A.1. Nodal formulation

Even though the mathematical approach based on the solution of stress/strain relationships, equations of motion and equilibrium, and constitutive laws was set in Chapter 2; it is necessary to reduce the equations from an integral or differential form to a finite-difference form as illustrated in earlier examples on exponential growth. In other words the FLAC models will be discretised in a nodal formulation. Therefore finite-difference approximation is needed both in time and space to allow computer simulation in FLAC. The time is commonly discretised by subdivision in small constant calculation steps, whereas the use of discretised grids of nodes represents an approximation of the spatial continuum. A final important constraint imposed in FLAC is that the material derivation of the nodal velocities has to be reduced to zero to obtain a condition of static equilibrium. This is reasonable if it is imagined a natural example such as an earthquake. Earth motion in this case is due to accumulation of stresses due to plate tectonics abruptly released and converted into strain energy until an equilibrium state is reached. FLAC aims at equilibrium to simulate such natural tendency to minimise energy.

Reduction to a nodal form of classes of equations introduced (e.g. translational \( \xi_{ij} \) and rotational \( \omega_{ij} \) strain-rates) is firstly obtained reducing the laws at the tetrahedron scale. A deforming tetrahedron can be thought as a velocity field. In analogy to other conservative fields using the Gauss or divergence theorem:
it can be demonstrated that the flux escaping from an element of volume \((V)\) containing a source of constant intensity (e.g. a stationary fluid) is equivalent to the sum of the fluxes exiting from a closed surface surrounding the volume independent of the surface area interested by the flux. The symmetry of the problem lead then to a reduction to a finite summation of fluxes escaping from multiple surfaces of the FLAC tetrahedron, allowing a linear approximation of the nodal velocity as follows:

\[
v_{ij} = -\frac{1}{3V} \sum_{l=1}^{4} v_{il} n_{jl} S^{(l)} \tag{A.2}
\]

The \((A.2)\) is essentially a finite summation over the indexes \((l)\) that represent the four nodes of each tetrahedron. The superscript \((l)\) indicates a nodal property that is considering the sum of the contributions of all adjacent tetrahedrons acting on a certain node. \((S)\) is the area of each face.

Substitution into (2.66 – Chapter2) and (2.67) leads to a nodal formulation of translational and rotational strain rates \((\sigma_{ij}, \xi_{ij})\):
Appendix A

FLAC formulation

\[ \xi_{ij} = -\frac{1}{6V} \sum_{l=1}^{d} \left( v_{i}^{l} n_{j}^{(l)} + v_{j}^{l} n_{i}^{(l)} \right) S^{(l)} \]  
(A.3)

\[ \omega_{ij} = -\frac{1}{6V} \sum_{l=1}^{d} \left( v_{i}^{l} n_{j}^{(l)} - v_{j}^{l} n_{i}^{(l)} \right) S^{(l)} \]  
(A.4)

The (A.3) and (A.4) shows that nodal velocities can be used to represent forces applied to the whole tetrahedral grid and the relative strains and their rates resulting from the application of such stresses in time. In this regard seems to be clearer the meaning of a Lagrangian representation that look at the kinematic behaviour of individual nodes. In contrast, the required generalisation is brought by the nodal formulation of the laws of motion.

A.2. Theorem of the virtual work

The principle of virtual work is a convenient way to treat the laws of motion and it is used here to derive a nodal formulation for the Cauchy’s equations. It is based on the concept of kinetic energy in the form of thermodynamic work internal and external \((W_i, W_e)\) although unnecessary as the two represent the same measurement of strain in the considered model, the separation is however instructive. Taking the Gauss theorem as an analogy: the work performed on a generic surface would be equivalent to a flux of energy transferred from the environment into the system incrementing its energy.
although such energy rather than be stored is taken up by deformation and released internally producing work that corresponds to the rearrangement of the nodes of the tetrahedrons. The two are therefore the same quantity although described from a different frame of reference:

\[ W_i = \int \int v_{i,j} \, dV \quad (A.5) \]

\[ W_e = \int \int v_i n_j \, dS \quad (A.6) \]

Comparing with the (A.1) it can be defined:

\[ W_i = W_e \quad (A.7) \]

After these considerations the (2.68) can be expressed as representing \( W_e \) in the form:

\[ W_e = \sum_{n=1}^4 \delta v_i^n \, f_i^n + \int \delta v_i \, B_i \, dV \quad (A.8) \]

the two terms on the left side of (A.8) represent respectively the contribution of contact forces \( (f_i) \) and body forces here considered coupled with the material derivative of the velocity \( (B_i) \), in this case note that rather than calculating the work for a tetrahedron in term of virtual displacement, it is considered the nodal virtual velocity \( (\delta v_i^n) \) representing therefore a rate of external virtual work \( (W_e) \). On the other hand, the Cauchy’s relationships for the theorem of virtual work (A.7) have to be equated to the nodal formulation of the internal rate of displacement \( (W_i) \). It is in this case convenient to recall the (A.72) replacing strain with internal stresses as follows:
Such relationship can be further simplified in light of the symmetrical character of the stress tensor regrouping in the (A.10), where \( (T_i) \) is the stress vector:

\[
T_i' = \sigma_{ij} n_j^{(i)} S^{(i)}
\]  

(A.10)

\( W_i \) becomes:

\[
W_i = -\frac{1}{3} \sum_{n=1}^{4} \delta v_i^{(i)} T_i
\]  

(A.11)

With further rearranging and combining the (A.8) with the (A.11) a relationship that provides a value for the nodal force \( (f_i) \):

\[
- f_i^{(n)} = \frac{3}{3} T_i^{(n)} + \frac{1}{4} b_i V - m_i \left( \frac{dv_i}{dt} \right)^n
\]  

(A.12)

where the first term on the right side represents the internal work component whereas the other two terms account respectively for the body forces here considered function of the material derivative of the nodal velocity.
\[ b_i = \left( \frac{B_i}{\rho} + \frac{d^2 v_i}{dt^2} \right) \quad \text{(A.13)} \]

and the contact forces obtained considering fictitious nodal masses \((m)\) that are adjusted in FLAC to stabilise the solution. The (A.12) can be further generalised to the whole body considering the sum of the contributions of all nodes \((n_n)\) and relative inertial terms \((P_i)\):

\[ F_{i<\ell>} = M_{i<\ell>} \left( \frac{T_i}{3} + \frac{\rho b_i V_{i<\ell>}}{4} \right) + P_{i<\ell>} \quad \forall \ell = 1, \ldots n_n \quad \text{(A.14)} \]

To reach a static condition, after a perturbation has been imposed to the body, as seen from definition (A.7) the sum of all forces should progressively tend to zero. \( F_{i<\ell>} \to 0 \) during reorganisation resulting in stress accumulation and release of strain in time.

From the outlined relationships it is concluded that the spatial approximation is soundly based on the concepts of discretization and minimization of energy of the modelled body. Similarly to the form discretization is adopted in FLAC to reduce the continuity of time to finite intervals. FLAC computes nodal velocities on the basis of a central finite difference approximation in which each time step is half of the \(\Delta t\) used to compute the forces and displacements of the nodes. The approach can be demonstrated to give a second order approximation error of time derivatives the velocity.
Appendix A

FLAC formulation

approximation is given as an example; it can be expressed in finite difference form after integration of the second law of Newton as follows:

\[ v^<l>_{i}(t + \frac{\Delta t}{2}) = v^<l>_{i}(t - \frac{\Delta t}{2}) + \frac{\Delta t}{M^<l>} F^<l> \]

(A.15)

here the term \( M^<l> \) refers to the sum of all the contributions (\( m^<l> \)) of the tetrahedrons surrounding the node of interest \(< l >\).

A.3. The Mohr-Coulomb constitutive model and its relationship to the motion equations

The constitutive models work conceptually in a similar way, they try to differentiate between the plastic and the elastic component of the strain tensor. The distinction is possible thanks to the experimental mechanics models available. The simulated materials will be deforming in a variable manner for an applied stress accordingly to their relative properties (e.g. cohesion, the friction angle, the elastic/shear modulus, etc.). The objective is to describe the co-rotational stress increments in time (co-rotational is a convention in which the stress rates are measured by an observer rotating at the same angular velocity of the particle considered – here the suffix on top of the co-rotational stress (\( \sigma^<l> \)) is omitted because all stresses are co-rotational). Note that in a plastic deformation the co-rotational stress increments represent only the elastic part of
the deformation. To make a distinction between a plastic and elastic response of a material, a yield function is therefore assigned to the model (e.g. Fig. A.1):

\[ f(\sigma_{ij}) = 0 \]  \hspace{1cm} (A.16)

The (A.16) is equal to zero when the material fails for a certain combination of compressive or tensile stresses. For the Mohr-Coulomb material the (A.16) is represented by two functions (Fig. A.2) becoming:

\[ f^t = \sigma_3 - \sigma_{\phi} N_{\phi} + 2c \sqrt{N_{\phi}} \]  \hspace{1cm} (A.17)

which is the Mohr-Coulomb criteria, and also for a tensile failure the criteria becomes:

\[ f^t = \sigma_3 - \sigma' \]  \hspace{1cm} (A.18)

in such examples of yield functions it is outlined the dependency of the point of yield from mechanical terms other than confining stresses, i.e., the cohesion (c), a function of the friction angle \( (N_{\phi}) \) (A.17), and the tensile strength \( (\sigma') \) in the (A.18). Usually a generic Mohr-Coulomb material subject to an applied stress reacts firstly in an elastic manner until it yields and subsequently deform plastically.
Fig. A. 1 Example of yield surface function in stress space respectively for: (a) Drucker-Prager and Von Mises yield (conical) surfaces in principal stress space; (b) Mohr-Coulomb and Tresca failure envelopes (irregular hexagonal). Adapted from (Itasca, 2003).
Appendix A

Leonardo Feltrin

FLAC formulation

Fig. A. 2 Composite Mohr-Coulomb criterion with tension cut-off. represented in planar space \((\sigma_1, \sigma_3)\).

Compressive stress considered negative with \(\sigma_1 \leq \sigma_2 \leq \sigma_3\). The failure envelope \(f(\sigma_1, \sigma_3) = 0\) is defined within the interval A-B following Mohr-Coulomb type behaviour, whereas the curve in section B-C is characterised by a tensile failure criterion in which \(\sigma_3 \leq \frac{c}{\tan(\Phi)}\) with cohesion \(c\) and friction angle \(\Phi\). The yield function for shear/tensile failure \((f_s, f_t)\) are violated where \((\sigma_1, \sigma_3)\) fall above the lines corresponding to higher deviatoric stresses \((\sigma_1, \sigma_3)\) or when the tensile strength \((\sigma_t)\) is exceeded (Itasca, 2003).

The total strain increment \((\Delta \varepsilon_i)\) is therefore defined by:

\[
\Delta \varepsilon_{ij} = \Delta \varepsilon_{ij}^e + \Delta \varepsilon_{ij}^p
\]

(A.19)

both strain increments can be used to calculate the stress increments that generated them. Usually the elastic component responds to a linear law such as Hook’s law whereas non-linearity may result from the plastic component of strain that is represented by:

\[
\Delta \varepsilon_{ij} = \Delta \varepsilon_{ij}^e + \Delta \varepsilon_{ij}^p
\]
This relationship based on the flow rule outlines that the direction of the plastic strain increment vector is normal to the potential surfaces of \((g)\) with \(\lambda\) representing a constant. When the function \((g)\) is equal to the yield function \((f)\) the material is considered associative. The spatial association of the two tensorial fields (stress-rate and strain rate) is linear as in the elastic case. Non-associative materials however are the most common in natural examples as they commonly dilate during deformation (e.g. Vermeer and de Borst, 1984; Ord, 1991; McLellan, 2004). Non-associative materials have \(f\left(\sigma_{ij}\right) \neq g\left(\varepsilon_{ij}^p\right)\); therefore, there are conditions in which the material may deform plastically before reaching a yield condition based for instance on the Mohr-Coulomb criterion.

An iterative approach is used to guess a possible value for the state of stress after a certain stress increment has occurred; the new stress \((N)\) would be defined by:

\[
\sigma_{ij}^N = \sigma_{ij} + \Delta\sigma_{ij}
\]  

(A.21)
If it is considered that the stress increment, as seen, is reduced by the plastic component of the strain rate tensor then a possible way to make a definitive distinction between the two components of deformation is to formulate what is defined as an elastic guess in which it is assumed that the material is non-plastic or perfectly elastic and then using an iterative method the stress increment values are recomputed until they meet the yield function for the chosen constitutive relationship. It is then given the following:

\[
\sigma_{ij}^{N} = \sigma_{ij}^{I} - \lambda \sigma_{ij} \left( \frac{\partial g}{\partial \sigma_{ij}} \right)
\]  

(A.22)

in which the new stress state is obtained, as discussed, subtracting the linear term \(S_i(.)\) from the elastic guess \(\sigma_{ij}^{I}\). Once the co-rotational stress increment is defined it can be used in conjunction with the equations of motion to derive more realistic nodal velocities in the FLAC models.

**A.4. Effect of fluid flow in a deforming porous media**

One of the advantages of FLAC is the availability of a fluid flow module that can be coupled to the deformation module. This is represented by a set of equations with a general organisation similar to the mechanical module. In this regard the equations of motion are replaced by the Biot and Darcy’s laws that defines the variation...
Appendix A

FLAC formulation

of fluid flow, a quantity that can be characterised with a fluid intensity field such as the one described by fluid discharge vectors \( (q_i) \) in 3D, or in alternative a scalar function that is represented by the spatial variation of pore pressure \( (p) \) to understand the distribution of equipotentials in the fluid flow field. This latter representation is particularly useful as seen in chapter 4 and chapter 5 to evaluate possible fluid pathways during deformation. In this context FLAC has the advantage that it can capture the effect of deformation on fluid flow or alternatively the effect of fluid pressure in dissipating the confining pressure (Terzaghi, 1945). In particular, FLAC makes use of the Biot coefficient \( (\alpha) \) to couple mechanical calculations with transient fluid flow. In addition to this the software allows also to consider the effect on temperature on the volume variation of modelled materials using a linear thermal expansion coefficient \( (\alpha_t) \), and in undrained conditions adopting a thermal coefficient \( (\beta) \).

