regression analysis of various estimates of the

sensitivity of the whole field or parts of it [11, 20, 28]. These summary measures largely or completely ignore the detailed spatial information contained within computerized field tests: different regions of the visual field may deteriorate at different rates [12, 18, 20].

The problems with the methods outlined above have led to methods of analysis which concentrate upon the behaviour of individual test locations over a whole series of visual field tests, such as pointwise linear regression of sensitivity on time [17, 19, 21]. This technique demonstrates field loss undetected by summary measures of sensitivity and is a good predictor of the future behaviour of the field [6].

With respect to point-by-point analysis of sequential fields, it has been demonstrated that the most reliable estimates of progressive glaucomatous visual field decay are obtained by using a linear fit, i.e. postulating a constant reduction of sensitivity over time for each retinal location tested [15]. Thus it would seem self-evident that increasing the frequency of visual field tests would provide a reliable estimate of the rate of decay more quickly and therefore lead to the detection of progressing retinal locations sooner. This has been assumed [14, 23], but not directly addressed.

However, there is a significant degree of variation both within a visual field test (short-term fluctuation, SF) [7] and between tests (long-term fluctuation, LF) [2]. These contribute 'noise' to the underlying 'signal': both SF and LF are greater in patients with glaucoma than in normals [8]. When attempting to estimate the underlying behaviour of a signal in a noisy environment from a series of samples, the accuracy of the estimate increases as the interval between samples falls and the number of samples grows. However, it is possible that if the frequency of visual field testing rises above a certain level the estimate of visual field decay will cease to improve significantly, since noise rather than signal will predominate. If this effect occurred at the frequencies of visual field testing commonly used in clinical practice for the follow-up of glaucoma patients (i.e. 4 months to 1 year) it would have important implications for cost-benefit analysis. The aim of this study was to ascertain whether increasing the frequency of visual field testing from 1 test per year to 3 tests per year provides a useful improvement in the detection of progressing points.

Materials and methods

Subjects

The Moorfields Eye Hospital visual field database contains 64949 automated visual field records from 9482 patients. Records were selected for study on the basis of the following criteria:

1. Untreated normal tension glaucoma patients were chosen, as it was felt important to analyse the natural history of glaucomatous visual damage in the absence of therapy: current clinical practice dictates that intraocular pressure (IOP) should be lowered in confirmed early primary open angle glaucoma (POAG), whereas no consensus exists regarding normal-tension glaucoma (NTG). Following the demonstration that lowering IOP by 25% or more significantly affects the rate of progression (A. Bhandari, D. Poinoosawmy, J. Wu, R.A. Hitchings, presentation at the ARVO Annual Meeting 1995), this option is offered to all patients. A cohort of 220 patients who were untreated at the time of the study and whose diagnosis had been confirmed on phasing (IOP≤21 mmHg) was identified.
2. Their visual fields appeared to be deteriorating progressively in a typically glaucomatous manner based on the subjective clinical assessment of an experienced observer: serial Humphrey 30-2 printouts were examined, and test series demonstrating unequivocal glaucomatous change based on global indices, total deviation, pattern deviation or grey scales were selected.
3. They had had visual field tests at fairly regularly spaced intervals (Table 1). This was necessary in order to simulate less frequent tests by 'thinning' the examinations (see below).
4. All subjects were experienced in Humphrey 30-2 tests and able to produce reliable computerized visual fields results (less than 30% fixation losses and false negatives, less than 15% false positives). Each had had at least two tests over 4 months prior to the observation period: this is sufficient to obviate any learning effects [26, 27] which may delay the diagnosis of progression.
5. All subjects had visual acuity of 20/40 or better. None had significant ocular pathology apart from NTG.

On the basis of the foregoing criteria, six eyes from six subjects were selected. The subjects had an age range of 44–67 years (58.0

Table 1 Times of visual field tests in years from baseline (times included in the 'thinned' set are shown in bold)

	Patient 1	Patient 2	Patient 3	Patient 4	Patient 5	Patient 6
Time (test 1)/years	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
Time (test 2)/years	0.499	0.057	0.556	**1.285**	0.269	**1.018**
Time (test 3)/years	0.690	0.574	0.824	1.440	0.538	1.286
Time (test 4)/years	**1.036**	**0.863**	**1.110**	1.553	**0.898**	1.536
Time (test 5)/years	1.611	1.247	1.321	1.819	1.244	1.782
Time (test 6)/years	1.858	1.515	1.781	**2.164**	1.513	**2.204**
Time (test 7)/years	**2.068**	**1.995**	**2.011**	2.452	**1.986**	2.511
Time (test 8)/years	1.509	2.394	2.299	**2.912**	2.337	**2.818**
Time (test 9)/years	**3.022**	2.739	2.701	3.238	2.647	3.335
Time (test 10)/years	3.487	**3.066**	**2.951**	3.545	**2.973**	3.683
Time (test 11)/years	3.718	3.871	3.643	3.894	3.260	4.009
Time (test 12)/years	**3.970**	**4.258**	**3.816**	**4.047**	**3.644**	**4.373**

Mean interval between tests: 4.60±2.20 months (complete fields)
12.1±2.4 months ('thinned' fields)
Mean period of observation: 4.02±0.27 years

years±10.33 years) at the start of the observation period. Four of the subjects were female, two were male.

Testing strategy

All tests were performed on a standard Humphrey model 630 automated perimeter. The 30-2 full-threshold program with standard 4-2 dB double reversal strategy was used throughout.

Progression criteria

In order to identify whether a given retinal location was progressively deteriorating, linear regression of retinal sensitivity on time was performed. A negative slope of 1 dB per year or worse associated with $P<0.1$ for a two-tailed t-test of the slope against zero (i.e. the null hypothesis of no deterioration) was used to diagnose progression for a given location. These criteria have been demonstrated to compare closely with the Humphrey Statpac 2 Glaucoma Change Probability analysis [6]. Pointwise linear regression was perferred to Statpac 2 because this study demanded a reliable estimate of progression for each individual field test. Statpac 2 cannot provide this: it is more effective at diagnosing progression when a location is repeatedly ascribed a high probability of change over a series of consecutive tests [9].

Detection of progression

Test locations satisfying the progression criteria when pointwise linear regression was performed using all the field test in the 4-year (approx.) observation period for a given eye were defined as 'gold standard' progression points. For each of these points, the earliest test in which the progressing criteria were satisfied was defined as the time of detection of progression. The first test in the observation period was taken as baseline (time=0).

'Thinning' the tests

Initially, detection times were calculated for all progressing points. Next, in order to simulate a decrease in frequency of testing, the analysis was repeated using only one test per year for each eye instead of the complete three per year. For example, if tests were performed at times 0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44 and 48 months the 'thinned' tests would be those done at 0, 12, 24, 36, and 48 months. Table 1 shows the actual times for both complete and 'thinned' sets of tests for each patient.

