* Lectura de fitxers. MDT en format .pif (laser scanner), esta en binari. 2. lecturaCalibracio() Per pasar de coordenades fotografia a MDT. 3. lecturamatrix() per projectes pif. 4. lecturaFichResPix() i lecturaFichcoord() son els arxius on s’enmagatzema les coordenades pixels de les fotografies digitalitzades i els fitxers de resultats amb una codificacio per saber quan es línia, final de línia i punt. 5. Lecturatxt
* Imports System
* Imports System.IO
* Imports System.Xml
* Module ReadFiles
* Public midapixel As Double
* Public oripixelx As Double
* Public oripixely As Double
* Public matrius(,,) As Double
* Public memoriacolor As Pen
* Public value As New ArrayList
* Public textcolor As Integer = 0
* Public NumMatrius() As Integer
* ' Public contador As Integer
* Public asa As Integer
* Public pointx() As Double '16 sep 2013 de single a double
* Public pointy() As Double '16 sep 2013 de single a double
* Public pointz() As Double '16 sep 2013 de single a double
* Public NumTotalBlocks As Integer
* Public seles As Integer
* Public DireccioFichResPix() As String
* Public DireccioFichCoord() As String
* Public direccioimatge(,) As String
* Public direccioPIF(,) As String
* Public direccioCalib(,) As String
* Public direcciomatrius(,) As String
* Public Rotacions(9) As Double
* Public Traslacions(3) As Double
* Public lectura() As String
* Public properties() As Boolean
* Public resx() As Single
* Public resy() As Single
* Public pixmax As Double = 0
* Public pixmin As Double = 1000000
* Public piymax As Double = 0
* Public piymin As Double = 1000000
* '\*\*\*\*\* Fitxer calibracio
* Public omega As Double ' 12 Rotació eix x de la càmera (radiants), original 0,amunt-
* Public phi As Double ' 13 Rotació eix y de la càmera (radiants), original 0, esquerra-
* Public kappa As Double '14 Rotació eix z de la càmera (radiants), original 0,lev -
* Public Xcam As Double ' 9 Coordenada x de la càmera
* Public Ycam As Double ' 10 Coordenada y de la càmera
* Public Zcam As Double ' 11 Coordenada z de la càmera
* Public f As Double '0.017 ' 6 Distancia focal en metres
* Public xpp As Double '5.5 ' 7 Punt Principal x (xpp-(nºpixels/2))en pixels\*micras
* Public ypp As Double '2.1 ' 8 Punt Principal y (ypp-(nºpixels/2))en pixels\*micras
* Public k1 As Double ' = 0.0004534 ' 17 Coeficient radial distorsio 1
* Public k2 As Double '= -0.000001469 ' 18 Coeficient radial distorsio 2
* Public P1 As Double '= -0.00000387 '19 Coeficient tangencial de distorsió
* Public P2 As Double '= 0 '20 Coeficient tangencial de distorsió
* Public psx As Double '= 0.0000056 '15 mida de pixel x (micras/1000000)en metres
* Public psy As Double '= 0.0000056 '16 mida de pixel y (micras/1000000)en metres
* Public m11, m12, m13 As Double
* Public m21, m22, m23 As Double
* Public m31, m32, m33 As Double
* '\*\*\*\*\* Fitxer matrius
* Public ang(8) As Double