Here it is given a brief description of the fluid-flow equations and their interaction with the constitutive functions introduced in the mechanical part of this review. The linear quasi-static theory of Biot is used in this context to couple deformation and diffusion processes in Darcy’s type flow. It is considered a porous material as a media that can transfer and store fluids. However, such properties can vary accordingly to a series of parameters that can be defined internal if they depend upon the system subjected to flow (e.g. permeability) or external if they are controlled by the environment. An example is given providing the equation that serves to calculate the coefficient of diffusivity \( (c) \):
Appendix A

FLAC formulation

\[ c = \frac{k}{1 + \frac{\alpha^2}{\alpha_1}} \tag{A.23} \]

where \( k \) is the permeability, \( M \) represents the Biot modulus, \( \alpha \) is the Biot coefficient, \( \alpha_1 \) is a function of the bulk and shear modulus. An external parameter could be for instance exemplified by temperature variations or the control of hydraulic gradient. To mathematically constrain these components a mass-balance equation is formulated below. This relationship opportunely combined with the fluid constitutive relation provides a differential equation in terms of pore pressure that can be solvable for certain conditions providing a means for the fluid flow. For small deformations the equation is given by:

\[ -q_{r,i} + q_v = \frac{\partial \zeta}{\partial t} \tag{A.24} \]

the (A.23) can be interpreted similarly to the divergence theorem presented above in the (A.1) because it uses the same principle of conservation of mass. In other words the intensity of the flow is a measure of the divergence of the flow field, which is here in (A.24) represented by the partial derivative in time on the right side. On the left side of the same equation \( q_{r,i} \) is the spatial variation of the discharging vectors (outflow) whereas \( q_v \) represents the volumetric fluid source intensity. The shape of potential surfaces is then controlled by fluid sources, leaks or other morphological boundaries.
that directly defines the fluid flow directions within a confined reservoir. However, other more intrinsic parameters need to be considered, as seen, to fully describe the problem of modelling fluid migration within saturated porous media. According to this the fluid flow intensity gradients are also governed directly by the storage capacity of a reservoir. Volumetric variation is the prominent controlling factor, but this in turn is function of strain distributions, temperature and pore pressure variations. This lead to a different formulation of the (A.24) that incorporates these variables:

\[
\frac{\partial \zeta}{\partial t} = \left( \frac{1}{M} \frac{\partial p}{\partial t} \right) + \alpha \frac{\partial \varepsilon}{\partial t} - \beta \frac{\partial T}{\partial t} \tag{A.25}
\]

Here pore pressure gradients \((p)\), strain-rate \((\varepsilon)\) and temperature gradients \((T)\) are all linearly related to the variation of fluid content per unit volume of porous material \((\zeta)\). Combining the (A.24) with the (A.25) and rearranging it follows:

\[- q_{i,i} + q_{i} = \frac{1}{M} \left( \frac{\partial p}{\partial t} \right) \tag{A.26}\]

where

\[q_{i} = q_{i} - \alpha \frac{\partial \varepsilon}{\partial t} + \beta \frac{\partial T}{\partial t} \tag{A.27}\]
the (A.26) is used to calculate the pore pressure \( p \) variation in time knowing the parameters condensed in (A.27) and also calculating \( q_i \) using the Darcy’s law that for an homogeneous isotropic solid is given by:

\[
q_i = - \frac{k}{\eta} p_i. \tag{A.28}
\]

This equation, by definition, relates the discharge vector intensity (flow velocity) to the intrinsic permeability coefficient \( k \), and the gradient of the pore pressure \( p \) in space.

If the velocity of the fluids percolating a porous media is controlled by the spatial variability of the scalar field of pore pressure distributions; in turn, the pore pressure itself may vary accordingly to the (A.26) and (A.27), depending on the parameters (e.g. material properties). Mechanical constitutive laws in this regard are accordingly modified to account for such variations. Recalling the general form of the constitutive equations (2.69) an updated incremental expression of the co-rotational stress increment is given by:

\[
\Delta \left[ \tilde{\sigma}_{ij} \right] + \alpha \Delta p \tilde{\delta}_{ij} = H_{ij}^t \left( \sigma_{ij} + \Delta \varepsilon_{ij} - \Delta \varepsilon_{ij}^f \right) \tag{A.29}
\]

the thermal-mechanical coupling is defined as:
fluid flow correction terms are present on both sides of the constitutive equation (A.29). In particular, the additional term on the left side outlines the influence of the pore pressure increment in reducing (similarly to the effective stress concept) the co-rotational stress component. The stress correction term comprehend also the Biot coefficient \((\alpha)\) and the Kronecker Delta (unitary tensor \((\delta_{ij})\), used to convert to tensorial form the scalar field of pore pressure). On the right side of (A.29) as also in (A.30) the strain increment is influenced by the effect of temperature increments that may for instance increase the volume of the porous media expanding its matrix. The (A.29) also shows the interconnection existing among the mechanical and fluid flow modules. Correction terms in the constitutive equation, for example, could lower the co-rotational stress increment causing a reduction of nodal velocities.

Outlined equations are solved in FLAC using a finite different approach. The numerical scheme rests on a nodal formulation of the mass balance equation. This approach is not reviewed here because it is equivalent to the mechanical formulation presented above, which leads to the nodal form of the Newton’s law. This simply involves the substitution of pore pressure, specific discharge vector and pore pressure gradient for velocity vector, stress and the strain-rate tensors, respectively. Nonetheless the solution of ordinary differential equation is obtained using two distinct discretization models in time (implicit/explicit formulation) in the fluid flow module.
Appendix A

FLAC formulation

However, the scope of this section on FLAC was primarily focused on the understanding of the concept of discretization and the general organisation of the software, for a deeper understanding the reader is referred to the FLAC documentation (Itasca, 2003).
B.1. Weights of Evidence formulation

The Weights of Evidence method and relative algorithms are briefly discussed here (adapted from Bonham-Carter, 1994 and Carranza, 2004).

The Weights of Evidence method is a way to express the likelihood of finding a mineral deposit based on a certain representation of knowledge, which is a conditional probability function $P(x)$. The knowledge itself is mathematically expressed as Weights of Evidence, numerical scores derived from a measure of the spatial association between known deposits ($D$) and a considered pattern ($B_n$). The spatial association represents a conditional probability expressed as follows:

$$P(D \mid B_n) = \frac{P(D \cap B_n)}{P(B_n)}$$  \hspace{1cm} (B.1)

where $P(D \mid B_n)$ is the conditional probability of finding $D$ overlapping with a pattern $B_n$ which is proportional to the area of $D$ and $B_n$ and inversely proportional to $N(T)$ as $P(B_n) = N(B_n)/N(T)$. It is also possible to express the conditional probability of finding a pattern ($B_n$) overlapping with ($D$). $P(D \cap B_n)$ is equal to $P(B_n \cap D)$; however the conditional probability is different because the same area of intersection is divided by diverse prior probabilities as follows:
Appendix B

Weights of Evidence

\[ P \left( B_n | D \right) = P \left( B_n \cap D \right) / P \left( D \right) \]  \hspace{1cm} (B.2)

Equations (10 and 11) can be combined to obtain the following representation of posterior probability (equivalent to conditional probability):

\[ P \left( D | B_n \right) = P \left( B_n | D \right) P \left( D \right) / P \left( B_n \right) \]  \hspace{1cm} (B.3)

A similar expression can be derived as a measure of correlation of \( D \) with the absence of a pattern \( (B_n) \) from known deposits \( (D) \):

\[ P \left( D | B_n^c \right) = P \left( B_n^c | D \right) P \left( D \right) / P \left( B_n^c \right) \]  \hspace{1cm} (B.4)

These conditional probabilities are expressed for convenience as odds and also are converted in a logarithmic form to obtain the Weights of Evidence values. Odds can be defined as:

\[ O = \frac{P}{1 - P} = \frac{P}{1 - P} = P \left( D | B_n \right) / P \left( D | B_n \right) \]  \hspace{1cm} (B.5)

From equations (B.3, B.4) conditional probabilities can be substituted to obtain, for instance, the conditional odd of deposits given the presence of \( B_n \):
In logarithmic form, both the presence or absence of \( B_n \) are expressed for convenience as follows, where \( W_n \) is the weight depending on the pixel considered, as this may fall within either an area where \( B_n \) occurs or is missing:

\[
\text{postlogit} (D | B_n) = \text{prilogit} (D) + W_n^+ \\
\text{postlogit} (D | \overline{B_n}) = \text{prilogit} (D) + W_n^-
\]  

(B.7) \hspace{1cm} (B.8)

Rearranging these equations (B.7, B.8) the Weights of Evidence can be calculated as follows:

\[
W_n^+ = \ln \left[ \frac{P (B_n | D)}{P (B_n | \overline{D})} \right] \\
W_n^- = \ln \left[ \frac{P (\overline{B_n} | D)}{P (\overline{B_n} | \overline{D})} \right]
\]  

(B.9) \hspace{1cm} (B.10)

Equations (B.7, B.8, B.9, B.10) can be combined in a single algorithm, representing the Bayes rule of combination, where the \( k \) coefficient defines if the pattern is present or absent as a function of the pixel \( (n) \) considered:
Appendix B

Weights of Evidence

\[
\text{postlogit} \left( D | B_n^{k(n)} \right) = \text{prilogit} (D) + \sum_{n=1}^{k} W_n^{k(n)}
\]  

(B.11)

A posterior probability value that represents the sum of all the evidential layers is computed from equation (B.11) back-calculating its value as follows:

\[
P_{Post} = \left( \frac{\sum_{n=1}^{k} w_n^{k(n)} + \ln O (D)}{1 + \left( \sum_{n=1}^{k} w_n^{k(n)} + \ln O (D) \right)} \right)
\]  

(B.12)

An estimate of the error involved with the calculation of the Weights of Evidence can be expressed using the asymptotic assumption of Bishop et al. (1975). These represent the variances of the Weights of Evidence as a function of the total area of study expressed as total number of pixels (unit cells) \( N(T) \). The mathematical relationships are also function of the area of known occurrences \( N(D) \). The equations are:

\[
s^2(W_n^*) = \left( \frac{1}{N(B \cap D)} \right) + \left( \frac{1}{N(B \cap \overline{D})} \right)
\]  

(B.13)

\[
s^2(W_n^*) = \left( \frac{1}{N(\overline{B} \cap D)} \right) + \left( \frac{1}{N(\overline{B} \cap \overline{D})} \right)
\]  

(B.14)
Appendix B  Weights of Evidence

These values were used to compute either the studentised value of the Contrast \( C \) and also for the calculation of the error involved in the estimation of posterior probabilities \( P_{\text{post}} \). This is derived from a multiple combination of layers that form multiple classes \( k \) of pixel column combinations:

\[
s^2(P_{\text{post}}) = \left[ \frac{1}{N(D)} + \sum_{n=1}^{n} \left( s^2(W_{n}^{k(n)}) \right) \right] \cdot P_{\text{post}}^2 \tag{B.15}
\]

The available estimate of the variance of the posterior probability is used to compute the error involved in the estimation of the number of predicted mineral deposits in a certain region:

\[
s(N(D)_{\text{pred}}) = \sqrt{\sum_{n=1}^{n} \left[ (N(k))^2 \times s^2(P_{\text{post}}) \right]} \tag{B.16}
\]

This latter equation can be used to increase the statistical robustness of the NOT test (Chapter 3).
C.1. Wofe Modeler

Compiled in VB 2005 (Express Edition)

Software used to compute Bayesiam probability in WofE (Chapter 3)

Region " Software developed by Feltrin Leonardo - James Cook University"

Imports System.IO
Imports System.Text
Imports System
Imports System.Drawing
Imports System.Drawing.Printing
Imports System.Collections
Imports System.ComponentModel
Imports System.Windows.Forms
Imports System.Data
Imports Rebuild_wofe.Form5

' These instructions are initialising "libraries" to open and write txt files- see line and drawing
Public Class Form1
    Inherits System.Windows.Forms.Form

    ' Variable used in case of multi selection of bool geology
    Dim BoolM() As String

    ' Variables inserted to allow bitmap functionalities
    Private bmpImage As System.Drawing.Bitmap
    Private bayesmap As System.Drawing.Bitmap
    Private curZoom As Double = 1.0
    Private curRect As Rectangle
    Private originalSize As New System.Drawing.Size(0, 0)
    Private mouseDownPt As New Point(0, 0)
    Private mouseUpPt As New Point(0, 0)
    Private zoomMode As Boolean = False
    Private imagesize As System.Drawing.Bitmap
    Private myfile() As Byte
    Private cellsz As Double

    ' Variables that are used by missing evidence functionalities
    Dim d() As Byte
    Dim PkD() As Single
    Dim PDx() As Single
    ' Dim Px() As Single
    Dim SigmaSQm() As Double
    Dim s2Pkm() As Double
    Dim s2Pk() As Double
    Dim s2Pkf As Double
    Dim s2Pkf_miss As Double
Dim Dep As Integer
Dim SigmaSQ_missingev() As Double

Public Property Image_size()
    Get
        Return imagesize
    End Get
    Set(ByVal value)
        imagesize = value
    End Set
End Property

'variables for wofe
Dim FileNames() As String
Private Shared bmpsizepub As Integer
Private Shared AD_w, AI_w, AG_w, AT_w As Double 'shared variables expressing areas as cell numbers; they work in all sub routines
Private Shared SumArray() As Byte
Private Shared SumArray2() As Byte
Private Shared WplusARR(), WminusARR(), ContrastARR(),
    Stud_CwARR(), sqvar_WplusARR(), sqvar_WminusARR(), _
    stdv_ContrastARR(), stdv_WplusARR(), stdv_WminusARR() As Double
Private Shared ar() As String

'it is left to test 2bmp under process might be some inconsistency of overlap of data as
'exporting we lose precision
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim cellsz = TextBox9.Text
    If TextBox9.Text = Nothing Then
        MsgBox("Please insert the required cell size in square kilometres")
        Exit Sub
    End If

    'we delete all files with old data
    If RadioButton2.Checked = True Then
        Try
            'code...
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
    Else
        'code...
    End If
End Sub
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\Wplus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\Wminus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\Contrast.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\sqvar_Wplus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\stdv_Wplus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\sqvar_Wminus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\stdv_Wminus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\stdv_Contrast.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\Stud_Cw.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\ArrayCuMAI.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\ArrayCuMAG.txt")

Catch : MsgBox("no files to delete")
End Try

Dim myarray2() As Byte

'we have to import the string header of multilayer info (multiple binari files 1 0 maps)
'to do that we need a list with the names of the files we need to load and then we use for to create

'Dim header As String = Nothing
Dim a As String
a = TextBox1.Text

'header = My.Computer.FileSystem.ReadAllText(a)
'MsgBox(header & "Check and delete any final space character")
'MsgBox("Check and delete any final space character")
`Dim myarray() As String = BoolM

' Dim c As Integer
' For c = 0 To myarray.Length - 1
'    myarray(c) = "c:\wofe\bmp_geol\" & myarray(c)
' Next

' now we need a for loop to load the files if multiple layers are chosen
Dim s As Integer
For s = 0 To myarray.Length - 1
    myarray2 = My.Computer.FileSystem.ReadAllText(myarray(s).ToString)

    ' this convert one file.bmp to a myarray
    ' Dim binary(myarray2.Length - 1) As Byte

    ' we jump on the cleaning algo each time to clean up the bitmap

    Dim r As Integer
    ' Here we need some code that cleans up the bitmaps for us, we declare 2 new myarrays that will be locally storing
    ' the original BMP in binary format. Then we get the data out of them and feed SumArray and SumArray2
    Dim OriginalAD() As Byte = My.Computer.FileSystem.ReadAllText(TextBox8.Text) ' we use the deposit layer

    Dim OriginalAG(myarray2.Length - 1) As Byte
    OriginalAG = myarray2

    Try
        r = (ComboBox3.Text * ComboBox4.Text) ' the number of cells needed to get the number of good data pixels
    Catch : MsgBox("provide rows and columns numbers") Exit Sub
    End Try

    ' MsgBox("Numeber of Cells " & r)
    ' Dim po As New Integer
    ' Dim ps As Integer
    ' Dim pq As Integer
    Dim SumArray(r - 1) As Byte
    Dim SumArray2(r - 1) As Byte

    ' tronca arrays con remove command
Appendix C

Source code

Array.Reverse(OriginalAD)
Array.Resize(OriginalAD, r)
Array.Reverse(OriginalAD)

SumArray = OriginalAD

Array.Reverse(OriginalAG)
Array.Resize(OriginalAG, r)
Array.Reverse(OriginalAG)

SumArray2 = OriginalAG

'For cycles to load arrays (SumArray...
'For po = (OriginalAD.Length) - r To OriginalAD.Length - 1
'    ps = (po - ((OriginalAD.Length) - r))
'    SumArray.SetValue(OriginalAD(po), ps) ' SumArray
'Next