Statistical analysis

Progressing points which were detected using the 'thinned' tests were identified. Since, by definition, all progressing points were detected using the complete series of tests, the sensitivity and specificity of the 'thinned' tests could be calculated relative to the gold standard of the complete fields.

For those points which were detected by the 'thinned' tests, detection times were compared with their correlates in the complete set of tests using a non-parametric test for paired data from two related samples (Wilcoxon signed rank Z test).

Results

Ability to detect progression

One hundred and nineteen locations were diagnosed as gold standard progressing points by the complete set of visual field tests. Of these, only 54 were also identified by the 'thinned' set of tests. Thus the sensitivity of the 'thinned' fields is 54÷119×100=45.4%. Of the 337 points diagnosed as not progressing using the complete tests, 321 were also identified as not progressing by the thinned' tests. The specificity of the 'thinned' tests is thus 321÷337×100=95.3%.

The likelihood ratio (the ratio of the probability that a point is progressing, given that it is identified by the 'thinned' tests as such, to the corresponding probability if it is not progressing) is 9.55. The 'thinned' tests gave rise to 16 false positives and 65 false negatives (as measured against the gold standard of the complete tests). These findings are summarized as a contingency table in Table 2.

Detection times

The complete set of tests gave a mean detection time of 2.52 years with a standard deviation of 1.10 years. The 'thinned' set of tests gave a mean detection time of 3.46 years with a standard deviation of 0.78 years. These findings are displayed as a drop-line graph in Fig. 1. The 'thinned' tests give consistently later detection times than the complete set of tests; there are only three points for which the detection time for the complete tests (represented by a solid circle) is greater than that for the 'thinned' tests (represented by a hollow circle). These points are amongst the latest detection times for the complete tests.

For those progressing points which were detected by the 'thinned' tests, delay was calculated as the difference between the detection time for the complete tests and that for the 'thinned' tests. The mean delay in detection associated with the 'thinned' tests was 1.10 years and the standard deviation was 1.12 years. As can be seen in Fig. 2,

Table 2 Contingency table for the performance of the 'thinned' set of visual field tests

[sensitivity=54÷119×100=45.4%; specificity=321÷337×100=95.3%; Likelihood ratio=sensitivity/(1-specificity)=9.55]

		State according to complete set of tests		
		Progressing	Not progressing	Total
State according to 'thinned' set of tests	Progressing	54	16	70
	Not progressing	65	321	386
	Total	119	337	76×6=456

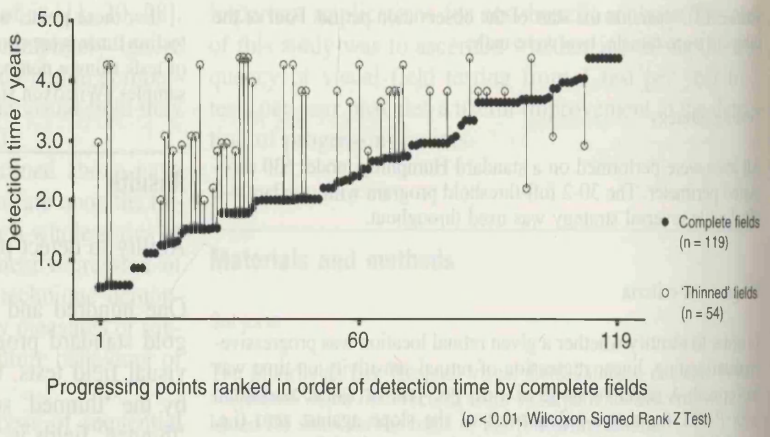


Fig. 1 Detection times for complete and thinned' visual field tests

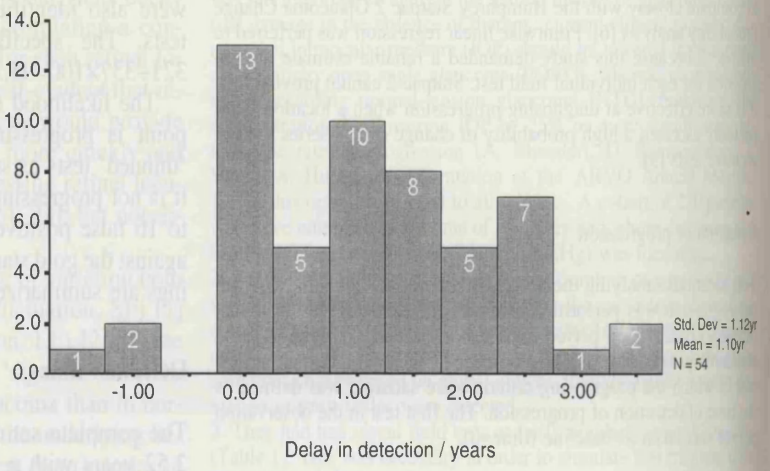


Fig. 2 Histogram of delay in detection of progressing points using 'thinned' visual field tests

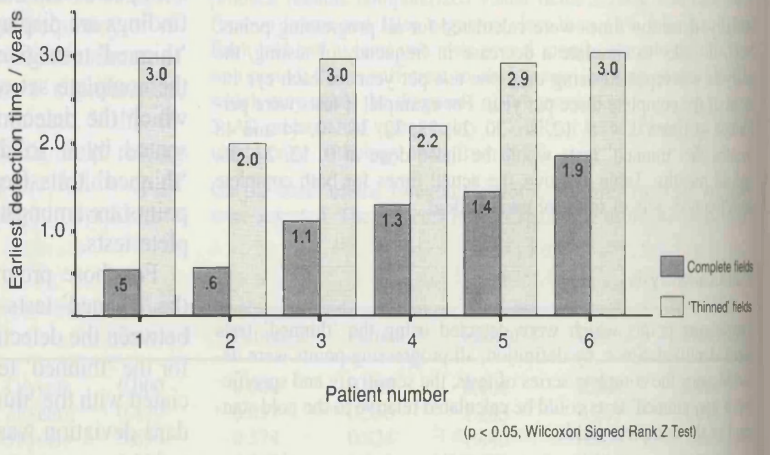


Fig. 3 Earliest detection times for each patient

the great majority of values of delay are greater than zero. In other words, the complete set of tests detect progression sooner than the 'thinned' tests. This is borne out by the results of the Wilcoxon signed rank Z test ($P<0.01$, Table 3).

For each patient, the earliest detection time for any progressing point in the whole visual field is the time at which that patient might first be said to have progressive disease. As Fig. 3 shows, these times are consistently lower for the complete tests than for the 'thinned' tests ($P<0.05$). The mean value of the delay associated with the 'thinned' tests for these earliest whole-field detection times is 1.54 years with a standard deviation of 0.54 years.