* Public tras(2) As Double
* '\*\*\*\*\* Fitxers coordenades 3D
* Public solx() As Double
* Public soly() As Double
* Public solz() As Double
* Public delator() As Integer
* Sub lecturaPIF()
* Dim formatversion As String
* Dim user\_comments As String
* Dim dummy1 As String
* Dim image\_param\_flag As UInt32
* Dim image\_data\_type As UInt32
* Dim invalid\_point As Single
* Dim array\_width As UInt32
* Dim array\_height As UInt32
* Dim data\_block\_length As UInt32
* Dim scale\_flag As UInt32
* Dim i\_scale As Single
* Dim j\_scale As Single
* Dim transfo\_matrix\_flag As UInt32
* Dim transfo\_matrix(16) As Double
* Dim image\_color\_flag As UInt32
* Dim color\_block\_length As UInt32 = 0
* Dim camera\_position\_flag As UInt32
* Dim camera\_x As Single
* Dim camera\_y As Single
* Dim camera\_z As Single
* Dim dummy2 As String
* Dim buffer() As Byte
* On Error GoTo ErrorHandler
* Dim FS As New FileStream(direccioPIF(seles, 0), FileMode.Open, FileAccess.Read, FileShare.Read)
* ReDim buffer(512 - 1)
* Dim lectura1(512 - 1) As Byte
* 'Rotacio
* Dim aa As Double
* Dim bb As Double
* Dim cc As Double
* Dim dd As Double
* Dim ee As Double
* FS.Read(lectura1, 0, 512)
* For i = 200 To 239 Step 4
* buffer(i + 3) = lectura1(i)
* buffer(i + 2) = lectura1(i + 1)
* buffer(i + 1) = lectura1(i + 2)
* buffer(i) = lectura1(i + 3)
* Next
* For i = 240 To 360 Step 8
* buffer(i + 7) = lectura1(i)
* buffer(i + 6) = lectura1(i + 1)
* buffer(i + 5) = lectura1(i + 2)
* buffer(i + 4) = lectura1(i + 3)
* buffer(i + 3) = lectura1(i + 4)
* buffer(i + 2) = lectura1(i + 5)
* buffer(i + 1) = lectura1(i + 6)
* buffer(i) = lectura1(i + 7)
* Next
* For i = 368 To 391 Step 4
* buffer(i + 3) = lectura1(i)
* buffer(i + 2) = lectura1(i + 1)
* buffer(i + 1) = lectura1(i + 2)
* buffer(i) = lectura1(i + 3)
* Next
* formatversion = ByteToString(lectura1, 0, 64)
* user\_comments = ByteToString(lectura1, 64, 128)
* dummy1 = ByteToString(lectura1, 192, 8)
* image\_param\_flag = BitConverter.ToInt32(buffer, 200)
* image\_data\_type = BitConverter.ToInt32(buffer, 204)
* invalid\_point = BitConverter.ToInt32(buffer, 208)
* array\_width = BitConverter.ToInt32(buffer, 212)
* array\_height = BitConverter.ToInt32(buffer, 216)
* data\_block\_length = BitConverter.ToInt32(buffer, 220)
* scale\_flag = BitConverter.ToInt32(buffer, 224)
* i\_scale = BitConverter.ToInt32(buffer, 228)
* j\_scale = BitConverter.ToInt32(buffer, 232)
* transfo\_matrix\_flag = BitConverter.ToInt32(buffer, 236)
* transfo\_matrix(1) = BitConverter.ToDouble(buffer, 240)
* transfo\_matrix(2) = BitConverter.ToDouble(buffer, 248)
* transfo\_matrix(3) = BitConverter.ToDouble(buffer, 256)
* transfo\_matrix(4) = BitConverter.ToDouble(buffer, 264)
* transfo\_matrix(5) = BitConverter.ToDouble(buffer, 272)
* transfo\_matrix(6) = BitConverter.ToDouble(buffer, 280)