'For po = (OriginalAG.Length) - r To OriginalAG.Length - 1
'    pq = (po - ((OriginalAG.Length) - r))
'    SumArray2.SetValue(OriginalAG(po), pq) ' SumArray2
'Next

'MsgBox(SumArray2.GetValue(r - 1))
'MsgBox(OriginalAG(OriginalAG.Length - 1))

Dim b As Integer = 0
Dim Sum As Double = 0
Dim Sum2 As Double = 0
'Dim SumCounter As Integer
'Dim Counter(255) As Integer
'Dim CounterTwo(255) As Integer

'These following are two constants AD and AT Area of deposits and Total Study area

'algebraic sum of array (AD) we get just 1s not 0 counted, therefore the area of deposits cells
For b = 0 To SumArray.Length - 1
    Sum = Sum + SumArray(b) '-1 is inserted as the array starts from 0
Next b
Label9.Text = Sum.ToString() 'Output AD
Dim AD As Double = Sum
'assign shared variable for w calc
AD_w = Sum * cellsz

Leonardo Feltrin
Appendix C

---

```
' here we add the conversion to deposit number
Label35.Text = Sum * cellsz & "pixels"

' Sum of all cells to get total area expressed as cell units (AT)
Dim AT As Double = SumArray.Length
Label8.Text = AT.ToString() ' Output AT
AT_w = r * cellsz

'This part is inserted to calculate the weights

Dim a2 As Integer = SumArray2.Length ' array limit for cycle
Dim b2 As Integer

'The following code has to be run in case of binary 0 1 classes of files
' algebraic sum of array (AG) we get just 1's not 0 counted, therefore the area of geology or other things
For b2 = 0 To SumArray2.Length - 1
        Sum2 = Sum2 + SumArray2(b2) ' -1 is inserted as the array starts from 0
Next
Label10.Text = Sum2.ToString() ' Output AG

Dim AG As Double = Sum2
AG_w = AG * cellsz ' see above

' we want to add each element of an array with an element of a second array with same index
' firstly we declare the 2 arrays, first geo second deposit
' If geo is multiclass this code cannot handle it therefore it has to run only in case of 0,1 image
' To make it work with the multiclass it has to cycle with AG variation, we need a new array (AI)
' we want also store the result in a third array
declare the lenght as Array have specific lenght

' we declare the boolean array
Dim third(SumArray.Length - 1) As Byte
Dim q As Integer
' here we create the for loop, which makes a multiplication of pixels (= to boolean intersection)
For q = 0 To SumArray.Length - 1
        third(q) = (SumArray(q) * SumArray2(q))
Next
```
Dim AI As Double
For q = 0 To SumArray.Length - 1
    AI = AI + third(q)
Next

My.Computer.FileSystem.WriteAllBytes("c:\wofe\pmh_final.rst", third, False)
Labell1.Text = AI.ToString 'Output AI
AI_w = AI * cellsz ' see above

'Here we get the weight using previous variables

WEIGHT MODULE

Dim Wplus As Double
Dim Wminus As Double
Dim Contrast As Double
Dim Stud_Cw As Double
Dim sqvar_Wplus As Double
Dim sqvar_Wminus As Double
Dim stdv_Contrast As Double
Dim stdv_Wplus As Double
Dim stdv_Wminus As Double

Try

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Array CumAI.txt", AI_w.ToString & " ", True)
Catch ex As Exception

My.Computer.FileSystem.CreateDirectory("c:\wofe\weights_calc_output\")

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Array CumAI.txt", AI_w.ToString & " ", True)

End Try

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Array CumAG.txt", AG_w.ToString & " ", True)

'Equation for W+
Wplus = Math.Log(((AI_w / AD_w) * ((AT_w - AD_w) / (AG_w - AI_w))))
Labell9.Text = Wplus.ToString

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Wplus .txt", Wplus.ToString & " ", True)
'Equation for W-

$$W_{\text{minus}} = \log\left(\frac{AD_{\text{w}} - AI_{\text{w}}}{AD_{\text{w}}} \cdot \frac{AT_{\text{w}} - AD_{\text{w}} - AG_{\text{w}} + AI_{\text{w}}}{AT_{\text{w}} - AD_{\text{w}}}\right)$$

`Label20.Text = Wminus.ToString`

`My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Wminu\s.txt", Wminus.ToString & " ", True)`

'Equation for Cw

$$\text{Contrast} = W_{\text{plus}} - W_{\text{minus}}$$

`Label21.Text = Contrast.ToString`

`My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Contrast.txt", Contrast.ToString & " ", True)`

'Equation for v(W+)

$$\text{sqvar}_{W_{\text{plus}}} = \frac{1}{AI_{\text{w}}} + \frac{1}{AG_{\text{w}} - AI_{\text{w}}}$$

`My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\sqvar_Wplus.txt", sqvar_Wplus.ToString & " ", True)`

'Equation for s(W+)

$$\text{stdv}_{W_{\text{plus}}} = \sqrt{\text{sqvar}_{W_{\text{plus}}}}$$

`Label23.Text = stdv_Wplus.ToString`

`My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\stdv_Wplus.txt", stdv_Wplus.ToString & " ", True)`

'Equation for v(W-)

$$\text{sqvar}_{W_{\text{minus}}} = \frac{1}{AD_{\text{w}} - AI_{\text{w}}} + \frac{1}{AT_{\text{w}} - AD_{\text{w}} + AI_{\text{w}}}$$

`My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\sqvar_Wminus.txt", sqvar_Wminus.ToString & " ", True)`

'Equation for s(W-)

$$\text{stdv}_{W_{\text{minus}}} = \sqrt{\text{sqvar}_{W_{\text{minus}}}}$$

`Label24.Text = stdv_Wminus.ToString`

`My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\stdv_Wminus.txt", stdv_Wminus.ToString & " ", True)`
' Equation for \( s(Cw) \)
stdv_Contrast = Math.Sqrt(sqvar_Wplus + sqvar_Wminus)

'25
Label25.Text = stdv_Contrast.ToString

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\stdv_Contrast.txt", stdv_Contrast.ToString & " ", True)

' Equation for the studentized value of Cw '22
Dim Stud_CwARR(255) As Double
Stud_Cw = Contrast / stdv_Contrast
Label22.Text = Stud_Cw.ToString

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Stud_Cw.txt", Stud_Cw.ToString & " ", True)

' output lines relative to each value we get a number of array

AI = Nothing
AG = Nothing
AI_w = Nothing
AG_w = Nothing
Wplus = Nothing
Wminus = Nothing
Contrast = Nothing
Stud_Cw = Nothing
sqvar_Wplus = Nothing
sqvar_Wminus = Nothing
stdv_Contrast = Nothing
stdv_Wplus = Nothing
stdv_Wminus = Nothing
'WplusARR(255) = Nothing
'WminusARR(255) = Nothing
'sqvar_WplusARR(255) = Nothing
'stdv_WplusARR(255) = Nothing
'sqvar_WminusARR(255) = Nothing
'stdv_WminusARR(255) = Nothing
'stdv_ContrastARR(255) = Nothing
'ContrastARR(255) = Nothing
SumArray = Nothing
SumArray2 = Nothing

Next
MsgBox("Well done leo!")
Exit Sub

Else
Try

My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\Wplus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\Wminus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\Contrast.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\sqvar_Wplus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\stdv_Wplus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\sqvar_Wminus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\stdv_Wminus.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\stdv_Ccontrast.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\Stud_Cw.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\ArrayCumAI.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\AIhisto.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\AGhisto.txt")
My.Computer.FileSystem.DeleteFile("c:\wofe\weights_calc_output\ArrayCumAG.txt")

Catch : MsgBox("no files to delete")

End Try

'Here we need some code that cleans up the bitmaps for us, we declare 2 new arrays that will be locally storing the original BMP in binary format. Then we get the data out of them and feed SumArray and SumArray2
Dim OriginalAD() as Byte = My.Computer.FileSystem.ReadAllBytes(TextBox8.Text) 'we use the deposit layer
Dim OriginalAG() As Byte = My.Computer.FileSystem.ReadAllBytes(TextBox7.Text) 'we use the geo

Dim r As Integer = (ComboBox3.Text * ComboBox4.Text) 'MsgBox("Numeber of Cells " & r)
Dim po As Integer
Dim ps As Integer
Dim pq As Integer
Dim SumArray(r - 1) As Byte
Dim SumArray2(r - 1) As Byte

'For cycles to load arrays (SumArray...
For po = (OriginalAD.Length) - r To OriginalAD.Length - 1
  ps = (po - ((OriginalAD.Length) - r))
  SumArray.SetValue(OriginalAD(po), ps)
Next
po = 0
For po = (OriginalAG.Length) - r To OriginalAG.Length - 1
  pq = (po - ((OriginalAG.Length) - r))
  SumArray2.SetValue(OriginalAG(po), pq)
Next

'MsgBox(SumArray2.GetValue(r - 1))
'MsgBox(OriginalAG(OriginalAG.Length - 1))

Dim b As Integer = 0
Dim Sum As Double = 0
Dim Sum2 As Double = 0
Dim SumCounter As Integer
Dim Counter(255) As Integer
Dim CounterTwo(255) As Integer

'These following are two constants AD and AT Area of deposits and Total Study area

' algebraic sum of array (AD) we get just 1s not 0 counted, therefore the area of deposits cells
Dim a As Integer = SumArray.Length - 1
For b = 1 To a
  Sum = Sum + SumArray(b - 1) '-1 is inserted as the array starts from 0
Next b
Label9.Text = Sum.ToString() 'Output AD
Dim AD As Double = Sum
'assign shared variable for w calc
AD_w = Sum * cellsz

'here we add the conversion to deposit number
Label35.Text = Sum * cellsz & " km^2"

'Sum of all cells to get total area expressed as cell units (AT)
Dim AT As Double = SumArray.Length
Label8.Text = AT.ToString() 'Output AT
AT_w = r * cellsz ' input by user

'This code creates an histogram array (counter) used to
detect the type of evidence image used.
'Works analysing the third element of Counter() if the sum
of all the values except the first two of Counter() is equal to
'0, then we deal with a binary 0 1 file.
'Compute the sum of all elements of Counter() except the
first two (0,1) why?
'Counter for a 0 1 type file will give a final sum = to 0.
This is true as we
'we are using cleaned arrays (SumArray see up stripping)
'Counter refers to AG array
For a = 0 To SumArray2.Length - 1 'histo for AG class
    Counter(SumArray2(a)) += 1
    MsgBox(Counter(a))
Next a
Try
    For b = 0 To Counter.Length - 1
        My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\AGhis
to.txt", Counter(b).ToString & " ", True)
    Next
Catch ex As Exception
    My.Computer.FileSystem.CreateDirectory("c:\wofe\weights_calc_output\")
    For b = 0 To Counter.Length - 1
        My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\AGHis
to.txt", Counter(b).ToString & " ", True)
    Next
End Try

'Checking the type of data....
For a = 2 To Counter.Length - 1
    SumCounter = SumCounter + Counter(a)
Next a

'we need a second histo for AI not sure if I have to
insert it below
'now we write on a file txt or compile a database.....

'Dim objStreamWriter As StreamWriter

''Open the file. The software runs and save each time on hist, therefore we might need to solve the issue of ''copying above the same file (problem can be solved with user input or using a delete option instruction) ''
objStreamWriter = New StreamWriter("c:\wofe\histoAG.txt", True, _ '' Encoding.Unicode)

''Write out the numbers on the same line.
'Dim c As Integer
'For c = 0 To 255 '' objStreamWriter.Write(Counter(c) & " ") 'Next c
'objStreamWriter.Close()

'If the sum is 0 then the image is binary therefore use the following code to calculate AG
If SumCounter = 0 Then
    MsgBox("Binary file [0,1]")
    Dim a2 As Integer = SumArray2.Length 'array limit for cycle
    Dim b2 As Integer
    'The following code has to be run in case of binary 0 1 classes of files
    'algebraic sum of array (AG) we get just 1's not 0 counted, therefore the area of geology or other things
    For b2 = 1 To a2
        Sum2 = Sum2 + SumArray2(b2 - 1) ' -1 is inserted as the array starts from 0
    Next b2
    Label10.Text = Sum2.ToString() 'Output AG
    Dim AG As Double = Sum2
    AG_w = AG * cellsz ' see above

    'we want to add each element of an array with an element of a second array with same index
    'firstly we declare the 2 arrays, first geo second deposit
    'If geo is multiclass this code cannot handle it therefore it has to run only in case of 0,1 image
'To make it work with the multiclass it has to cycle with AG variation, we need a new array \{AI\}

'we want also store the result in a third array

'define the length as Array have specific length
b = SumArray.Length
Dim third(b) As Byte

'here we create the for cycle, which makes a multiplication of pixels (= to boolean intersection)
For a = 1 To b
    'third is an array that stores the value AI for each pixel
    third.SetValue(SumArray(a - 1) * SumArray2(a - 1), a - 1)
Next a

Dim AI As Double = 0
For a = 1 To b
    AI = AI + third(a - 1)
Next a

'My.Computer.FileSystem.WriteAllBytes("c:\wofe\pmh_final.rst", third, False)
Label11.Text = AI.ToString 'Output AI
AI_w = AI * cellsz 'see above

'Here we get the weight using previous variables

WEIGHT MODULE

Dim Wplus As Double
Dim Wminus As Double
Dim Contrast As Double
Dim Stud_Cw As Double
Dim sqvar_Wplus As Double
Dim sqvar_Wminus As Double
Dim stdv_Contrast As Double
Dim stdv_Wplus As Double
Dim stdv_Wminus As Double

'Equation for W+
Wplus = Math.Log(((AI_w / AD_w) * ((AT_w - AD_w) / (AG_w - AI_w))))
Label19.Text = Wplus.ToString

'Equation for W-
Wminus = Math.Log(((AD_w - AI_w) / AD_w) * ((AT_w - AD_w - AG_w + AI_w))))
Label20.Text = Wminus.ToString

'Equation for Cw
Contrast = Wplus - Wminus '21
Label21.Text = Contrast.ToString

'Equation for v(W+)
sqvar_Wplus = (1 / AI_w) + (1 / (AG_w - AI_w))
'Equation for $s(W^+)$

\[
\text{stdv}_W^+ = \sqrt{\text{sqvar}_W^+}
\]

Label23.Text = stdv_Wplus.ToString

'Equation for $v(W^-)$

\[
\text{sqvar}_W^- = \left( \frac{1}{(\text{AD}_w - \text{AI}_w)} + \frac{1}{(\text{AT}_w - \text{AG}_w - \text{AD}_w + \text{AI}_w)} \right)
\]

stdv_Wminus = Math.Sqrt(sqvar_Wminus)

Label24.Text = stdv_Wminus.ToString

'Equation for $s(W^-)$

\[
\text{stdv}_W^- = \sqrt{\text{sqvar}_W^-}
\]

Label25.Text = stdv_Wminus.ToString

'Equation for $s(C_w)$

\[
\text{stdv}_\text{Contrast} = \sqrt{\text{sqvar}_W^+ + \text{sqvar}_W^-}
\]

Label25.Text = stdv_Contrast.ToString

'Equation for the studentized value of $C_w$

\[
\text{Stud}_C_w = \frac{\text{Contrast}}{\text{stdv}_\text{Contrast}}
\]