Table 3 Comparison of detection times between complete and 'thinned' sets of visual field tester (Wilcoxon signed rank Z test; $Z=-5.1816$, $P<0.01$)

	Cases	Mean rank
Negative ranks[a]	3	17.00
Positive ranks[b]	41	22.90
Ties[c]	10	
Total	54	

[a] Negative ranks occur when the 'thinned' set detects progression at a given location earlier than the complete set of tests
[b] Positive ranks occur when the complete set detects progression at a given location earlier than the 'thinned' set of tests
[c] Ties occur when both sets of tests detect progression at a given location at the same time

Discussion

One of the main aims of administering regular visual field tests to glaucoma patients is in order to detect progressive disease so that appropriate therapeutic measures may be instituted.

This study demonstrates that an increase in the frequency of visual field testing from one examination per year to three per year provides more rapid, more sensitive diagnosis of progressive glaucoma. It is possible that these effects have been underestimated owing to the technique of 'thinning' the tests. In reality, patients who have undergone one visual field test per year over a given period will be less experienced subjects than those performing three tests per year over the same period. For this reason the former group may take longer to overcome learning effects [26, 27] and to produce reliable test results. This effect has not been accounted for in the present study, since patients are effectively acting as internal controls: the 'thinned' tests are a subset of the complete tests.

In addition to the generalized delay of detection of progression associated with a simulated lower frequency of field testing (Fig. 2, Table 3), the delay for the earliest whole-field detection times (Fig. 3) is of particular interest. These times correspond to the times at which progressive disease in at least one retinal location would be detected and decisions concerning the clinical management of the patient would be prompted. In our stimulation of less frequent visual field testing this delay is marked: 1.54 years over an observation period of 4.02 years.

The sample size in this study was necessarily quite small (119 retinal locations, six eyes), since patients were chosen, albeit from a very large database, according to strict criteria. These criteria were chosen to isolate cases worthy of initial study. The high statistical significance of the results, notwithstanding the small sample size, appears to validate the criteria. Further work is required in order to determine whether these results may be generalized to patients with other forms of glaucoma and to those who have received or are receiving medical treatment.

Leaving aside the benefits which earlier diagnosis and treatment of progressive glaucoma may confer upon the individual patient, in pure cost terms the increased use of resources implicit in performing visual field tests three times per year instead of once per year must be set against the wider cost of allowing a potentially blinding disease to progress undetected.

Acknowledgements Ananth Viswanathan is supported by the International Glaucoma Association, King's College Hospital, Denmark Hill, London SE5 9RS, UK.

References

1. Bebie H, Frankhauser F (1982) Delta manual. Interzeag AG, Schlieren
2. Boeglin RJ, Caprioli J, Zulauf M (1992) Long-term fluctuation of the visual field in glaucoma. Am J Ophthalmol 113:396–400
3. Crick RP, Newson RB, Shipley MJ, Blackmore H, Bulpitt CJ (1990) The progress of the visual field in chronic simple glaucoma and ocular hypertension treated topically with pilocarpine or with timolol. Eye 4:563–571
4. Dunbar H-H Jr (1992) Does computerised perimetry offer practical advances in choice of therapy in the glaucoma patient? Eye 6:43–46
5. Fitzke FW, McNaught AI (1994) The diagnosis of visual field progression in glaucoma. Curr Opin Ophthalmol 5:110–115
6. Fitzke FW, Hitchings RA, Poinoosawmy D, McNaught AI, Crabb DP (1996) Analysis of visual field progression in glaucoma. Br J Ophthalmol 80:40–48
7. Flammer J (1986) The concept of visual field indices. Graefe's Arch Clin Exp Ophthalmol 224:389–392
8. Flammer J, Drance SM, Zulauf M (1984) Differential light threshold. Short- and long-term fluctuation in patients with glaucoma, normal controls, and patients with suspected glaucoma. Arch Ophthalmol 102:704–706
9. Heijl A, Lindgren G, Lindgren A et al. (1991) Extended empirical statistical package for evaluation of single and multiple fields in glaucoma: Statpac 2. In: Mills RP, Heijl A (eds) Perimetry update 1990/1991. Kugler & Ghedini, Amsterdam, pp 303–315
10. Hitchings RA (1993) Psychophysical testing in glaucoma. Br J Ophthalmol 77:471–472
11. Holmin C, Krakau CE (1982) Regression analysis of the central visual field in chronic glaucoma cases. A follow-up study using automatic perimetry. Acta Ophthalmol Copenh 60:267–274
12. Hoskins HD, Jensvold N, Zaretsky M, Hetherington J (1989) Rate of progression of discrete areas of the visual field. In: Heijl A (ed) Perimetry update 1988/1989. Kugler & Ghedini, Amsterdam, pp 173–176
13. Johnson CA (1993) Modern developments in clinical perimetry. Curr Opin Ophthalmol 4(11):7–13
14. Lachenmayr BJ, Vivell PM (1993) Perimetry and its clinical correlations. Thieme, New York

15. McNaught AI, Crabb DP, Fitzke FW, Hitchings RA (1995) Modelling series of visual vields to detect progression in normal tension glaucoma. Graefe's Arch Clin Exp Ophthalmol (233):750–755
16. Mikelberg FS (1992) Do computerised visual fields and automated optic disc analysis assist in the choice of therapy in glaucoma? Eye 6:47–49
17. Noureddin BN, Boinoosawmy D, Fitzke FW, Hitchings RA (1991) Regression analysis of visual field progression in low tension glaucoma. Br J Ophthalmol 75:493–495
18. O'Brien C, Schwartz B (1990) The visual field in chronic open angle glaucoma: the rate of change in different regions of the field. Eye 4:557–562
19. O'Brien C, Schwartz B (1993) Point by point linear regression analysis of automated visual fields in primary open-angle glaucoma. In: Mills RP (ed) Perimetry update 1992/1993. Kugler & Ghedini, Amsterdam, pp 149–152
20. O'Brien C, Schwartz B, Takamoto T, Wu DC (1991) Intraocular pressure and the rate of visual field loss in chronic open-angle glaucoma. Am J Ophthalmol 111:491–500
21. Poinoosawmy D, Wu J, Fitzke FW, Hitchings RA (1993) Discrimination between progression and non-progression visual field loss in low tension glaucoma using MDT. In: Mills RP (ed) Perimetry update 1992/1993. Kugler & Ghedini, Amsterdam, pp 109–114
22. Vogel R, Crick RP, Mills KB, Reynolds PM, Sass W, Clineschmidt CM, Tipping R (1992) Effect of timolol versus pilocarpine on visual field progression in patients with primary open-angle glaucoma. Ophthalmology 99:1505–151123.
Weber J, Diestelhorst M (1992) Perimetric follow-up in glaucoma with a reduced set of test points. German J Ophthalmol 1:409–414
24. Weber J, Koll W, Krieglstein GK (1993) Intraocular pressure and visual field decay in chronic glaucoma. German J Ophthalmol 2(3):165–169
25. Wegner A, Ugi I, Hofman A (1993) A long-term visual field evaluation of glaucoma patients treated topically with timolol or carteolol. In: Mills RP (ed) Perimetry update 1992/1993. Kugler & Ghedini, Amsterdam, pp 143–145
26. Werner EB, Adelson A, Krupin T (1988) Effect of patient experience on the results of automated perimetry in clinically stable glaucoma patients. Ophthalmology 95:764–767
27. Werner EB, Krupin T, Adelson A, Feitl ME (1990) Effect of patient experience on the results of automated perimetry in glaucoma suspect patients. Ophthalmology 97:44–48
28. Wu D, Schwartz B, Nagin P (1987) Trend analysis of automated visual fields. Doc Ophthalmol Proc Ser 49:175–189
29. Zulauf M, Caprioli J (1992) What constitutes progression of visual field defects? Semin Ophthalmol 7:130–146