* transfo\_matrix(7) = BitConverter.ToDouble(buffer, 288)
* transfo\_matrix(8) = BitConverter.ToDouble(buffer, 296)
* transfo\_matrix(9) = BitConverter.ToDouble(buffer, 304)
* transfo\_matrix(10) = BitConverter.ToDouble(buffer, 312)
* transfo\_matrix(11) = BitConverter.ToDouble(buffer, 320)
* transfo\_matrix(12) = BitConverter.ToDouble(buffer, 328)
* transfo\_matrix(13) = BitConverter.ToDouble(buffer, 336)
* transfo\_matrix(14) = BitConverter.ToDouble(buffer, 344)
* transfo\_matrix(15) = BitConverter.ToDouble(buffer, 352)
* transfo\_matrix(16) = BitConverter.ToDouble(buffer, 360)
* image\_color\_flag = BitConverter.ToInt32(buffer, 368)
* color\_block\_length = BitConverter.ToInt32(buffer, 372)
* camera\_position\_flag = BitConverter.ToInt32(buffer, 376)
* camera\_x = BitConverter.ToInt32(buffer, 380)
* camera\_y = BitConverter.ToInt32(buffer, 384)
* camera\_z = BitConverter.ToInt32(buffer, 388)
* dummy2 = ByteToString(lectura1, 392, 120)
* ' Block\_data\_3D
* Dim lectura2(data\_block\_length) As Byte
* ReDim buffer(data\_block\_length)
* FS.Read(lectura2, 0, data\_block\_length)
* FS.Close()
* For i = 0 To data\_block\_length - 1 Step 4
* buffer(i + 3) = lectura2(i)
* buffer(i + 2) = lectura2(i + 1)
* buffer(i + 1) = lectura2(i + 2)
* buffer(i) = lectura2(i + 3)
* Next
* asa = 0
* Array.Resize(resx, 0)
* Array.Resize(resy, 0)
* Array.Resize(pointx, 0)
* Array.Resize(pointy, 0)
* Array.Resize(pointz, 0)
* ReDim pointx(color\_block\_length)
* ReDim pointy(color\_block\_length)
* ReDim pointz(color\_block\_length)
* ReDim resx(color\_block\_length)
* ReDim resy(color\_block\_length)
* For i = 0 To data\_block\_length - 1 Step 12
* 'input de coordenades pif 3D
* pointx(asa) = BitConverter.ToSingle(buffer, i)
* pointz(asa) = BitConverter.ToSingle(buffer, i + 4)
* pointy(asa) = BitConverter.ToSingle(buffer, i + 8)
* 'conversion a la S.R. camara 2D
* cc = (((pointx(asa) - Xcam) \* m11) + ((pointy(asa) - Ycam) \* m12) + ((pointz(asa) - Zcam) \* m13))
* dd = (((pointx(asa) - Xcam) \* m21) + ((pointy(asa) - Ycam) \* m22) + ((pointz(asa) - Zcam) \* m23))
* ee = (((pointx(asa) - Xcam) \* m31) + ((pointy(asa) - Ycam) \* m32) + ((pointz(asa) - Zcam) \* m33))
* aa = (((cc / ee) \* f) / psx) + (imagewidth / 2)
* bb = imageheight - ((((dd / ee) \* f) / psy) + (imageheight / 2))
* If aa > 0 And bb > 0 Then
* resx(asa) = aa
* resy(asa) = bb
* If resx(asa) > pixmax Then pixmax = resx(asa)
* If resx(asa) < pixmin Then pixmin = resx(asa)
* If resy(asa) > piymax Then piymax = resy(asa)
* If resy(asa) < piymin Then piymin = resy(asa)
* Else
* resx(asa) = Nothing
* resy(asa) = Nothing
* End If
* asa = asa + 1
* Next
* LoadArrayPixels()
* Exit Sub
* ErrorHandler:
* MsgBox("Error on scanner file path -----> " + direccioPIF(seles, 0), MsgBoxStyle.Exclamation, "Warming: Error")
* Form1.Close()
* End Sub
* Sub lecturaCalibracio()
* Dim skew As Integer