Label22.Text = Stud_Cw.ToString

AI = Nothing
AG = Nothing
AI_w = Nothing
AG_w = Nothing
Wplus = Nothing
Wminus = Nothing
Contrast = Nothing
Stud_Cw = Nothing
sqvar_Wplus = Nothing
sqvar_Wminus = Nothing
stdv_Contrast = Nothing
stdv_Wplus = Nothing
stdv_Wminus = Nothing
WplusARR(255) = Nothing
WminusARR(255) = Nothing
sqvar_WplusARR(255) = Nothing
stdv_WplusARR(255) = Nothing
sqvar_WminusARR(255) = Nothing
stdv_WminusARR(255) = Nothing
stdv_ContrastARR(255) = Nothing
ContrastARR(255) = Nothing
SumArray = Nothing
SumArray2 = Nothing

'-----------------------------------------------------

El
'we want to add each element of an array with an
element of a second array with same index
'firstly we declare the 2 arrays, first geo second
deposit
'If geo is multiclass this code cannot handle it
therefore it has to run only in case of 0,1 image
'To make it work with the multiclass it has to cycle
with AG variation, we need a new array (AI)
'we want also store the result in a third array
'Counter() is a list of all AG in order from 0 to 255,
we need to feed the AG to produce the AI array
'Looping through Counter we pick each value and
perform the computation of AI and W

'this code works for a single AG array calculation
'define the lenght as Array have specific lenght
Dim third(SumArray.Length - 1) As Byte

'here we create the for cycle, which makes a
multiplication of pixels ( = to boolean intersection)
For a = 1 To SumArray.Length
'third is an array that stores the value AI for
each pixel
    third.SetValue(SumArray(a - 1) * SumArray2(a - 1), a - 1) 'SumArray2 is a single array of AG
Next a
'We get third() which represents the array
intersection

'Now we run HISTO on third()
'CounterTwo refers to AD array
For a = 0 To third.Length - 1 'histo for AG class
    CounterTwo(third(a)) += 1
Next a
For b = 0 To Counter.Length - 1
My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\AIhisto.txt", CounterTwo(b).ToString & " ", True)
Next
'MsgBox(Counter(a))

'in both cases AG and AI has to be estimated as
cumulative proportions, therefore we need to progressively sum up
'the different elements of the two arrays to obtain 2
new arrays with cumulative growth of areas
Dim n As Integer
'Dim s As Integer
'Dim p As Integer
Dim ArrayCumAG(Counter.Length - 1) As Double
ArrayCumAG(0) = Counter(0)
For n = 1 To Counter.Length - 1
    ArrayCumAG(n) = ((Counter(n)) + ArrayCumAG(n - 1))
End For

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Array CumAG.txt", ArrayCumAG(n).ToString & " ", True)

' This code requires too much resources
' For n = 0 To SumArray2.Length - 1
'    For s = 0 To n
'        p = p + SumArray2(s)
'    Next s
'    ArrayCumAG(n) = p
'    p = 0
' ArrayCum is the array that contains cumulative AI proportions
Next n

' we correct the first value for (0) derived by non intersection during bool
CounterTwo(0) = (CounterTwo(0) - (r - Sum))
' AI cumulative prop... Changed we keep same values as CounterTwo

Dim ArrayCumAI(CounterTwo.Length - 1) As Double
' For b = 0 To 255
'    ArrayCumAI(b) = CounterTwo(b)
' Next

' Cumulative of AI increasing with distance analysis
ArrayCumAI(0) = CounterTwo(0)
For n = 1 To CounterTwo.Length - 1
    ArrayCumAI(n) = ((CounterTwo(n)) + ArrayCumAI(n - 1))
End For

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Array CumAI.txt", ArrayCumAI(n).ToString & " ", True)

' After evaluation of multiclass layers we obtain two final arrays (ArrayCumAI(); ArrayCumAG())

' In this case the module WEIGHT is integrated and modified to run with two arrays of AG and AI values that run in parallel
'Here we get the weight using previous variables

**WEIGHT MODULE**

```vbnet
Dim Wplus As Double
Dim Wminus As Double
Dim Contrast As Double
Dim Stud_Cw As Double
Dim sqvar_Wplus As Double
Dim sqvar_Wminus As Double
Dim stdv_Contrast As Double
Dim stdv_Wplus As Double
Dim stdv_Wminus As Double
Dim WplusARR(255) As Double
Dim WminusARR(255) As Double
Dim sqvar_WplusARR(255) As Double
Dim stdv_WplusARR(255) As Double
Dim sqvar_WminusARR(255) As Double
Dim stdv_WminusARR(255) As Double
Dim stdv_ContrastARR(255) As Double
Dim ContrastARR(255) As Double
' delete all files in data folder

For a = 0 To Counter.Length - 1
    ' Equation for W+
    AI_w = ArrayCumAI(a) * cellsz  ' we convert in area of deposits km2
    AG_w = ArrayCumAG(a) * cellsz  ' we convert the area in km^2 from cells
    Wplus = Math.Log(((AI_w / AD_w) * ((AT_w - AD_w) / (AG_w - AI_w))))
    Label19.Text = Wplus.ToString
    WplusARR.SetValue(Wplus, a)
    My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Wplus.txt", WplusARR(a).ToString & " ", True)

    ' Equation for W-
    Wminus = Math.Log(((AD_w - AI_w) / AD_w) * ((AT_w - AD_w) / (AT_w - AD_w - AG_w + AI_w)))
    Label20.Text = Wminus.ToString
    WminusARR.SetValue(Wminus, a)
    My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Wminus.txt", WminusARR(a).ToString & " ", True)
```

---

Leonardo Feltrin
'Equation for Cw

Contrast = WplusARR(a) - WminusARR(a) '21
Label21.Text = Contrast.ToString
ContrastARR.SetValue(Contrast, a)

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Contrast.txt", ContrastARR(a).ToString & " ", True)

'Equation for v(W+)

sqvar_Wplus = (1 / AI_w) + (1 / (AG_w - AI_w))
sqvar_WplusARR.SetValue(sqvar_Wplus, a)

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\sqvar_Wplus.txt", sqvar_WplusARR(a).ToString & " ", True)

'Equation for s(W+)

stdv_Wplus = Math.Sqrt(sqvar_WplusARR(a)) '23
Label23.Text = stdv_Wplus.ToString
stdv_WplusARR.SetValue(stdv_Wplus, a)

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\stdv_Wplus.txt", stdv_WplusARR(a).ToString & " ", True)

'Equation for v(W-)

sqvar_Wminus = (1 / (AD_w - AI_w)) + (1 / (AT_w - AG_w - AD_w + AI_w))
sqvar_WminusARR.SetValue(sqvar_Wminus, a)

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\sqvar_Wminus.txt", sqvar_WminusARR(a).ToString & " ", True)

'Equation for s(W-)

stdv_Wminus = Math.Sqrt(sqvar_WminusARR(a)) '24
Label24.Text = stdv_Wminus.ToString
stdv_WminusARR.SetValue(stdv_Wminus, a)

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\stdv_Wminus.txt", stdv_WminusARR(a).ToString & " ", True)

'Equation for s(Cw)

stdv_Contrast = Math.Sqrt(sqvar_WplusARR(a) + sqvar_WminusARR(a)) '25
Label25.Text = stdv_Contrast.ToString
stdv_ContrastARR.SetValue(stdv_Contrast, a)

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\stdv_Contrast.txt", stdv_ContrastARR(a).ToString & " ", True)
'Equation for the studentized value of Cw '22
Dim Stud_CwARR(255) As Double
Stud_Cw = ContrastARR(a) / stdv_ContrastARR(a)
Label22.Text = Stud_Cw.ToString
Stud_CwARR.SetValue(Stud_Cw, a)

My.Computer.FileSystem.WriteAllText("c:\wofe\weights_calc_output\Stud_Cw.txt", Stud_CwARR(a).ToString & " ", True)

Next a

'output lines relative to each value we get a number of arrays

'AI = Nothing
'AG = Nothing
AI_w = Nothing
AG_w = Nothing
Wplus = Nothing
Wminus = Nothing
Contrast = Nothing
Stud_Cw = Nothing
sqvar_Wplus = Nothing
sqvar_Wminus = Nothing
stdv_Counter = Nothing
stdv_Wplus = Nothing
stdv_Wminus = Nothing
WplusARR(255) = Nothing
WminusARR(255) = Nothing
sqvar_WplusARR(255) = Nothing
sqvar_WminusARR(255) = Nothing
stdv_WplusARR(255) = Nothing
stdv_WminusARR(255) = Nothing
ContrastARR(255) = Nothing
SumArray = Nothing
SumArray2 = Nothing

MsgBox("Weighting Completed!")

End If

'Make sure the 2 images have same size and resolution before u start intersecting and weighting
End If
Private Sub Button2_Click(sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    My.Forms.Form2.Show()
End Sub

Private Sub Button3_Click(sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    'Dim myStream As Stream
    Dim selectFileDialog1 As New OpenFileDialog()
    selectFileDialog1.InitialDirectory = "c:\"
    selectFileDialog1.Filter = "bmp files (*.bmp)|*.bmp|All files (*.*)|*.*"
    selectFileDialog1.FilterIndex = 2
    selectFileDialog1.RestoreDirectory = True
    If selectFileDialog1.ShowDialog() = DialogResult.OK Then
        TextBox7.Text = selectFileDialog1.FileName
        'myStream = selectFileDialog1.OpenFile()
        'If Not (myStream Is Nothing) Then
        '    ' Insert code to read the stream here.
        '    myStream.Close()
        'End If
    End If
End Sub

Private Sub Button4_Click(sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    'Dim myStream As Stream
    Dim selectFileDialog1 As New OpenFileDialog()
    selectFileDialog1.InitialDirectory = "c:\"
    selectFileDialog1.Filter = "bmp files (*.bmp)|*.bmp|All files (*.*)|*.*"
    selectFileDialog1.FilterIndex = 2
    selectFileDialog1.RestoreDirectory = True
    If selectFileDialog1.ShowDialog() = DialogResult.OK Then
        TextBox8.Text = selectFileDialog1.FileName
        'myStream = selectFileDialog1.OpenFile()
        'If Not (myStream Is Nothing) Then
        '    ' Insert code to read the stream here.
        '    myStream.Close()
'End If
End If
End Sub

Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
'multiselect option
'Dim myStream As Stream
Dim selectFileDialog1 As New OpenFileDialog()

selectFileDialog1.InitialDirectory = "c:\"
selectFileDialog1.Filter = "bitmap file (*.bmp)\*.*.bmp"
selectFileDialog1.FilterIndex = 1
selectFileDialog1.RestoreDirectory = True
selectFileDialog1.Multiselect = True

If selectFileDialog1.ShowDialog() = DialogResult.OK Then
'ar() can be used to process the data for calculation of weights
Dim ar() As String = selectFileDialog1.FileNames
Dim a As String = Join(ar, "")
TextBox1.Text = a

' myStream = selectFileDialog1.OpenFile()
' If Not (myStream Is Nothing) Then
' ' Insert code to read the stream here.
' 'myStream.Close()
' 'End If
'BoolM = ar
End If
End Sub

Public Function BinaryConv(ByVal bound As Byte, ByVal myarray() As Byte, ByVal bmpSize As Integer, ByVal FileName As String) As Double

' we need to strip the header and store it somewhere
Dim headerLength As Integer
headerLength = (myarray.Length) - bmpSize
'Dim ab As Integer
Dim stripArray() As Byte
stripArray = myarray
Array.Resize(stripArray, headerLength)
Array.Reverse(myarray)
Array.Resize(myarray, bmpSize)
Array.Reverse(myarray)

'For ab = 0 To headerLength
'    stripArray(ab) = myarray(ab)
'Next
'My.Computer.FileSystem.WriteAllBytes("c:\wofe\hearer.bmp", stripArray, False)
'we need to perform a bit conversion to get a 0,1 binary type
Dim a As Integer
Dim value As Byte
For a = 0 To myarray.Length - 1
    value = myarray(a)
    If value >= bound Then
        myarray(a) = 0
    Else
        If value < bound Then
            myarray(a) = 255
        End If
    End If
Next

'problem is that final...(2) give an array with length 3, with
3 elements not 2, therefore use -1
Dim finalarray((stripArray.Length + myarray.Length) - 1) As Byte
Dim ac As Integer
For ac = 0 To stripArray.Length - 1
    finalarray(ac) = stripArray(ac)
Next
For ac = stripArray.Length To finalarray.Length - 1
    finalarray(ac) = myarray(ac - stripArray.Length)
Next
'we need to create a folder where we can store the evidential
boolean layers
My.Computer.FileSystem.WriteAllBytes("c:\wofe\image.bmp", finalarray, False)
Form4.Show()
End Function

Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button6.Click
    'Dim myStream As Stream
    Dim selectFileDialog1 As New OpenFileDialog()
    selectFileDialog1.InitialDirectory = "c:\wofe\"
    selectFileDialog1.Filter = "bmp files (*.bmp)|*.bmp|All files (*.*)|*.*"
    selectFileDialog1.FilterIndex = 2
    selectFileDialog1.RestoreDirectory = True
    If selectFileDialog1.ShowDialog() = DialogResult.OK Then
TextBox2.Text = selectFileDialog1.FileName

' myStream = selectFileDialog1.OpenFile()
' If Not (myStream Is Nothing) Then
'   Insert code to read the stream here.
' myStream.Close()
' End If
End If
End Sub

Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button7.Click

Dim myarray() As Byte = My.Computer.FileSystem.ReadAllBytes(TextBox2.Text)
Dim bound As Integer
Dim bmpsize As Integer
bound = Byte.Parse(TextBox3.Text)
bmpsize = Integer.Parse(TextBox4.Text)
bmpsizepub = bmpsize
Me.BinaryConv(bound, myarray, bmpsize, TextBox2.Text)

End Sub

Private Sub Button9_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button9.Click

Dim a As Integer
' Dim b As Integer
Dim FileName(Bayes_Data_Source_ModelDataGridView.Rows.Count) As String
Dim Wplus(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim Wminus(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single

For a = 0 To Bayes_Data_Source_ModelDataGridView.Rows.Count
    Try
        FileName(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(6).Value
        Wplus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(1).Value
        Wminus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(2).Value
    Catch : Exit For
End Sub
End Try
Next
Bayes(FileName, Wplus, Wminus)

End Sub

Public Function Bayes(ByVal FileName() As String, ByVal Wplus() As Single, ByVal Wminus() As Single) As Byte

' I need to firstly select the bitmaps created with convert,
then store the strings in the first column
'manually we type in the weights and ask for that with a msg
box
'all is set and ready for the calculation
'we need to select each row and use the loop to convert the
pixels to an array of weights
'we need to sum up the images to the prior probability image
Dim Form5Inst As New Form5
Dim Dep As Integer
Dep = Form5Inst.AD(TextBox8.Text)

imagesize = Bitmap.FromFile("imagesize.bmp")

Dim postlogit((imagesize.Height * imagesize.Width) - 1) As Single
Dim weightedArray((imagesize.Height * imagesize.Width) - 1) As Single
Dim rescale((imagesize.Height * imagesize.Width) - 1) As Single
Dim ab As Integer
For ab = 0 To FileName.Length - 1

' imagesize = Bitmap.FromFile(FileName(ab))
If FileName(ab) = Nothing Then
    Exit For
End If
Dim evidence As Bitmap = Bitmap.FromFile(FileName(ab).ToString)

Dim myfile((evidence.Width * evidence.Height) - 1) As Byte
Dim color As System.Drawing.Color
Dim county As Integer
Dim countx As Integer
Dim s As Integer
For county = 0 To evidence.Height - 1
For countx = 0 To evidence.Width - 1
color = evidence.GetPixel(countx, county)
If CInt(color.B) <> 0 And CInt(color.G) <> 0 And CInt(color.R) <> 0 Then
    myfile(s) = 1
Else
    myfile(s) = 0
End If
s = s + 1
Next
Next
s = 0
Dim value As Byte
Dim ard As Integer
If FileName(ab) Is Nothing Then
    Exit For
Else