Reprinted from BRITISH JOURNAL OF OPHTHALMOLOGY, December 1997, Vol 81, No 12, p 1037-1042

Early detection of visual field progression in glaucoma: a comparison of PROGRESSOR and STATPAC 2

Ananth C Viswanathan, Fred W Fitzke, Roger A Hitchings

Abstract

***Aim*—To compare the performance of PROGRESSOR (pointwise linear regression) and STATPAC 2 (comparison with baseline values) in detecting early deterioration in the visual fields of glaucoma patients.**

***Methods*—Visual field series from 19 untreated normal tension glaucoma eyes which were deteriorating on clinical grounds were analysed by PROGRESSOR and STATPAC 2. Progression criteria for PROGRESSOR were (1) inner points: slope < −1 dB/year, $p < 0.05$ and (2) edge points: slope < −2 dB/year, $p < 0.05$. Criteria for STATPAC 2 were $p < 0.05$ change probability for any point on three consecutive fields. Detection time was defined as the time interval between the initial field and the first field in which at least one progressing point was identified. Detection times produced by the two techniques were compared.**

***Results*—PROGRESSOR and STATPAC 2 agreed on progression in all 19 eyes. Mean detection time for PROGRESSOR was 1.077 (SD 0.985) years and for STATPAC 2 was 2.161 (1.357) years. PROGRESSOR detected progression sooner than STATPAC 2 in 18 eyes ($p < 0.01$, Wilcoxon matched pairs signed rank test). PROGRESSOR detected progression earlier by a mean of 1.085 (0.936) years.**

***Conclusions*—PROGRESSOR consistently detected progression earlier than STATPAC 2. The PROGRESSOR software is a useful tool for the early detection of visual field deterioration in glaucoma.**

(*Br J Ophthalmol* 1997;81:1037–1042)

Institute of Ophthalmology, London EC1V 9EL
A C Viswanathan
F W Fitzke

Moorfields Eye Hospital, City Road, London EC1V 2PD
R A Hitchings

Correspondence to:
Fred W Fitzke, PhD, Institute of Ophthalmology, 11-43 Bath Street, London EC1V 9EL.

Accepted for publication 28 April 1997

An important goal in the management of chronic glaucoma is the early, reliable detection of deterioration in the visual field. There is inherent 'noise' in the data obtained from serial automated field tests (long term fluctuation[1]): this fluctuation is greater in glaucoma patients than in normals.[2] The criteria by which progression should be measured are a continuing subject of debate.[3-8] Visual inspection of a series of fields and the use of clinical judgment are unreliable methods for diagnosing progression or stability, even when performed by experienced observers.[9] For this reason, and because the results of automated perimetry invite numerical analysis, a number of methods to estimate change in glaucomatous visual fields have been developed.

One group of methods rely on estimates of change in summary measures of the field such as regression analysis of the mean defect value,[10] mean deviation,[11] other global measures,[11] measurement of whole field and quadrantic sensitivity losses,[12] and trend and regression analysis of various estimates of the sensitivity of the whole field or parts of it.[13-15] However, the analysis of summary measures, whether based on the whole field or on clusters of points within it, has been found to be 'remarkably poor'[16] and 'of little value'[17] in detecting glaucomatous change. Summary measures largely or completely ignore the detailed spatial information contained within computerised field tests and are insensitive to early localised change.[18] Furthermore, different regions of the visual field may deteriorate at different rates.[4 15 19]

A widely available software package for estimating deterioration in serial glaucomatous visual fields is the STATPAC 2 glaucoma change probability analysis[20] for the Humphrey field analyser (Humphrey Instruments Inc, San Leandro, CA, USA). This software avoids the problems inherent in the methods discussed above by considering the field on a point by point basis. Two of the three initial fields are selected as a 'baseline' and subsequent fields are compared with this baseline in a pointwise manner. Points are labelled with a black triangle (Fig 1) if they are associated with a p value of <0.05 for change against a reference database.

PROGRESSOR[21] is a software package which analyses visual field progression using pointwise linear regression of sensitivity on time. This technique has been used for several years to investigate glaucomatous visual field change[22 23] and has recently been re-examined.[11] PROGRESSOR produces a cumulative graphical output as shown in Figure 2A. Each test location is shown as a bar graph in which each bar represents one test. The length of the bar relates to the depth of the defect (longer bars represent lower sensitivities) and the colour of the bar relates to the p value of the regression slope (Fig 2B). The pointwise linear model has been demonstrated to provide a valid framework for detecting and forecasting glaucomatous loss.[24]

PROGRESSOR has been found to emulate STATPAC 2 closely in diagnosing individual locations in a single field of a series as either