* Dim dummy1 As String = ""
* FileClose(1)
* On Error GoTo Errorhandler
* FileOpen(1, direccioCalib(seles, 0), OpenMode.Input)
* Input(1, dummy1)
* Input(1, phi)
* Input(1, kappa)
* Input(1, omega)
* Input(1, Ycam)
* Input(1, Zcam)
* Input(1, Xcam)
* Input(1, f)
* Input(1, xpp)
* Input(1, ypp)
* Input(1, k1)
* Input(1, k2)
* Input(1, psx)
* Input(1, psy)
* Input(1, dummy1)
* Input(1, P1)
* Input(1, P2)
* Input(1, skew)
* f = f / 1000 'distancia focal en metros
* xpp = (xpp - (Image1.Width / 2)) ' Punt Principal x (xpp-(nºpixels/2))en pixels\*micras
* ypp = (ypp - (Image1.Height / 2)) ' Punt Principal y (ypp-(nºpixels/2))en pixels\*micras
* psx = psx / 1000000 '15 mida de pixel x (micras/1000000)en metres
* psy = psy / 1000000 '16 mida de pixel y (micras/1000000)en metres
* FileClose(1)
* ' Calculs angle camera
* omega = omega \* ((2 \* Math.PI) / 360)
* phi = phi \* ((2 \* Math.PI) / 360)
* kappa = kappa \* ((2 \* Math.PI) / 360)
* ' Calculs angle camera
* m11 = Math.Cos(phi) \* Math.Cos(kappa)
* m12 = -Math.Cos(phi) \* Math.Sin(kappa)
* m13 = Math.Sin(phi)
* m21 = (Math.Cos(omega) \* Math.Sin(kappa)) + (Math.Sin(omega) \* Math.Sin(phi) \* Math.Cos(kappa))
* m22 = (Math.Cos(omega) \* Math.Cos(kappa)) - (Math.Sin(omega) \* Math.Sin(phi) \* Math.Sin(kappa))
* m23 = -Math.Sin(omega) \* Math.Cos(phi)
* m31 = (Math.Sin(omega) \* Math.Sin(kappa)) - (Math.Cos(omega) \* Math.Sin(phi) \* Math.Cos(kappa))
* m32 = (Math.Sin(omega) \* Math.Cos(kappa)) + (Math.Cos(omega) \* Math.Sin(phi) \* Math.Sin(kappa))
* m33 = Math.Cos(omega) \* Math.Cos(phi)
* Exit Sub
* Errorhandler:
* MsgBox("1Error on calibration file path -----> " + direccioCalib(seles, 0), MsgBoxStyle.Exclamation, "Warning: Error")
* Form1.Close()
* End Sub
* Sub lecturamatrix()
* ReDim matrius(NumMatrius(seles) - 1, 3, 3)
* Dim dummy1 As String = ""
* On Error GoTo ErrorHandler
* For i = 0 To NumMatrius(seles) - 1
* FileOpen(1, direcciomatrius(seles, i), OpenMode.Input)
* Input(1, dummy1)
* Input(1, dummy1)
* Input(1, ang(0))
* matrius(i, 0, 0) = ang(0)
* Input(1, ang(1))
* matrius(i, 0, 1) = ang(1)
* Input(1, ang(2))
* matrius(i, 0, 2) = ang(2)
* Input(1, tras(0))
* matrius(i, 0, 3) = tras(0)
* Input(1, ang(3))
* matrius(i, 1, 0) = ang(3)
* Input(1, ang(4))
* matrius(i, 1, 1) = ang(4)
* Input(1, ang(5))
* matrius(i, 1, 2) = ang(5)
* Input(1, tras(1))
* matrius(i, 1, 3) = tras(1)
* Input(1, ang(6))
* matrius(i, 2, 0) = ang(6)
* Input(1, ang(7))
* matrius(i, 2, 1) = ang(7)
* Input(1, ang(8))
* matrius(i, 2, 2) = ang(8)
* Input(1, tras(2))
* matrius(i, 2, 3) = tras(2)
* matrius(i, 3, 0) = 0
* matrius(i, 3, 1) = 0
* matrius(i, 3, 2) = 0
* matrius(i, 3, 3) = 1
* FileClose(1)
* rotation()
* Next
* Exit Sub
* ErrorHandler:
* MsgBox("Error on matrix files path -----> " + direcciomatrius(seles, 0), MsgBoxStyle.Exclamation, "Warming: Error")
* Form1.Close()