    'here we convert the image in evidence layer using the weights
    For ard = 0 To myfile.Length - 1
        value = myfile(ard)
        If value = 0 Then
            weightedArray(ard) = Wminus(ab)
        Else
            weightedArray(ard) = Wplus(ab)
        End If
    Next
End If

' modified to exclude textbox6 input

' now we need to add the evidence layer created to the prior probability
' prilogit should be calculated as the ratio D/T the conversion in ODLS and
' ln function gives prilogit value
' I need the deposit layer and the tot number of pixels

' Dim Dep_conv As Single = Dep * 0.000269
Dim priprob As Single = (Dep / (imagesize.Height * imagesize.Width))
Dim priOdd As Single = priprob / (1 - priprob)
Dim con As Single
Appendix C

Leonardo Feltrin

con = Math.Log(priOdd)

Dim ac As Integer
Dim priorlogit((imagesize.Height * imagesize.Width) - 1)
As Single 'five

' con = Single.Parse(TextBox6.Text)
For ac = 0 To ((imagesize.Height * imagesize.Width) - 1)
priorlogit(ac) = con
Next

For ac = 0 To postlogit.Length - 1
If ab = 0 Then
    postlogit(ac) = postlogit(ac) + priorlogit(ac) +
weightedArray(ac)
Else
    postlogit(ac) = postlogit(ac) + weightedArray(ac)
End If
Next

' now we need to rescale the double in a way that we can
generate a byte array

Dim ad As Integer
Dim aq As Integer
Dim postodds(postlogit.Length - 1) As Single
Dim postprob(postlogit.Length - 1) As Single
Dim postprobByte(postlogit.Length - 1) As Byte
For aq = 0 To postlogit.Length - 1
    ' we convert to probability

    postodds(aq) = Math.Exp(postlogit(aq))
    postprob(aq) = postodds(aq) / (1 + postodds(aq))

Next
Array.Copy(postprob, rescale, postlogit.Length)
Array.Sort(rescale)

Dim min As Single = rescale(0)
Dim max As Single = rescale(rescale.Length - 1)
My.Computer.FileSystem.WriteAllText("c:\wofe\scale.txt",
min.ToString, False)
My.Computer.FileSystem.WriteAllText("c:\wofe\scale.txt",
max.ToString, True)
Appendix C Source code

For ad = 0 To postprob.Length - 1
    ' we shift or translate the scale to get a minimum of 0
    ' min must become firstly equal to 1 so if min*x=1
    Try
        Dim alfa As Single = (1 / min)
        postprob(ad) = postprob(ad) * alfa
        postprob(ad) = (postprob(ad) - (min * alfa)) / ((max * alfa) - (min * alfa))
        postprobByte(ad) = CByte((postprob(ad)))
        postprob(ad) = (postprob(ad) + (min * alfa)) / alfa
    Catch ex As Exception
        MsgBox("something wrong with the input?")
        Exit Function
    End Try
    Next

My.Computer.FileSystem.WriteAllBytes("c:\wofe\postprobebyte.bin", postprobByte, False)
    MsgBox("Pprob Completed!")
End Function

Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button8.Click
    'Dim myStream As Stream
    Dim selectFileDialog1 As OpenFileDialog()
    selectFileDialog1.InitialDirectory = "c:\"
    selectFileDialog1.Filter = "bitmap file (*.bmp)| *.bmp"
    selectFileDialog1.FilterIndex = 1
    selectFileDialog1.RestoreDirectory = True
    selectFileDialog1.Multiselect = True
    If selectFileDialog1.ShowDialog() = DialogResult.OK Then
        bindingNavigatorAddNewItem.PerformClick()
        Dim ar() As String = selectFileDialog1.FileNames
        ' Dim a As String = Join(ar, " ")
Dim a As Integer
Dim b As String
Dim c As String
FileNames = ar
For a = 0 To ar.Length - 1
    b = ar(a)
    c = ar(a)
    b = b.Substring(b.LastIndexOf("\") + 1)
    Try
        Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(0).Value = b
        Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(6).Value = c
        Catch ex As Exception
            bindingNavigatorAddNewItem.PerformClick()
        End Try
    Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(0).Value = b
    Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(6).Value = c
Next
' bindingNavigatorDeleteItem.PerformClick()
' MsgBox("Insert relative weights")
' myStream = selectFileDialog1.OpenFile()
' If Not (myStream Is Nothing) Then
'    Insert code to read the stream here.
' myStream.Close()
' End If
End If
End Sub

' Public Function ArrayMax(ByVal math() As Double) As Double
'
    Dim a As Integer
    Dim av As Double
    Dim b As Double
    Dim c As Integer
    Dim d As Integer
    ' For a = 0 To math.Length - 1
    '    b = b + math(a)
    Next
    ' av = b / math.Length
Appendix C

Source code

' Dim newmath(math.Length - 1) As Double
' For c = 0 To math.Length - 1
'   If math(c) >= av Then
'       newmath(c) = math(c)
'   End If
' Next
'
'End Function

Public Function ArrayAn(ByVal ar() As Single) As Integer
    Dim a As Integer
    Dim c As Integer
    Dim counter(10) As Single
    c = 0
    Try
        For a = 0 To ar.Length - 1
            If ar(a) <> ar(a + 1) Then
                counter(c) = ar(a)
                counter(c + 1) = ar(a + 1)
                c = c + 2
            End If
        Next
    Catch : Exit Try
    End Try
    Dim d As Integer
    For d = 0 To counter.Length - 1
    Next
'    Return MsgBox("ok")
End Function

Private Sub Button10_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button10.Click
    If TextBox9.Text = "" Then
        MsgBox("Please insert a valid cell size value")
        Exit Sub
    End If
    My.Forms.Form5.Show()
End Sub

Public Function imagesizefunc() As Integer
    Dim a As Integer
Appendix C

Source code

imagesize = Bitmap.FromFile("imagesize.bmp")

a = (imagesize.Width * imagesize.Height)
Return a
End Function

Private Sub Button13_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button13.Click
' start new ot ot
Try
cell = Me.TextBox9.Text
Catch ex As Exception
MsgBox("Please provide cell size")
Exit Sub
End Try

d = Nothing
PkD = Nothing
PDx = Nothing
SigmaSQm = Nothing
s2Pk = Nothing
s2Pkf = Nothing
s2Pkf_miss = Nothing
Dep = Nothing
bmpsizepub = Nothing
AD_w = Nothing
AI_w = Nothing
AG_w = Nothing
AT_w = Nothing 'shared variables expressing areas as cell numbers; they work in all sub routines

Dim a As Integer
'Dim b As Integer
Dim FileName(Bayes_Data_Source_ModelDataGridView.Rows.Count) As String
Dim Wplus(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim Wminus(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim sWplus(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim sWminus(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single

For a = 0 To Bayes_Data_Source_ModelDataGridView.Rows.Count
Try
Appendix C

FileName(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(6).Value
Wplus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(1).Value
Wminus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(2).Value
sWplus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(4).Value
sWminus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(5).Value

Catch : Exit For
Next

missing_evidence_Pk(FileName, Wplus, Wminus)
'missing_evidence_PDx(FileName, Wplus, Wminus)
'SigmaSQ_missing_evidence()
SigmaSQ(FileName, sWplus, sWminus)
OminbusTest_NewOminibus()

End Sub

Private Function missing_evidence_Pk(ByVal FileName() As String,
ByVal Wplus() As Single, ByVal Wminus() As Single) As Integer

'looks fine
'here we use the first part of bayes to get the value of posterior probability
'we have to use the postlogit array as prior probability when we calculate P(d:x)
'and P(d)
'weights are automatically picked from the data grid view, but the arrays with final summation of weights
'has to be stripped of values that are not included within deposits and missing evidence ???
'we need to input a layer representing the area of missing evidence = bmp indexed image then perform a boolean with deposit and this layer to get only values useful for the calculation, we consider only missing evidence as we are using it to calculate the posterior probability resulting when missing evidence is intersected
'therefore a cumulative area obtained combining all the missing evidence on a single layer positively defines
'the pixel where this calculation is meaningful
'in any case we cannot calculate sigma^2(p) for missing evidence if there is no missing evidence
Dim Form5Inst As New Form5

Dep = Form5Inst.AD(TextBox8.Text)
imagesize = Bitmap.FromFile(FileName(0))

'Static header() As Byte 'on
Appendix C

Dim postlogit(((imagesize.Height * imagesize.Width) - 1) As Single
Dim postlogit_miss(((imagesize.Height * imagesize.Width) - 1) As Single
'Dim weightedArray(((imagesize.Height * imagesize.Width) - 1) As Single
'Static rescale(((imagesize.Height * imagesize.Width) - 1) As Single

Dim ab As Integer
For ab = 0 To FileName.Length - 1
  If FileName(ab) = Nothing Then
    Exit For
  End If
  Dim evidence As Bitmap = Bitmap.FromFile(FileName(ab).ToString)
  Dim myfile(((evidence.Width * evidence.Height) - 1) As Byte
  Dim color As System.Drawing.Color
  Dim county As Integer
  Dim countx As Integer
  Dim s As Integer
  For county = 0 To evidence.Height - 1
    For countx = 0 To evidence.Width - 1
      color = evidence.GetPixel(countx, county)
      If CInt(color.B) <> 0 And CInt(color.G) <> 0 And CInt(color.R) <> 0 Then
        myfile(s) = 1
      Else
        myfile(s) = 0
      End If
      s = s + 1
    Next
  Next
  s = 0
  Dim value As Byte
  Dim ard As Integer
  If FileName(ab) Is Nothing Then 'solves issue of final (0)
    Exit For
  Else
    'here we convert the image in evidence layer using the weights
    For ard = 0 To myfile.Length - 1
      value = myfile(ard)
If value = 0 Then
    weightedArray(ard) = Wminus(ab)
Else
    weightedArray(ard) = Wplus(ab)
End If

'For y = 0 To imagesize.Height - 1
    'For x = 0 To imagesize.Width - 1
    '  'value = imagesize(a)
    
    If imagesize.GetPixel(x, y) = Color.FromArgb(0, 0, 0) Then
        weightedArray(x * y) = Wminus(ab)
    Else
        weightedArray(x * y) = Wplus(ab)
    End If
    'Next
'Next

End If

' now we need to add the evidence layer created to the prior probability
' priologit should be calculated as the ratio D/T the conversion in ODDS and
' ln function gives priologit value
' I need the deposit layer and the tot number of pixels

' Dep = Form5Inst.AD(TextBox8.Text)
' Dim Dep_conv As Single = Dep * 0.000269
Dim priprob As Single = (Dep / (imagesize.Height * imagesize.Width))
Dim priOdd As Single = priprob / (1 - priprob)
Dim con As Single
con = Math.Log(Double.Parse(priOdd))

Dim ac As Integer
Dim priorlogit((imagesize.Height * imagesize.Width) - 1)
As Single 'five

' con = Single.Parse(TextBox6.Text)
For ac = 0 To ((imagesize.Height * imagesize.Width) - 1)
    priorlogit(ac) = con
Next
For ac = 0 To postlogit.Length - 1
    If ab = 0 Then
        postlogit(ac) = postlogit(ac) + priorlogit(ac) + weightedArray(ac)
    Else
        postlogit(ac) = postlogit(ac) + weightedArray(ac)
    End If
    'we get a postlogit value that is the sum of all images- as postlogit is declared
    'outside the For "ab" loop, each cycle updates its value
Next

'From this point we introduce new code that considers the missing evidence
'the code loads the missing evidence layer a boolean image
'where pixel columns contain missing information the weights are turned to (0)
Dim MissingEv As String
Dim MEArray() As Byte
Dim test As String = Bayes_Data_Source_ModelDataGridView.Rows(ab).Cells(3).Value.ToString
    If Not Bayes_Data_Source_ModelDataGridView.Rows(ab).Cells(3).Value.ToString = "" Then
        MissingEv = Bayes_Data_Source_ModelDataGridView.Rows(ab).Cells(3).Value
        MEArray = My.Computer.FileSystem.ReadAllBytes(MissingEv)
        Array.Reverse(MEArray)
        Array.Resize(MEArray, imagesize.Height * imagesize.Width)
    'Array.Reverse(MEArray) changed as all others were not reversed to original like bayes to display purpose
        Dim a As Integer
        For a = 0 To MEArray.Length - 1
            If Not MEArray(a) = 0 Then
                'seems that here we turn to 0 all the pixels that has at least one layer with missing evidence in it
                'the weightedArray is computed multiple times for each j layer so we put 0 in each layer with missing evidence
                weightedArray(a) = 0
            End If
        Next
Appendix C

End If
Next

End If

For ac = 0 To postlogit.Length - 1
    If ab = 0 Then
        postlogit_miss(ac) = postlogit_miss(ac) + priorlogit(ac) + weightedArray(ac)
    Else
        postlogit_miss(ac) = postlogit_miss(ac) + weightedArray(ac)
    End If
    ' we get a postlogit value that is the sum of all images- as postlogit is declared
    ' outside the For "ab" loop, each cycle updates its value
    Next
Next
Next

' I think that here we can insert some code to filter out the postlogit that are needed for the missing ev.
' postlogit will be used as p(d:x)

' MsgBox(postlogit(postlogit.Length - 1).ToString)
' Dim am As Integer
' For am = 0 To postlogit.Length - 1

'    ' we need to convert to integer
'    ' Dim MyDouble As Double = 42.72
'    ' Dim MyInt As Integer = CType(MyDouble, Integer)
'    ' MyInt has the value of 43.