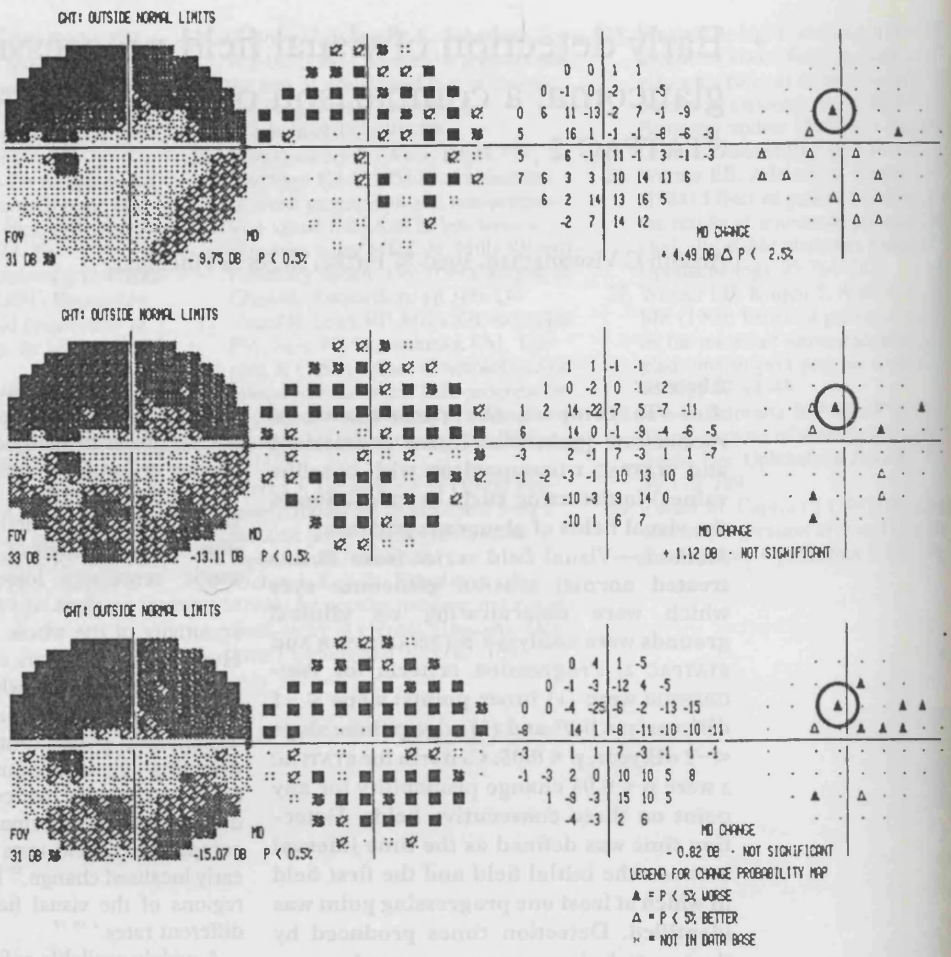


Figure 1 Example of a section of the STATPAC 2 glaucoma change probability analysis. The test location highlighted is labelled as showing significant deterioration in each of the three consecutive fields.

stable or progressing when the appropriate progression criteria are used.[25] However, the relative ability of the two algorithms to detect the first sign of deterioration has not been investigated directly. This is an important aspect of the technique's behaviour to compare, since the timely and reliable detection of glaucomatous visual field progression is of paramount clinical importance: in the presence of adequate intraocular pressure control, questions concerning the commencement or alteration of therapy are prompted only when definite visual deterioration is seen.

Methods

SUBJECTS

The Moorfields Eye Hospital visual field database currently contains 64 949 automated visual field records from 9482 patients. Records were selected for study on the basis of the following criteria:

(1) Untreated normal tension glaucoma patients were chosen as it was felt to be important to analyse the natural history of glaucomatous visual damage in the absence of the potentially confounding effects of medical therapy. A cohort of 220 such patients, whose diagnosis had been confirmed on phasing, was identified.

(2) Their visual fields showed progressive deterioration of a typically glaucomatous nature. This was done by inspection of the STATPAC overview analysis of serial Humphrey visual fields by an experienced observer.

(3) All subjects were experienced in Humphrey 30-2 tests and able to produce reliable computerised visual fields (less than 30% fixation losses and false negatives and less than 15% false positives). Each had had at least two tests over 4 months before the observation period: this is sufficient to obviate any learning effects[26 27] which may delay the diagnosis of progression.

(4) All subjects had visual acuity of 6/12 or better. None had significant ocular pathology apart from normal tension glaucoma.

On the basis of the foregoing criteria, 19 eyes from 13 subjects were selected.

An indication of the degree of glaucomatous damage in the selected group is given by the following summary measures of the mean deviation (MD) of the initial field in each test series: the mean of the MDs was −6.81 (SD 6.01) dB, the median was −5.43 dB, and the range was −22.40 dB to +1.07 dB.

TESTING STRATEGY

All tests were performed on a standard Humphrey automated perimeter. The full threshold 30-2 program with standard 4-2 dB staircase strategy was used throughout. Tests

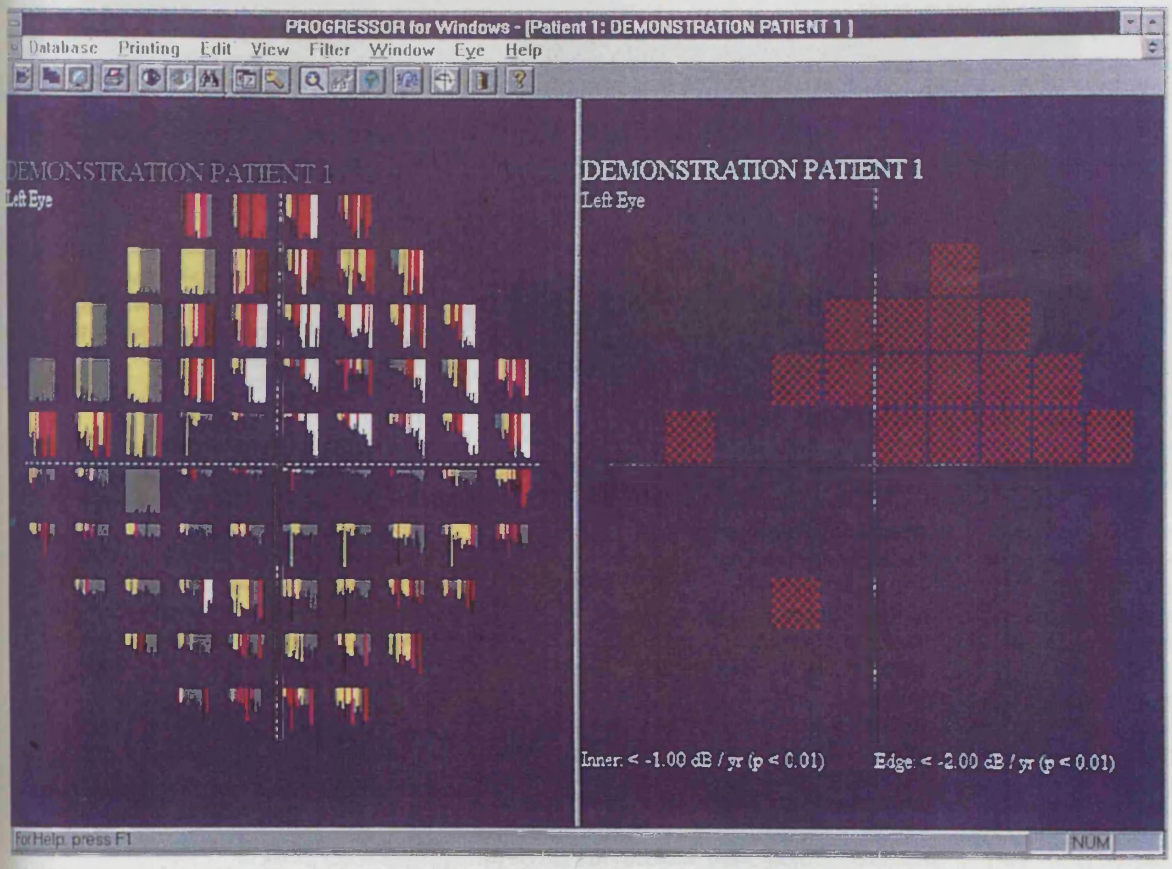


Figure 2 (A) Example of the cumulative graphical output of PROGRESSOR *for Windows for the left eye of an untreated normal tension glaucoma patient with visual field progression. The left pane shows the bar graph output. The right pane shows the locations which satisfy the progression criteria.*

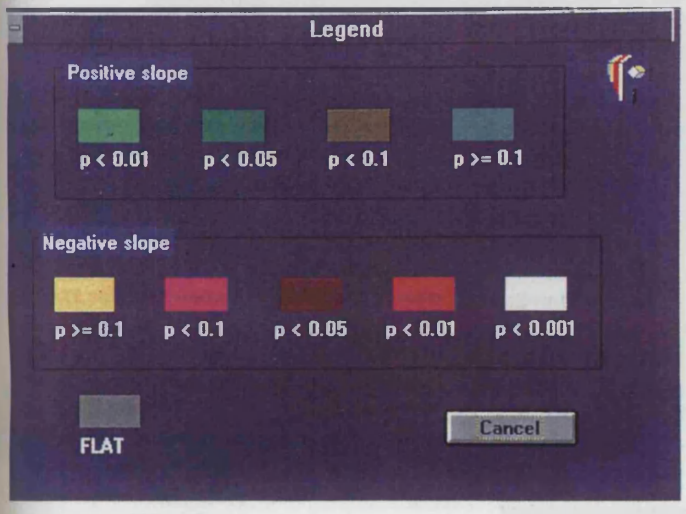


Figure 2 (B) Legend for PROGRESSOR *for Windows.*

were performed at intervals of approximately 4 months.

PROGRESSION CRITERIA

Glaucoma change probability: STATPAC 2

A field series was regarded as progressing if the glaucoma change probability analysis of STATPAC 2 marked any test location as showing significant deterioration ($p < 0.05$) from the baseline relative to the Humphrey normal database on three consecutive occasions. The requirement for a location to be repeatedly ascribed a high probability of change over a series of consecutive fields has been recommended by the originators of STATPAC 2[20] and has been used in previous studies.[21 25] The specific criterion of three consecutive fields is currently in use in the Early Manifest Glaucoma Trial in Malmö, and the importance of confirmatory testing to establish progression relative to a baseline has been upheld in the ongoing collaborative normal tension glaucoma study.[28]

PROGRESSOR

A field series was regarded as progressing if PROGRESSOR identified at least one test location with a negative slope of 1 dB per year or worse associated with $p < 0.05$ for a two tailed t test of the slope against zero (that is, the null hypothesis of no deterioration). The slope criterion of 1 dB per year represents a rate of sensitivity loss approximately 10 times greater than the normal age related decline.[29] Edge points are known to be more subject to fluctuation[29] so a stricter slope criterion of 2 dB per year (also with $p < 0.05$) was introduced for them. These slope criteria, in combination with a less stringent slope significance criterion of $p < 0.1$, have been demonstrated to compare closely with the Humphrey STATPAC 2 glaucoma change probability analysis.[21 25]

DETECTION TIME

The detection time for a given field series for a given algorithm was defined as the time interval between the initial field in the series

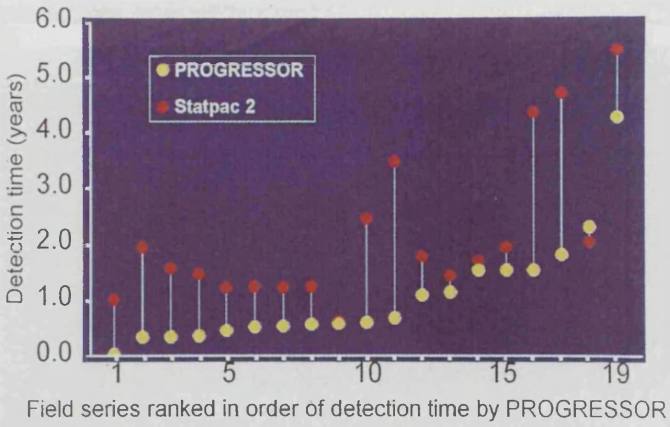


Figure 3 Drop line graph of detection times for each field series for PROGRESSOR and STATPAC 2. The field series are ranked in order of detection time by PROGRESSOR.

and the field when the progression criteria for that algorithm (see above) were first satisfied.

RELIABILITY OF EARLY DETECTION

There is at present no 'gold standard' for the identification of visual field progression in glaucoma. Thus, it is very difficult to assess whether a given technique is detecting 'true' progression. Some authors have used clinical impression against which to compare the performance of various algorithms,[17] but this has been shown to be a largely subjective measure.[9] Others have avoided the problem entirely by not attempting to estimate the reliability of their techniques.[11]

In the field series selected for this study, all three methods of assessing change (clinical expertise, PROGRESSOR, and STATPAC 2) agreed that all of the series showed progression. However, this study was designed to examine early detection of progression by the algorithms, so it was important to assess whether progression detected at a given field in a series was sustained in the rest of the series: it is likely that if progression is diagnosed in one field but not in subsequent fields the diagnosis is incorrect.

The reliability of early detection of progression by PROGRESSOR and STATPAC 2 was assessed by examining whether at least one of the locations initially detected as progressing was still progressing in the final field of the series. For STATPAC 2, the criterion of repeatability over three fields was not applied for the final field of the series in this analysis: STATPAC 2 was judged as reliable if at least one location initially diagnosed as progressing was labelled with a black triangle in the single final field.