' Next

' For am = 0 To postlogit.Length - 1
'   My.Computer.FileSystem.WriteAllText("c:\wofe\test", postlogit(am), True)
' Next
'now we need to rescale the double in a way that we can generate a byte array

'Dim ad As Integer
Dim aq As Integer

Dim postprob(postlogit.Length - 1) As Single
Dim postprob_miss(postlogit.Length - 1) As Single
' Dim postprobByte(postlogit.Length - 1) As Byte
For aq = 0 To postlogit.Length - 1
've we convert to probability
'postodds= exp(postlogit)
'postprob= postodds/(1+postodds)

'this post prob is Pk of Carranza 2004
postprob(aq) = Math.Exp(postlogit(aq)) / (1 + Math.Exp(postlogit(aq)))
postprob_miss(aq) = Math.Exp(postlogit_miss(aq)) / (1 + Math.Exp(postlogit_miss(aq)))

'we recover the alghorithm need to be modified the input as we have to filter out the areas without the missing evidence
'note that we need just the pixels intersecting a deposit therefore we have to perform the summation of weights only overlapping with pixel 1 of deposit layer
'an if statement should work
'we consider only the weights of layers holding the missing evidence as the other weights were previously updated, therefore we introduce an updated prior probability that already considers the weight of layers without missing evidence
Next

'These below are two arrays with values of Posterior prob in case of non-missing or missing evidence
'We can save these arrays as binary files but this will need to convert them in a scale of 255 bytes
PkD = postprob_miss
PDx = postprob

If CheckBox1.Checked Then

BinArcon(PkD)
Appendix C

Try

My.Computer.FileSystem.RenameFile("c:\wofe\arraybin.bin", "PkD_miss.bin")
  Catch ex As Exception
    File.Delete("c:\wofe\PkD_miss.bin")
  End Try

My.Computer.FileSystem.RenameFile("c:\wofe\arraybin.bin", "PkD_miss.bin")
Else
  PkD = Pdx
  File.Delete("c:\wofe\PkD_miss.bin")
End If

BinArcon(PDx)
Try
  My.Computer.FileSystem.RenameFile("c:\wofe\arraybin.bin", "Pdx.bin")
  Catch ex As Exception
    File.Delete("c:\wofe\Pdx.bin")
    My.Computer.FileSystem.RenameFile("c:\wofe\arraybin.bin", "Pdx.bin")
  End Try

' Dim acd As Integer
' Dim counter As Integer
' Dim d() As Byte = My.Computer.FileSystem.ReadAllBytes("d.bin")
' For acd = 0 To postprob.Length - 1
'   If d(acd) > 0 Then
'     PkD(counter) = postprob(acd)
'     counter = counter + 1
'   End If
' Next

End Function

Private Sub Button12_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button12.Click
  My.Forms.Form6WofeViewer.Close()
  My.Forms.Form6WofeViewer.Show()
End Sub
Private Function SigmaSQ(ByVal FileName() As String, ByVal sWplus() As Single, ByVal sWminus() As Single) As Integer

    ' I need to firstly select the bitmaps created with convert, then store the strings in the first column 'manually we type in the weights and ask for that with a msg box 'all is set and ready for the calculation 'we need to select each row and use the loop to convert the pixels to an array of weights 'we need to sum up the images to the prior probability image

    Dim s2SumWeights((imagesize.Height * imagesize.Width) - 1) As Single

    Dim s2SumWeights_miss((imagesize.Height * imagesize.Width) - 1) As Single

    Dim ab As Integer

    For ab = 0 To FileName.Length - 1
        Dim weightedArray((imagesize.Height * imagesize.Width) - 1) As Single ' six necessary to weight step
        ' ' ' new code from here
        If FileName(ab) = Nothing Then Exit For
        End If
        Dim evidence As Bitmap = Bitmap.FromFile(FileName(ab).ToString)

        Dim myfile((evidence.Width * evidence.Height) - 1) As Byte
        Dim color As System.Drawing.Color
        Dim county As Integer
        Dim countx As Integer
        Dim s As Integer

        For county = 0 To evidence.Height - 1
            For countx = 0 To evidence.Width - 1
                color = evidence.GetPixel(countx, county)
                If CInt(color.B) <> 0 And CInt(color.G) <> 0 And CInt(color.R) <> 0 Then
                    myfile(s) = 1
                Else
                    myfile(s) = 0
                End If
                s = s + 1
            Next countx
        Next county
    Next ab

    Dim s2SumWeights((imagesize.Height * imagesize.Width) - 1) As Single
    Dim s2SumWeights_miss((imagesize.Height * imagesize.Width) - 1) As Single

    Dim ab As Integer
    For ab = 0 To FileName.Length - 1
        Dim weightedArray((imagesize.Height * imagesize.Width) - 1) As Single ' six necessary to weight step
        ' ' ' new code from here
        If FileName(ab) = Nothing Then Exit For
        End If
        Dim evidence As Bitmap = Bitmap.FromFile(FileName(ab).ToString)

        Dim myfile((evidence.Width * evidence.Height) - 1) As Byte
        Dim color As System.Drawing.Color
        Dim county As Integer
        Dim countx As Integer
        Dim s As Integer

        For county = 0 To evidence.Height - 1
            For countx = 0 To evidence.Width - 1
                color = evidence.GetPixel(countx, county)
                If CInt(color.B) <> 0 And CInt(color.G) <> 0 And CInt(color.R) <> 0 Then
                    myfile(s) = 1
                Else
                    myfile(s) = 0
                End If
                s = s + 1
            Next countx
        Next county
    Next ab
Next
Next
s = 0
' here we start looping the different layers
Dim value As Byte
Dim ard As Integer
If FileName(ab) Is Nothing Then Exit For
Else
' here we convert the image in evidence layer using the
weights
For ard = 0 To myfile.Length - 1
    value = myfile(ard)
    If value = 0 Then
        weightedArray(ard) = Math.Pow(sWminus(ab), 2)
    Else
        weightedArray(ard) = Math.Pow(sWplus(ab), 2)
    End If
Next
End If
Dim ac As Integer
' progressively the weights grow
For ac = 0 To s2SumWeights.Length - 1
    ' this result works if no missing evidence is
considered
    s2SumWeights(ac) = s2SumWeights(ac) + weightedArray(ac)
Next

' From this point we introduce new code that considers the
missing evidence
' the code loads the missing evidence layer a boolean
image
' where pixel columns contain missing information the
weights are turned to (0)
Dim MissingEv As String
Dim MEArray() As Byte
Dim test As String = Bayes_Data_Source_ModelDataGridView.Rows(ab).Cells(3).Value.ToString
If Not Bayes_Data_Source_ModelDataGridView.Rows(ab).Cells(3).Value.ToString = "" Then
    MissingEv = Bayes_Data_Source_ModelDataGridView.Rows(ab).Cells(3).Value
    MEArray = My.Computer.FileSystem.ReadAllBytes(MissingEv)
    Array.Reverse(MEArray)
    Array.Resize(MEArray, imageSize.Height * imageSize.Width)
    'Array.Reverse(MEArray) changed as all others were not reversed to original like bayes to display purpose
    Dim a As Integer
    For a = 0 To MEArray.Length - 1
        If Not MEArray(a) = 0 Then
            weightedArray(a) = 0
        End If
    Next

'progressively the weights grow
For ac = 0 To s2SumWeights.Length - 1
    'this result works if missing evidence is considered
    'note that s2SumWeights_miss is different
    s2SumWeights_miss(ac) = s2SumWeights_miss(ac) + weightedArray(ac)
Next

Dim cellsz = TextBox9.Text
Dim s2Pk(PkD.Length - 1) As Single
Dim arl As Integer
Dim s2Pk_tot(PkD.Length - 1) As Single
Dim s2PDx(PDx.Length - 1) As Single
For arl = 0 To s2SumWeights.Length - 1
    'here we consider missing evidence as not really missing
    'so we use either positive or negative s2(weights)
    s2PDx(arl) = (((1 / (Dep * cellsz)) + s2SumWeights(arl)) * Math.Pow(PDx(arl), 2))

    'here we consider the missing evidence as 0 so the s(W)
    'become 0 when the evidence is missing, this is equivalent to
    'summing up only patterns that have weights on them
    If CheckBox1.Checked Then
        s2Pk(arl) = (((1 / (Dep * cellsz)) + s2SumWeights_miss(arl)) * Math.Pow(PkD(arl), 2))
'here we get the total s(Pk) adding the re-estimated influence of missing evidence due to its uncertainty
'this should improve our error estimate
'here I want a message box that split the calculation

If CheckBox1.Checked Then
    s2Pk_tot(arl) = s2Pk(arl) + SigmaSQ_missingev(arl)
End If

Next
'out of this we get 2 matrix one is s2pk and the other is s2Pk_tot, we have already created the function that converts probability arrays to maps so we just need to provide the files to that function
'for uncertainty maps we just then want s2pk and s2Pk_tot
If CheckBox1.Checked Then
    s2Pkf_miss = Spk_sum(s2Pk_tot)
    BinArcon(s2Pk_tot)
    Try
      My.Computer.FileSystem RENAMEFILE("c:\wofe\arraybin.bin", "s2Pk_tot_miss.bin")
      Catch ex As Exception
        File.Delete("c:\wofe\s2Pk_tot_miss.bin")
      End Try
End If

s2Pkd = Spk_sum(s2Pdx)
BinArcon(s2Pdx)
Try
  My.Computer.FileSystem RENAMEFILE("c:\wofe\arraybin.bin", "s2Pdx.bin")
  Catch ex As Exception
    File.Delete("c:\wofe\s2Pdx.bin")
    My.Computer.FileSystem RENAMEFILE("c:\wofe\arraybin.bin", "s2Pdx.bin")
  End Try
' this function provides the final s^2(Pk) results considering the two cases of missing or non missing evidence, these also represent the s^2(N|D)pred
'remains to estimate the values of N(D) and N(D|pred), we have to be careful
'as there is a change of variables also during the estimation of N(D|pred)
'if we consider the missing evidence

End Function
'Public Sub SigmaSQ_missing_evidence()
'
'    Dim a As Integer
'    Dim b(PkD.Length - 1) As Double
'
'    For a = 0 To PkD.Length - 1
'        If PDx(a) - PkD(a) = 0 Then
'            b(a) = 0
'        Else
'            b(a) = Math.Pow((PDx(a) - PkD(a)), 2) ' * 'frequency
of occurrence for class deltaPost(k)(1 / (image.Post.Width *
image.Post.Height) * cellsz)
'        End If
'    Next
'    SigmaSQm = b
'End Sub

Private Function Spk_sum(ByVal Input() As Single) As Single
    cellsz = TextBox9.Text
    Dim arr1(Input.Length - 1) As Single
    Input.CopyTo(arr1, 0) ' e.g. s2Pk
    Dim arr2(arr1.Length - 1) As Single
    Dim arr3(arr2.Length - 1) As Single
    Dim b As Integer = 0
    Dim spk As Double = 0
    Array.Sort(arr1)
    Dim a As Integer = Nothing
    'This cycle loop through the array s2Pk and define its classes
    that are summarised in arr2
    For a = 0 To arr1.Length - 2
        If Not arr1(a) = arr1(a + 1) Then
            Array.ConstrainedCopy(arr1, a, arr2, b, 1)
            b = b + 1
            arr2(b) = arr1(a + 1) ' classes
        End If
    Next
    Array.Resize(arr2, Array.IndexOf(arr2, Nothing))
    'This cycle counts the number of elements within s2Pk for each
    defined class
    For a = 0 To arr2.Length - 1
        For b = 0 To arr1.Length - 1
            If arr2(a) = arr1(b) Then
                arr3(a) = arr3(a) + 1 ' counter
            End If
        Next
    Next
    Array.Resize(arr3, Array.IndexOf(arr3, Nothing))
    Dim arr4(arr2.Length - 1) As Single
'This cycle creates an array that computes the square value of the area of each class mutated for its value then finally all the cumulative classes of spk are summed up to get the total value (this number when is big it means that there might be overestimation)

For a = 0 To arr2.Length - 1
    arr4(a) = Math.Pow((arr3(a) * cellsz), 2) * arr2(a)
    spk = spk + arr4(a)
Next
'MsgBox((spk), MsgBoxStyle.OKOnly)
Return spk

End Function

Private Sub OminbusTest_NewOminibus()

'we need to compute the summation of Pk or PDx depending if we consider missing evidence or not
'Carranza uses PDx instead of Pk to verify the influence of missing evidence layers

Dim cellsz = TextBox9.Text
Dim a As Integer
Dim NDpred As Double = 0
Dim NDpred_m As Double = 0
For a = 0 To PkD.Length - 1
    NDpred_m = NDpred_m + PkD(a) 'these are not standard deviations
Next
NDpred = NDpred + PDx(a)

Next
PkD = Nothing
PDx = Nothing
Dim OT As Single = 0
Dim OT_m As Single = 0
OT = Dep / NDpred
Label39.Text = OT
OT_m = Dep / NDpred_m
Label41.Text = OT_m
' MsgBox("OT should be higher than 0.85<< " & "OT " & OT.ToString & "OT_m " & OT_m.ToString)

Dim NewOT As Single = 0
Dim NewOT_m As Single = 0

NewOT_m = ((NDpred_m * cellsz) - (Dep * cellsz)) / Math.Sqrt(s2Pkf_miss)
Label145.Text = NewOT_m
NewOT = ((NDpred * cellsz) - (Dep * cellsz)) / Math.Sqrt(s2Pkf)
Label143.Text = NewOT
'MsgBox("NewOT should be lower than 0.7>> " & "NewOT " & NewOT.ToString & "NewOT_m " & NewOT_m.ToString)

Dim file As System.IO.StreamWriter

file.WriteLine("OT   OT_m")
file.WriteLine(OT & "  " & OT_m)
file.WriteLine("NewOT   NewOT_m")
file.WriteLine(NewOT & "  " & NewOT_m)
file.Close()

End Sub

Public Sub open_dep()
    'Dim myStream As Stream

    Dim selectFileDialog1 As New OpenFileDialog()
    selectFileDialog1.InitialDirectory = "c:\wofe\"
    selectFileDialog1.Filter = "bmp files (*.bmp)|*.bmp|All files (*.*)|*.*"
    selectFileDialog1.FilterIndex = 2
    selectFileDialog1.RestoreDirectory = True

    If selectFileDialog1.ShowDialog() = DialogResult.OK Then
        Me.TextBox8.Text = selectFileDialog1.FileName
        
        'myStream = selectFileDialog1.OpenFile()
        'If Not (myStream Is Nothing) Then
        '    ' Insert code to read the stream here.
        'myStream.Close()
        'End If
        'End If

    End If

End Sub

Private Sub saveToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles saveToolStripMenuItem.Click
    If Me.Validate Then
        Me.Bayes_Data_Source_ModelBindingSource.EndEdit()
        Me.Bayes_Data_Source_ModelTableAdapter.Update(Me.BayesDataSet.Bayes_Data_Source_Model)
    Else
Appendix C

Leonardo Feltrin
End Try


ElseIf DialogResult.Cancel Then
  Exit Sub

End If

' TODO: This line of code loads data into the
' FirstDatabaseDataSet.sysdiagrams' table. You can move, or remove it, as needed.
Me.Bayes_Data_Source_ModelTableAdapter.Fill(Me.BayesDataSet.Bayes_Data_Source_Model)

'     bmpImage = CType(Bitmap.FromFile(openFileDialog.FileName, False), Bitmap)
'     Me.AutoScale = False
'     ' Me.AutoSizeMinSize = New Size(CInt(bmpImage.Width * curZoom), CInt(bmpImage.Height * curZoom))
'     ' Me.Invalidate()
'     ' zoomMode = True
'     'End If
'     ' curRect = New Rectangle(0, 0, bmpImage.Width, bmpImage.Height)
'     'originalSize.Width = bmpImage.Width
'     'originalSize.Height = bmpImage.Height
End Sub

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
  ' TODO: This line of code loads data into the
  ' BayesDataSet.Bayes_Data_Source_Model' table. You can move, or remove it, as needed.
  Me.Bayes_Data_Source_ModelTableAdapter.Fill(Me.BayesDataSet.Bayes_Data_Source_Model)
End Sub

Private Sub bindingNavigatorSaveItem_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles bindingNavigatorSaveItem.Click
  If Me.Validate Then
    Me.Bayes_Data_Source_ModelBindingSource.EndEdit()
Me.Bayes_Data_Source_ModelTableAdapter.Update(Me.BayesDataSet.Bayes_Data_Source_Model)
Else
End If
Try
    My.Computer.FileSystem.CopyFile("Bayes.mdb", "c:\wofe\DataBase\Bayes.mdb")
Catch ex As Exception
    My.Computer.FileSystem.DeleteFile("c:\wofe\DataBase\Bayes.mdb")
    My.Computer.FileSystem.CopyFile("Bayes.mdb", "c:\wofe\DataBase\Bayes.mdb")
End Try
End Sub

Private Sub Bayes_Data_Source_ModelDataGridView_MouseWheel(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs)
Handles Bayes_Data_Source_ModelDataGridView.MouseWheel
Dim selectFileDialog1 As New OpenFileDialog()
selectFileDialog1.InitialDirectory = "c:\"
selectFileDialog1.Filter = "bmp files (*.bmp)|*.bmp|All files (*.*)|*.*"
selectFileDialog1.FilterIndex = 2
selectFileDialog1.RestoreDirectory = True

'If selectFileDialog1.ShowDialog() = DialogResult.Cancel Then
    Exit Sub
'If selectFileDialog1.ShowDialog() = DialogResult.OK Then
    'myStream = selectFileDialog1.OpenFile()
    'If Not (myStream Is Nothing) Then
    '    Insert code to read the stream here.
    'myStream.Close()
End If
End Sub

Private Function BinArcon(ByVal input() As Single) As Byte

    'This function is a generalisation of the second part of the Bayes alghorithm, it is designed to solve a problem of compatibility between arrays and bmp files, basically we need a conversion from single to byte
    'The main difficulty is that this conversion has to represent probability in a scale of 255 colors.
'Casting of single without rescaling will likely reduce all  
the array values to 0. Therefore it is useful to  
'firstly define the minimum value of the array then multiply  
for an appropriate scaling factor to  
'obtain a range of values large than delta255  
'We need a string to save the name of each output file

Dim ad As Integer

Dim postprobByte(input.Length - 1) As Byte
Dim rescale(input.Length - 1) As Single

Array.Copy(input, rescale, input.Length)
Array.Sort(rescale)

Dim min As Single = rescale(0)
Dim max As Single = rescale(rescale.Length - 1)

My.Computer.FileSystem.WriteAllText("c:\wofe\scale_M.txt",  
"min = " & min.ToString & " max = " & max.ToString, False)

For ad = 0 To input.Length - 1
    ' we shift or translate the scale to get a minimum of 0  
    ' min must become firstly equal to 1 so if min*x=1  
    Dim alfa As Single = (1 / min)  
    input(ad) = input(ad) * alfa  
    input(ad) = (input(ad) - (min * alfa))  
    input(ad) = (input(ad) * 255) / ((max * alfa) - (min * alfa))
    postprobByte(ad) = CByte((input(ad)))
    input(ad) = (input(ad) * ((max * alfa) - (min * alfa))) / 255
    input(ad) = (input(ad) + (min * alfa))
    input(ad) = input(ad) / alfa
Next

Dim response As MsgBoxResult = MsgBox("Binary conversion  
completed, would you like to save the file?", MsgBoxStyle.YesNo)

If response = MsgBoxResult.Yes Then
    Dim saveFileDialog1 As New SaveFileDialog()
    saveFileDialog1.InitialDirectory = "c:\wofe"
    saveFileDialog1.Filter = "bin files (*.bin)|*.bin|All  
files (*.*)|*.*"
    saveFileDialog1.FilterIndex = 2
    saveFileDialog1.RestoreDirectory = True
If saveFileDialog1.ShowDialog() = DialogResult.OK Then

End If
Else

My.Computer.FileSystem.WriteAllBytes("c:\wofe\arraybin.bin", postprobByte, False)
End If

End Function

Private Sub Button11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button11.Click

' This function save a bmp file of an array of posterior probability

Dim myStream As Stream
Dim selectFileDialog1 As OpenFileDialog()

selectFileDialog1.InitialDirectory = "c:\wofe\"
selectFileDialog1.Filter = "bin files (*.bin)|*.bin|All files (*.*)|*.*"
selectFileDialog1.FilterIndex = 2
selectFileDialog1.RestoreDirectory = True

'first dialog fpr selection of binary array to map
If selectFileDialog1.ShowDialog() = DialogResult.OK Then

Dim name As String = selectFileDialog1.FileName
Dim postprobByte() As Byte = My.Computer.FileSystem.ReadAllBytes(name)
MsgBox("Please select reference (rows*column bmp file)", MsgBoxStyle.OKOnly)

'Array.Reverse(postprobByte)

'second dialogue box for selection of a bmp as reference
Dim selectFileDialog2 As New OpenFileDialog()

selectFileDialog2.InitialDirectory = "c:\wofe\"
selectFileDialog2.Filter = "bmp files (*.bmp)|*.bmp|All files (*.*)|*.*"
selectFileDialog2.FilterIndex = 2
selectFileDialog2.RestoreDirectory = True

If selectFileDialog2.ShowDialog() = DialogResult.OK Then

'There might be a problem with loading this binary file in term of header
bayesmap = Bitmap.FromFile(selectFileDialog2.FileName)
Dim x As Integer
Dim y As Integer
Dim count As Integer = 0

' (bayesmap.Width - 1) - inverted also y with x

For y = 0 To bayesmap.Height - 1
    For x = 0 To bayesmap.Width - 1
        bayesmap.SetPixel(x, y, Color.FromArgb(postprobByte(count), postprobByte(count), postprobByte(count)))
        count = count + 1
    Next
Next

File.Delete(name & ".bmp")
bayesmap.Save(name & ".bmp")

MsgBox("Map completed and saved in " & name & ".bmp")
End If
End If
End Sub

Private Sub Button14_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button14.Click
' This routine is compiled to solve the problem of calculating the additional error seen as variance of the posterior probability. The objective is to create a function that input info from the datagrid view in particular the weights and the relative missing patterns, in term of number and area

' Here we select the patterns with missing evidence
'start new ot ot

Try
    cell = Me.TextBox9.Text
Catch ex As Exception
    MsgBox("Please provide cell size")
    Exit Sub
End Try

d = Nothing
PkD = Nothing
PDx = Nothing
SigmaSQm = Nothing
s2Pk = Nothing
s2Pkf = Nothing
s2Pkf_miss = Nothing
Dep = Nothing
bmpsizepub = Nothing
AD_w = Nothing
AI_w = Nothing
AG_w = Nothing
AT_w = Nothing 'shared variables expressing areas as cell
numbers; they work in all sub routines

Dim a As Integer
'Dim b As Integer
Dim FileName(Bayes_Data_Source_ModelDataGridView.Rows.Count) As String
Dim Wplus(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim Wminus(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim sWplus(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim sWminus(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim MissEv(Bayes_Data_Source_ModelDataGridView.Rows.Count) As String
Dim FileName_r(Bayes_Data_Source_ModelDataGridView.Rows.Count) As String
Dim Wplus_r(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim Wminus_r(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim sWplus_r(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim sWminus_r(Bayes_Data_Source_ModelDataGridView.Rows.Count) As Single
Dim MissEv_r(Bayes_Data_Source_ModelDataGridView.Rows.Count) As String

For a = 0 To Bayes_Data_Source_ModelDataGridView.Rows.Count
  'DBnull
  Try
    If Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(3).Value Is System.DBNull.Value Then
      Exit Try
    End If
    FileName(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(6).Value
    Wplus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(1).Value
    Wminus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(2).Value
    sWplus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(4).Value
    sWminus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(4).Value
  Exit Try
  End If
  FileName_r(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(6).Value
  Wplus_r(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(1).Value
  Wminus_r(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(2).Value
  sWplus_r(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(4).Value
  sWminus_r(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(4).Value
  MissEv(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(4).Value
  MissEv_r(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(4).Value
Next a
sWminus(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(5).Value
MissEv(a) = Bayes_Data_Source_ModelDataGridView.Rows(a).Cells(3).Value

Catch : Exit For

End Try

Next

Dim b As Integer
For a = 0 To FileName.Length - 1
  If FileName(a) <> Nothing Then
    FileName_r(b) = FileName(a)
    Wplus_r(b) = Wplus(a)
    Wminus_r(b) = Wminus(a)
    sWplus_r(b) = sWplus(a)
    sWminus_r(b) = sWminus(a)
    MissEv_r(b) = MissEv(a)
    b = b + 1
  Else
  End If
Next

' we need to reduce FileName to just the layer with missing evidence in it
missing_evidence(FileName_r, Wplus_r, Wminus_r, MissEv_r)

End Sub

Private Function missing_evidence(ByVal FileName_r() As String, ByVal Wplus_r() As Single, ByVal Wminus_r() As Single, ByVal MissEv_r() As String) As Integer

' looks fine
' here we use the first part of bayes to get the value of posterior probability
' we have to use the postlogit array as prior probability when we calculate P(d:x)
' and P(d)
' weights are automatically picked from the data grid view, but the arrays with final summation of weights
' has to be stripped of values that are not included within deposits and missing evidence ???
' we need to input a layer representing the area of missing evidence = bmp indexed image then perform a boolean with
' deposit and this layer to get only values useful for the calculation, we consider only missing evidence as we are
'using it to calculate the posterior probability resulting
when missing evidence is intersected
therefore a cumulative area obtained combining all the
missing evidence on a single layer positively defines
the pixel where this calculation is meaningfull
'in any case we cannot calculate $\sigma^2(p)$ for missing
evidence if there is no missing evidence
Dim Form5Inst As New Form5
Dim SigmaSQ2(imagesize.Height * imagesize.Width - 1) As Double
Dep = Form5Inst.AD(TextBox8.Text)
imagesize = Bitmap.FromFile(FileName_r(0))

'Static header() As Byte 'on

'four
'six necessary to weight step
' Static rescale((imagesize.Height * imagesize.Width) - 1) As Single 'seven
Dim ab As Integer
For ab = 0 To FileName_r.Length - 1
    Dim weightedArray((imagesize.Height * imagesize.Width) - 1) As Single
    Dim postlogit((imagesize.Height * imagesize.Width) - 1) As Single
    Dim postlogit_miss((imagesize.Height * imagesize.Width) - 1) As Single
    Dim postlogit_wplus((imagesize.Height * imagesize.Width) - 1) As Single
    Dim postlogit_wminus((imagesize.Height * imagesize.Width) - 1) As Single
    Try
        ' imagesize = Bitmap.FromFile(FileName(ab))
        myfile = My.Computer.FileSystem.ReadAllText(FileName_r(ab)) 'may need to
        string -- 'three
        Catch : Exit Try
        End Try
    'need to strip the header
    ' Dim headerfile() As Byte =
    Dim headerfile() As Byte =
    My.Computer.FileSystem.ReadAllText("c:\wofe\imageheader.bmp")
    'header = headerfile
    'Array.Resize(header, (headerfile.Length -
    (imagesize.Height * imagesize.Width)))
    Array.Reverse(myfile)
    Array.Resize(myfile, (imagesize.Height * imagesize.Width))
    'Array.Reverse(myfile)
'here we start looping the different layers

'Dim x As Integer
'Dim y As Integer
Dim value As Byte
Dim ard As Integer

If FileName_r(ab) Is Nothing Then 'solves issue of final (0)
    Exit For
Else

'here we convert the image in evidence layer using the weights
For ard = 0 To myfile.Length - 1
    value = myfile(ard)
    If value = 0 Then
        weightedArray(ard) = Wminus_r(ab)
    Else
        weightedArray(ard) = Wplus_r(ab)
    End If
Next

'For y = 0 To imagesize.Height - 1
'    For x = 0 To imagesize.Width - 1
'        'value = imagesize(a)
'        If imagesize.GetPixel(x, y) = Color.FromArgb(0, 0, 0) Then
'            weightedArray(x * y) = Wminus(ab)
'        Else
'            weightedArray(x * y) = Wplus(ab)
'        End If
'    Next
'Next

End If

'now we need to add the evidence layer created to the prior probability
'prilogit should be calculated as the ratio D/T the conversion in ODDS and
'ln function gives prilogit value
'I need the deposit layer and the tot number of pixels

' Dep = Form5Inst.AD(TextBox8.Text)
' Dim Dep_conv As Single = Dep * 0.000269
Dim priprob As Single = (Dep / (imagesize.Height * imagesize.Width))
Dim priOdd As Single = priprob / (1 - priprob)
Dim con As Single
con = Math.Log(Double.Parse(priOdd))

Dim ac As Integer
Dim priorlogit((imagesize.Height * imagesize.Width) - 1) As Single
'five

' con = Single.Parse(TextBox6.Text)
For ac = 0 To ((imagesize.Height * imagesize.Width) - 1)
priorlogit(ac) = con
Next

'For ac = 0 To postlogit.Length - 1
'  If ab = 0 Then
'    postlogit(ac) = postlogit(ac) + priorlogit(ac) + weightedArray(ac)
'  Else
'    postlogit(ac) = postlogit(ac) + weightedArray(ac)
'  End If
'  'we get a postlogit value that is the sum of all images- as postlogit is declared
'  'outside the For "ab" loop, each cycle updates its value
'Next

'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'From this point we introduce new code that considers the missing evidence
'the code loads the missing evidence layer a boolean image
'where pixel columns contain missing information the weights are turned to (0)

Dim MissingEv As String
Dim MEArray() As Byte
' Dim test As String = Bayes_Data_Source_ModelDataGridView.Rows(ab).Cells(3).Value.ToString
MissingEv = MissEv_r(ab)
MEArray = My.Computer.FileSystem.ReadAllBytes(MissingEv)
Array.Reverse(MEArray)
Array.Resize(MEArray, imagesize.Height * imagesize.Width)
'Array.Reverse(MEArray) changed as all others were not reversed to original like bayes to display purpose

Dim a As Integer
For a = 0 To MEArray.Length - 1
 If Not MEArray(a) = 0 Then
   'seems that here we turn to 0 all the pixels that has at least one layer with missing evidence in it
   'the weightedArray is computed multiple times for each j layer so we put 0 in each layer with missing evidence
   weightedArray(a) = 0
 End If
Next

For ac = 0 To postlogit.Length - 1
 If ab = 0 Then
   postlogit_miss(ac) = postlogit_miss(ac) + priorlogit(ac) + weightedArray(ac)
 Else
   postlogit_miss(ac) = postlogit_miss(ac) + weightedArray(ac)
 End If
End If
'we get a postlogit value that is the sum of all images- as postlogit is declared
'outside the For "ab" loop, each cycle updates its value

Next

'LOOP to change to W+

For a = 0 To MEArray.Length - 1
 If Not MEArray(a) = 0 Then
   'seems that here we turn to 0 all the pixels that has at least one layer with missing evidence in it
   'the weightedArray is computed multiple times for each j layer so we put 0 in each layer with missing evidence
   weightedArray(a) = Wplus_r(ab)
End If
Next

For ac = 0 To postlogit.Length - 1
  If ab = 0 Then
    postlogit_wplus(ac) = postlogit_wplus(ac) + priorlogit(ac) + weightedArray(ac)
  Else
    postlogit_wplus(ac) = postlogit_wplus(ac) + weightedArray(ac)
  End If
  'we get a postlogit value that is the sum of all images- as postlogit is declared
  'outside the For "ab" loop, each cycle updates its value
Next

'LOOP to change to W-

For a = 0 To MEArray.Length - 1
  If Not MEArray(a) = 0 Then
    'seems that here we turn to 0 all the pixels that has at least one layer with missing evidence in it
    'the weightedArray is computed multiple times for each j layer so we put 0 in each layer with missing evidence
    weightedArray(a) = Wminus_r(ab)
  End If
Next

For ac = 0 To postlogit.Length - 1
  If ab = 0 Then
    postlogit_wminus(ac) = postlogit_wminus(ac) + priorlogit(ac) + weightedArray(ac)
  Else
    postlogit_wminus(ac) = postlogit_wminus(ac) + weightedArray(ac)
  End If
  'we get a postlogit value that is the sum of all images- as postlogit is declared
  'outside the For "ab" loop, each cycle updates its value

' now we have all three Pprob we need for the calculation
the area where the jth layer with Pattern Bj has 1 and the area with 0
value
' which is negBj

' Calculation of Area Bj
' For this we need the file path we call the binary file
and use a bit counter to find out the number of 0 and 1

Dim r As Integer
' Here we need some code that cleans up the bitmaps for us,
we declare 2 new myarrays that will be locally storing
'the original BMP in binary format. Then we get the data
out of them and feed SumArray
Dim PatternBj() As Byte =
My.Computer.FileSystem.ReadAllBytes(FileName_r(ab)) ' we call the file
with missing evidence

r = (imagesize.Height * imagesize.Width) ' the number of
cells needed to get the number of good data pixels