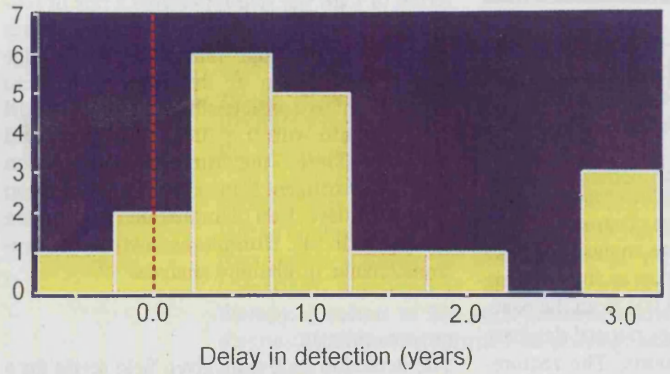


Figure 4 Histogram of delay in detection associated with STATPAC 2. The mean delay is 1.085 (SD 0.936) years.

Table 1 Detection times for PROGRESSOR and STATPAC 2

	Detection time (years)	
	Mean	*SD*
PROGRESSOR	1.077	0.985
STATPAC 2	2.161	1.357

($p<0.01$, Wilcoxon signed rank Z test)

STATISTICAL ANALYSIS

For each field series, detection time (see above) was calculated for each algorithm. Detection times for PROGRESSOR were compared with their correlates for STATPAC 2 using a non-parametric test for paired data from two related samples (Wilcoxon signed rank Z test).[30]

Statistical analysis was performed using the software package SPSS for Windows version 6.0.

Results

DETECTION TIMES

All 19 field series satisfied the progression criteria for both PROGRESSOR and STATPAC 2. The mean detection time for PROGRESSOR was 1.077 years with a standard deviation of 0.985 years. STATPAC 2 gave a mean detection time of 2.161 years with a standard deviation of 1.357 years. These findings are displayed in Table 1, and the individual differences between the algorithms for each field series are shown as a drop line graph in Figure 3. STATPAC 2 has consistently later detection times than PROGRESSOR: there is only one field series in which the detection time for PROGRESSOR is greater than that for STATPAC 2.

DELAY

Delay in detection was calculated as the difference between the detection time for STATPAC 2 and that for PROGRESSOR. The mean delay in detection associated with STATPAC 2 was 1.085 years and the standard deviation was 0.936 years. As can be seen in Figure 4, the great majority of values of delay are greater than zero. In other words, PROGRESSOR detects progression sooner than STATPAC 2. This is borne out by the results of the Wilcoxon signed rank Z test ($p < 0.01$, Table 2).

Table 2 Comparison of detection times between PROGRESSOR and STATPAC 2 (Wilcoxon signed rank Z test)

	Cases	*Mean rank*
Negative ranks	1	3.00
Positive ranks	18	10.39
Ties	0	
Total	19	

$Z = -5.1816$, $p = 0.0002$.
Negative ranks occur when STATPAC 2 detects progression in a given field series earlier than PROGRESSOR.
Positive ranks occur when PROGRESSOR detects progression in a given field series earlier than STATPAC 2.
Ties occur when both algorithms detect progression in a given field series at the same time.

RELIABILITY

For PROGRESSOR, at least one of the locations initially detected as progressing was still progressing in the final field of the series in 78.9% (15 out of 19) of the field series. For STATPAC 2, the corresponding figure was 68.4% (13 out of 19).

Discussion

Previous work has shown that PROGRESSOR agrees closely with STATPAC 2 in terms of which test locations are classified as progressing and which are classified as stable.[21 25] Thus, it seems paradoxical that PROGRESSOR should be found to be superior to STATPAC 2 in the detection of progression. This apparent contradiction may be explained by considering the differences in methodology between this and the previous studies.

In the absence of a gold standard for glaucomatous visual field progression, the previous authors chose STATPAC 2 as an arbitrary gold standard against which to measure PROGRESSOR. STATPAC 2 was used to identify test locations which had 'unequivocally deteriorated' in the last three fields of series consisting of 16 fields each.[25] The ability of PROGRESSOR to discriminate between these locations and the other locations (defined as stable by STATPAC 2) was then examined using the kappa statistic[31] to assess the level of agreement with STATPAC 2. A kappa coefficient of $\kappa = 0.62$ (SE = 0.04) was obtained, which represents good agreement for this length of follow up.[32] However, as the authors note,[21 25] this analysis is inherently unable to detect any potential superiority of PROGRESSOR in detecting progression: it would be interpreted as a lack of specificity.

There are also theoretical explanations for the finding that PROGRESSOR detects progression sooner than STATPAC 2. The two algorithms attempt to diagnose progression in very different ways. STATPAC 2 compares each field under analysis with the baseline: information in fields after the baseline but before the field under analysis is ignored. STATPAC 2 is thus an *event* type of analysis and would be particularly sensitive to catastrophic, stepwise change. In contrast, PROGRESSOR uses all the fields up to and including the field under analysis: it is thus a *trend* type of analysis and would be more sensitive to gradual, sustained change. The pattern of pointwise change in progressive glaucoma has been investigated: although sudden, stepwise change does occur[33] the mode of progression of the visual field of untreated normal tension glaucoma patients is best described by a pointwise linear analysis.[24] Furthermore, STATPAC 2 is only able to classify locations if their level of loss can be determined relative to a normal database: previous work comparing PROGRESSOR with STATPAC 2 has excluded 26% of test locations from analysis because they cannot be classified by STATPAC 2.[25]

The normal tension glaucoma study group has reported that an event type of analysis can be used to diagnose visual field progression in a timely, sensitive and specific way.[28] In this analysis, patients are followed every 3 months with visual field tests. If the criteria for suspected progression are met, the patient returns within 1–4 weeks for one or two confirmatory tests. If progression is confirmed in this way on two consecutive series of visits 3 months apart, true progression is diagnosed. Thus, the minimum time to diagnosis is 3 months and 1 week, which is earlier than for either algorithm in this study. The application of this testing protocol would enable earlier diagnosis of progression with STATPAC 2. However, PROGRESSOR would also be likely to detect change earlier in this case, since more frequent testing results in the earlier achievement of a significant slope: formal investigation would be required to determine whether PROGRESSOR or STATPAC 2 would benefit more from this strategy of confirmatory testing. Furthermore, this study protocol entails a high frequency of testing which would have profound resource implications if it were implemented on a routine basis, since it may involve three tests every 3 months.

In summary, this study was not designed to assess the sensitivity or specificity of PROGRESSOR or STATPAC 2 relative to an arbitrary gold standard (since this has been examined elsewhere[17 25]), but rather to investigate the performance of the two algorithms in reliably detecting early change in a group of patients known to be deteriorating unequivocally on clinical grounds. The fact that PROGRESSOR consistently detected progression earlier than STATPAC 2, and detected it more reliably, suggests that PROGRESSOR is a useful software tool for the analysis of visual field progression in glaucoma.

Presented in part at the Association for Research in Vision and Ophthalmology annual meeting, 25 April 1996.

Supported in part by grants from the International Glaucoma Association and the Medical Research Council.

The Institute of Ophthalmology retains the intellectual property rights to the PROGRESSOR software used in this study.

1 Boeglin RJ, Caprioli J, Zulauf M. Long-term fluctuation of the visual field in glaucoma. *Am J Ophthalmol* 1992; **113**:396–400.
2 Flammer J, Drance SM, Zulauf M. Differential light threshold. Short- and long-term fluctuation in patients with glaucoma, normal controls, and patients with suspected glaucoma. *Arch Ophthalmol* 1984;**102**:704–6.
3 Zulauf M, Caprioli J. What constitutes progression of visual field defects? *Sem Ophthalmol* 1992;7:130–46.
4 Hoskins HD, Jensvold N, Zaretsky M, Hetherington J. Rate of progression of discrete areas of the visual field. In: Heijl-A, ed. *Perimetry update 1988/1989*. Amsterdam: Kugler & Ghedini, 1989:173–6.
5 Mikelberg FS. Do computerised visual fields and automated optic disc analysis assist in the choice of therapy in glaucoma? *Eye* 1992;**6(Pt 1)**:47–9.
6 Hitchings K. Psychophysical testing in glaucoma. *Br J Ophthalmol* 1993;77:471–2.
7 Johnson CA. Modern developments in clinical perimetry. *Curr Opin Ophthalmol* 1993;4:7–13.
8 Fitzke FW, McNaught AI. The diagnosis of visual field progression in glaucoma. *Curr Opin Ophthalmol* 1994;5:110–5.
9 Werner EB, Bishop KI, Koelle J, Douglas GR, LeBlanc RP, Mills RP, *et al.* A comparison of experienced clinical observers and statistical tests in detection of progressive visual field loss in glaucoma using automated perimetry. *Arch Ophthalmol* 1988;**106**:619–23.
10 Weber J, Koll W, Krieglstein GK. Intraocular pressure and visual field decay in chronic glaucoma. *Ger J Ophthalmol* 1993;2:165–9.
11 Smith SD, Katz J, Quigley HA. Analysis of progressive change in automated visual fields in glaucoma. *Invest Ophthalmol Vis Sci* 1996;**37**:1419–28.
12 Wegner A, Ugi I, Hofman A. A long-term visual field evaluation of glaucoma patients treated topically with timolol or

carteolol. In: Mills RP, ed. *Perimetry update 1992/1993*. Amsterdam: Kugler & Ghedini, 1993:143–5.
13 Holmin C, Krakau CE. Regression analysis of the central visual field in chronic glaucoma cases. A follow-up study using automatic perimetry. *Acta Ophthalmol Copenh* 1982;**60**:267–74.
14 Wu D, Schwartz B, Nagin P. Trend analyses of automated visual fields. *Doc Ophthalmol Proc Ser* 1987;**49**:175–89.
15 O'Brien C, Schwartz B, Takamoto T, Wu DC. Intraocular pressure and the rate of visual field loss in chronic open-angle glaucoma. *Am J Ophthalmol* 1991;**111**:491–500.
16 Chauhan BC, Drance SM, Douglas GR. The use of visual field indices in detecting changes in the visual field in glaucoma. *Invest Ophthalmol Vis Sci* 1990;**31**:512–20.
17 Birch MK, Wishart PK, O'Donnell N. Determining progressive field loss. In: Mills RP, Wall M, eds. *Perimetry update 1994/1995*. Amsterdam: Kugler & Ghedini, 1995: 31–6.
18 Wild JM, Hussey MK, Flanagan JG, Trope GE. Pointwise topographical and longitudinal modeling of the visual field in glaucoma. *Invest Ophthalmol Vis Sci* 1993;**34**:1907–16.
19 O'Brien C, Schwartz B. The visual field in chronic open angle glaucoma: the rate of change in different regions of the field. *Eye* 1990;**4(Pt 4)**:557–62.
20 Heijl A, Lindgren G, Lindgren A, Olsson J, Asman P, Myers S, *et al*. Extended empirical statistical package for evaluation of single and multiple fields in glaucoma: Statpac 2. In: Mills RP, Heijl A, eds. *Perimetry update 1990/1991*. Amsterdam: Kugler & Ghedini, 1991:303–15.
21 Fitzke FW, Hitchings RA, Poinoosawmy D, McNaught AI, Crabb DP. Analysis of visual field progression in glaucoma. *Br J Ophthalmol* 1996;**80**:40–8.
22 Noureddin BN, Poinoosawmy D, Fitzke FW, Hitchings RA. Regression analysis of visual field progression in low tension glaucoma. *Br J Ophthalmol* 1991;**75**:493–5.
23 Poinoosawmy D, Wu J, Fitzke FW, Hitchings RA. Discrimination between progression and non-progression visual field loss in low tension glaucoma using MDT. In: Mills RP, ed. *Perimetry update 1992/1993*. Amsterdam: Kugler & Ghedini, 1993:109–14.
24 McNaught AI, Crabb DP, Fitzke FW, Hitchings RA. Modelling series of visual fields to detect progression in normal tension glaucoma. *Graefes Arch Clin Exp Ophthalmol* 1995; **233**:750–5.
25 McNaught AI, Crabb DP, Fitzke FW, Hitchings RA. Visual field progression: comparison of Humphrey Statpac 2 and pointwise linear regression analysis. *Graefes Arch Clin Exp Ophthalmol* 1996;**234**:411–8.
26 Werner EB, Adelson A, Krupin T. Effect of patient experience on the results of automated perimetry in clinically stable glaucoma patients. *Ophthalmology* 1988;**95**: 764–7.
27 Werner EB, Krupin T, Adelson A, Feitl ME. Effect of patient experience on the results of automated perimetry in glaucoma suspect patients. *Ophthalmology* 1990;**97**:44–8.
28 Schulzer M. Errors in the diagnosis of visual field progression in normal-tension glaucoma. *Ophthalmology* 1994;**101**:1589–94.
29 Heijl A, Lindgren G, Olsson J. Normal variability of static perimetric threshold values across the central visual field. *Arch Ophthalmol* 1987;**105**:1544–9.
30 Altman DG. *Practical statistics for medical research*. 1st ed. London: Chapman and Hall, 1991.
31 Fleiss JL. *Statistical methods for rates and proportions*. New York: John Wiley, 1981:212–25.
32 Landis JR, Koch GG. The measurement of observer agreement for categorical data. *Biometrics* 1977;**33**:159–74.
33 Mikelberg FS, Drance SM. The mode of progression of visual field defects in glaucoma. *Am J Ophthalmol* 1984;**98**: 443–5.

BMJ Publishing Group
BMA House, Tavistock Square, London WC1H 9JR. Tel. 0171 383 6305. Fax. 0171 383 6699

Printed in Great Britain by Meridian Print Centre Ltd. Derby. SJ/BJO/363/97/R