' tronca arrays con remove command
Array.Reverse(PatternBj)
Array.Resize(PatternBj, r)
Array.Reverse(PatternBj)

Dim b As Integer = 0
Dim Sum As Single = 0

' algebraic sum of array (PatternBj) we get just 1s not 0
counted, therefore the area of cells containing the pattern
For b = 0 To PatternBj.Length - 1
    Sum = Sum + PatternBj(b) '-1 is inserted as the array
Next b

Dim AreaPattern As Single = Sum
Appendix C

Source code

Dim Area_emptyPattern As Single = (r - AreaPattern)

'estimation of postprob for the 3 postlogits
Dim aq As Integer
Try

weightedArray = Nothing

Dim postprob_wminus(postlogit.Length - 1) As Single
Dim postprob_miss(postlogit.Length - 1) As Single
Dim postprob_wplus(postlogit.Length - 1) As Single

' Dim postprobByte(postlogit.Length - 1) As Byte
For aq = 0 To postlogit.Length - 1
' we convert to probability
'postodds= exp(postlogit)
'postprob= postodds/(1+postodds)

' this post prob is Pk of Carranza 2004
postprob_wplus(aq) = Math.Exp(postlogit_wplus(aq)) / (1 + Math.Exp(postlogit_wplus(aq)))
postprob_wminus(aq) = Math.Exp(postlogit_wminus(aq)) / (1 + Math.Exp(postlogit_wminus(aq)))
postprob_miss(aq) = Math.Exp(postlogit_miss(aq)) / (1 + Math.Exp(postlogit_miss(aq)))

Next

'sigmasq(Pk) = (P(DBj)-Pk)^2 * PBJ + (P(BnegBj)-Pk)^2 * PnegBj

cellsz = TextBox9.Text
For a = 0 To postlogit.Length - 1
SigmaSQ2(a) = (Math.Pow((postprob_wplus(a) - postprob_miss(a)), 2) * (AreaPattern * cellsz) / (imagesize.Height * imagesize.Width)) + (Math.Pow((postprob_wminus(a) - postprob_miss(a)), 2) * (Area_emptyPattern * cellsz) / ((imagesize.Height * imagesize.Width)))
Next
Catch ex As Exception
Exit Function
End Try
Appendix C

Leonardo Feltrin

Source code

C.2. Spatial Analyser

Compiled in VB 2005 (Express Edition)

Software used to compute minimum Euclidean distances (Chapter 6)

Imports System.IO
Imports System.Text
Imports System
Imports System.Drawing
Imports System.Drawing.Printing
Imports System.Collections
Imports System.ComponentModel
Imports System.Windows.Forms

Public Class Form1

    Dim breccia() As Double
    Dim faults() As Double
    Dim deltaMin() As Double

    'Array gen
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        'the button is used to convert a txt file to an array of data in double format
        'we need a the initialisation of the butto to get access to the hard drive and select the file
        Dim selectFileDialog1 As New OpenFileDialog()
        selectFileDialog1.InitialDirectory = "C:\Documents and Settings\Leonardo\My Documents"
selectFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"
selectFileDialog1.FilterIndex = 2
selectFileDialog1.RestoreDirectory = True

If selectFileDialog1.ShowDialog() = Windows.Forms.DialogResult.OK Then
    TextBox1.Text = selectFileDialog1.FileName
    'we need a call to the file that we host in a string
    Dim mystring As String = My.Computer.FileSystem.ReadAllText(TextBox1.Text)
    'This command split the string in an array of string containing xyz separated by space
    Dim mystring2() As String = Split(mystring, Environment.NewLine)

    Dim a As Integer
    Dim b As Integer
    Dim mystring4((mystring2.Length) * 3) As Double
    'the loop is used to split more all data are charging a single array
    For a = 0 To ((mystring2.Length - 1) * 3)
        Dim mystring3() As String = Split(mystring2(b), " ")
        mystring4(a) = Double.Parse(mystring3(0))
        mystring4(a + 1) = Double.Parse(mystring3(1))
        mystring4(a + 2) = Double.Parse(mystring3(2))
        a = a + 2
        'as b is always very small compared to a then it is reasonable the use of b in the same loop
        'the problem would be that the loop is based on a therefore if a ends earlier than b then
        b = b + 1
        If mystring2(b) = "" Then Exit For
    Next
    breccia = mystring4
    While breccia(breccia.Length - 1) = 0
        Array.Resize(breccia, breccia.Length - 1)
    End While
End If
End Sub

'Array gen
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    'the button is used to convert a txt file to an array of data in double format
    'we need a the initialisation of the button to get access to the hard drive and select the file

    Dim selectFileDialog1 As New OpenFileDialog()
selectFileDialog1.InitialDirectory = "C:\Documents and Settings\Leonardo\My Documents"
selectFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"
selectFileDialog1.FilterIndex = 2
selectFileDialog1.RestoreDirectory = True
If selectFileDialog1.ShowDialog() = Windows.Forms.DialogResult.OK Then
    TextBox2.Text = selectFileDialog1.FileName
    'we need a call to the file that we host in a string
    Dim mystring As String = My.Computer.FileSystem.ReadAllText(TextBox2.Text)
    Dim mystring2() As String = Split(mystring, Environment.NewLine)
    Dim a As Integer
    Dim b As Integer
    Dim mystring4(((mystring2.Length) * 5)) As Double
    'the loop is used to split more all data are charging a single array
    For a = 0 To (mystring2.Length * 5)
        Dim mystring3() As String = Split(mystring2(b), " ")
        mystring4(a) = Double.Parse(mystring3(0))
        mystring4(a + 1) = Double.Parse(mystring3(1))
        mystring4(a + 2) = Double.Parse(mystring3(2))
        mystring4(a + 3) = Double.Parse(mystring3(3))
        mystring4(a + 4) = Double.Parse(mystring3(4))
        a = a + 4
        b = b + 1
        If b > mystring2.Length - 1 Then Exit For
        If mystring2(b) = "" Then Exit For
    Next
    faults = mystring4
    While faults(faults.Length - 1) = 0
        Array.Resize(faults, faults.Length - 1)
    End While
End If

'Algorithm to compute minimum distances between the two pointsets
Private Sub MainAlgho()
    'variables
    Dim a As Integer
    Dim xb As Double
    Dim yb As Double
    Dim zb As Double

End Sub
Dim xf As Double
Dim yf As Double
Dim zf As Double
'clear txt file
Dim file As System.IO.StreamWriter
file.WriteLine(""")
file.Close()
'loops to select the coordinates to compute distance, store distances on delta
For a = 0 To breccia.Length - 1
    xb = breccia(a)
yb = breccia(a + 1)
zb = breccia(a + 2)
a = a + 2

Dim b As Integer = 0
Dim c As Integer = 0
Dim delta(faults.Length - 1) As Double 'delta lasts only inside the b loop
Dim deltacopy(faults.Length - 1) As Double
For b = 0 To faults.Length - 1
    xf = faults(b)
yf = faults(b + 1)
zf = faults(b + 2)
b = b + 4
    'delta gets all the distances of a single cbx point
    delta(c) = distance(xb, yb, zb, xf, yf, zf)
c = c + 1
Next
'copy delta for index search
Array.Copy(delta, deltacopy, delta.Length - 1)
'sort delta to get a minimum
Array.Resize(delta, c - 1)
Array.Sort(delta)

'compute index and retrieve d1 and d2 on faults array
Dim index As Integer = Array.IndexOf(deltacopy, delta(0))
Dim d1 As Double = faults((index * 5) + 3)
Dim d2 As Double = faults((index * 5) + 4)

'write halt instruction in case of 0 or multiple equal minimum values
If delta(0) = 0 Or delta(0) = delta(1) Then
'export values on a file text that is progressively updated for each breccia loop in case of 0 or multiple
file =
data\results.txt", True)
file.WriteLine(xb & "    & yb & " & zb & " & d1 & " & d2 & " & delta(0) & " err")
file.Close()

'export values on a file text that is progressively
updated for each breccia loop
Else
file =
data\results.txt", True)
file.WriteLine(xb & "    & yb & " & zb & " & d1 & " & d2 & " & delta(0))
file.Close()
End If

Next
MsgBox("Computation completed!")
End Sub

Function distance(ByVal xb As Double, ByVal yb As Double, ByVal zb As Double, ByVal xf As Double, ByVal yf As Double, ByVal zf As Double)
'Euclidean formula
Dim dist As Double = Math.Sqrt((Math.Pow(xb - xf, 2)) + (Math.Pow(yb - yf, 2)) + (Math.Pow(zb - zf, 2)))
Return dist
End Function

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
Me.mainAlgho()
End Sub
End Class
APPENDIX D
Table D.1. Summary of rock specimens used in this thesis, illustrating their relative location, age, lithology and stratigraphic collocation.

<table>
<thead>
<tr>
<th>Specimen ID</th>
<th>Location</th>
<th>Easting*</th>
<th>Northing</th>
<th>Age**</th>
<th>Lithotype</th>
<th>Stratigraphic Unit</th>
<th>Chapter</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDH114</td>
<td>PCM 338(171.41 m)</td>
<td>47132.61</td>
<td>27873.7</td>
<td>MP</td>
<td>Siltstone/Shale</td>
<td>Pmh4s</td>
<td>3</td>
<td>5a</td>
</tr>
<tr>
<td>CDH83</td>
<td>PCM325(271.17 m)</td>
<td>46849.8</td>
<td>27850.56</td>
<td>MP</td>
<td>Siltstone/Shale</td>
<td>Pmh4s</td>
<td>3</td>
<td>5b</td>
</tr>
<tr>
<td>CDH85</td>
<td>PCM325(235.20 m)</td>
<td>46849.8</td>
<td>27850.56</td>
<td>MP</td>
<td>Shale</td>
<td>Pmh4s</td>
<td>3</td>
<td>5c</td>
</tr>
<tr>
<td>HSCM103</td>
<td>Century Mine (St4)</td>
<td>47430</td>
<td>27460</td>
<td>MP</td>
<td>Shale</td>
<td>Pmh4s</td>
<td>3</td>
<td>5d</td>
</tr>
<tr>
<td>HSCMT</td>
<td>Century Mine (St4)</td>
<td>47400</td>
<td>27550</td>
<td>MP</td>
<td>Mudstone/Siltstone</td>
<td>Pmh4s</td>
<td>3</td>
<td>5e</td>
</tr>
<tr>
<td>RLWL01</td>
<td>Watson's Lode</td>
<td>246462</td>
<td>7916970</td>
<td>MP</td>
<td>Qtz/Sid vein</td>
<td>Pmh3</td>
<td>3</td>
<td>5c</td>
</tr>
<tr>
<td>RLWL02</td>
<td>Watson's Lode</td>
<td>246301</td>
<td>7916930</td>
<td>MP</td>
<td>Qtz vein</td>
<td>Pmh3</td>
<td>3</td>
<td>5d</td>
</tr>
<tr>
<td>HSCM103</td>
<td>Century Mine (St4)</td>
<td>47430</td>
<td>27460</td>
<td>MP</td>
<td>Siltstone/Shale</td>
<td>Pmh4s</td>
<td>5</td>
<td>2a</td>
</tr>
<tr>
<td>HSCM106</td>
<td>Century Mine (St4)</td>
<td>47410</td>
<td>27590</td>
<td>MP</td>
<td>Siltstone/Shale</td>
<td>Pmh4s</td>
<td>5</td>
<td>2b</td>
</tr>
<tr>
<td>HSCM109</td>
<td>Century Mine (St4)</td>
<td>47460</td>
<td>27510</td>
<td>MP</td>
<td>Siltstone/Shale</td>
<td>Pmh4s</td>
<td>5</td>
<td>2c</td>
</tr>
<tr>
<td>HSCM111</td>
<td>Century Mine (St4)</td>
<td>47340</td>
<td>27550</td>
<td>MP</td>
<td>Siltstone/Shale</td>
<td>Pmh4s</td>
<td>5</td>
<td>2d</td>
</tr>
<tr>
<td>HSCM33</td>
<td>Century Mine (St4)</td>
<td>47560</td>
<td>27313</td>
<td>MP</td>
<td>Siltstone/Shale</td>
<td>Pmh4s</td>
<td>5</td>
<td>2e</td>
</tr>
<tr>
<td>HSCM69</td>
<td>Century Mine (St4)</td>
<td>46920</td>
<td>28220</td>
<td>MC</td>
<td>CBX breccia</td>
<td>Thorntonia Limestones</td>
<td>6</td>
<td>11a</td>
</tr>
<tr>
<td>HSCM63</td>
<td>Century Mine (St4)</td>
<td>47070</td>
<td>28200</td>
<td>MC</td>
<td>MB Marl breccia</td>
<td>Thorntonia Limestones</td>
<td>6</td>
<td>11b</td>
</tr>
<tr>
<td>CDH55</td>
<td>PCM302(168 m)</td>
<td>47070</td>
<td>28250</td>
<td>MC</td>
<td>CLS nodular limestone</td>
<td>Thorntonia Limestones</td>
<td>6</td>
<td>11d</td>
</tr>
<tr>
<td>HSLHA67</td>
<td>Lawn Hill Annulus</td>
<td>251460</td>
<td>7928439</td>
<td>MC</td>
<td>Sandstone</td>
<td>Thorntonia Limestones</td>
<td>6</td>
<td>11e</td>
</tr>
<tr>
<td>HSCM64</td>
<td>Century Mine (St4)</td>
<td>46980</td>
<td>28240</td>
<td>MC</td>
<td>Karst breccia</td>
<td>Thorntonia Limestones</td>
<td>6</td>
<td>11f</td>
</tr>
<tr>
<td>LHCSO1</td>
<td>Lawn Hill Annulus</td>
<td>251049</td>
<td>7933520</td>
<td>MP</td>
<td>Siltstone</td>
<td>Pmh4s</td>
<td>6</td>
<td>25b</td>
</tr>
</tbody>
</table>

* Easting and Northing expressed either as AMG84 coordinates or Mine Grid coordinates.

** MP = Mesoproterozoic, MC = Middle Cambrian.
Basement metal scavenging during basin evolution: Cambrian and Proterozoic interaction at the Century Zn–Pb–Ag Deposit, Northern Australia

L. Feltrin, N.H.S. Oliver, I.J. Kelso, S. King

THE ABOVE ARTICLE FROM:

Journal of Geochemical Exploration, 4076 (2003), 1-4

HAS NOT BEEN INCLUDED DUE TO COPYRIGHT RESTRICTIONS
Evidence of multi-stage ore genesis at the Century zinc deposit, Northwestern Queensland, Australia

Leonardo Feltrin and Nicholas H.S. Oliver

THE ABOVE ARTICLE FROM:
Predictive Mineral Discovery CRC Conference, Barossa Valley, 1-3 June 2004
HAS NOT BEEN INCLUDED DUE TO COPYRIGHT RESTRICTIONS
A Visual Basic Express tool to perform 3D Weights of Evidence Modelling

Leonardo Feltrin

THE ABOVE ARTICLE FROM:

INT. ASSOC. FOR MATHEMATICAL GEOLOGY, XIth INTERNATIONAL CONGRESS

HAS NOT BEEN INCLUDED DUE TO COPYRIGHT RESTRICTIONS

Nicholas H.S. Oliver,¹ John G. McLellan,

Bruce E. Hobbs,

James S. Cleverley,

Alison Ord,

and Leonardo Feltrin

THE ABOVE ARTICLE FROM:

ECONOMIC GEOLOGY, Bulletin of the Society of Economic Geologists
v. 101(1), January-February 2006
100th Anniversary Special Paper

HAS NOT BEEN INCLUDED DUE TO COPYRIGHT RESTRICTIONS