* End Sub
* Sub lecturaFichResPix()
* Dim position1 As Integer
* Dim position2 As Integer
* Dim infoReader As System.IO.FileInfo
* FileClose(3)
* Form1.ListBox1.Items.Clear()
* On Error GoTo ErrorHandler
* FileOpen(3, DireccioFichResPix(seles), OpenMode.Input)
* infoReader = My.Computer.FileSystem.GetFileInfo(DireccioFichResPix(seles))
* coordenadeslinia.Clear()
* contadorobjecte = -1
* 'numeroregistres = 0
* If infoReader.Length <> 0 Then
* numeroregistres = 0
* contadorobjecte = -1
* Dim dummy1 As Integer
* Dim dum2 As String = Nothing
* Dim dummy3 As Double
* Dim dummy4 As Double
* ncontadorzselec1 = 0
* Do Until EOF(3)
* Input(3, dummy1)
* Select Case dummy1
* Case 5, 6, 7 'inici de línia
* ReDim Preserve codi(numeroregistres)
* ReDim Preserve contadorlineas(numeroregistres)
* ReDim Preserve textcolorline(numeroregistres)
* codi(numeroregistres) = dummy1
* Input(3, dummy3)
* Input(3, dummy4)
* coordenadeslinia.Add(New PointF(dummy3, dummy4))
* Input(3, contadorlineas(numeroregistres))
* Input(3, numeroregistres)
* Input(3, textcolorline(numeroregistres))
* numeroregistres += 1
* Case 8 ' tancament
* ReDim Preserve codi(numeroregistres)
* ReDim Preserve contadorlineas(numeroregistres)
* contadorobjecte += 1
* ReDim Preserve delator(contadorobjecte)
* ReDim Preserve nameobj(contadorobjecte)
* ReDim Preserve namefamily(contadorobjecte)
* delator(contadorobjecte) = numeroregistres
* codi(numeroregistres) = dummy1
* Input(3, contadorlineas(numeroregistres))
* Input(3, numeroregistres)
* Input(3, dum2)
* position1 = InStrRev(dum2, " ")
* position2 = InStrRev(dum2, " ", position1 - 1)
* nameobj(contadorobjecte) = dum2.Substring(0, position2 - 2)
* namefamily(contadorobjecte) = dum2.Substring(position2, dum2.Length - position2)
* numeroregistres += 1
* If EstatForm4 = True Then
* Select Case iform4
* Case 0
* Form4.ListBox1.Items.Add(nameobj(contadorobjecte))
* ReDim Preserve color1(contadorobjecte)
* color1(contadorobjecte) = textcolorline(numeroregistres - 2)
* Case 1
* Form4.ListBox2.Items.Add(nameobj(contadorobjecte))
* Case 2
* Form4.ListBox3.Items.Add(nameobj(contadorobjecte))
* End Select
* Else
* Form1.ListBox1.Items.Add(nameobj(contadorobjecte))
* End If
* ' Form1.ListBox1.Items.Add(nameobj(contadorobjecte))
* coordenadeslinia.Add(New PointF(0, 0))
* End Select
* Loop
* numeroregistres = numeroregistres - 1
* ncontadorzselec2 = contadorobjecte
* End If
* FileClose(3)
* Exit Sub
* ErrorHandler:
* MsgBox("Error on Pixel Coordinates file path -----> " + DireccioFichResPix(seles), MsgBoxStyle.Exclamation, "Warming: Error")
* Form1.Close()
* End Sub
* Sub lecturaFichcoord()
* Dim infoReader As System.IO.FileInfo
* Dim valor As Integer = -1
* Dim valor2 As Integer = -1
* FileOpen(3, DireccioFichCoord(seles), OpenMode.Input)
* infoReader = My.Computer.FileSystem.GetFileInfo(DireccioFichCoord(seles))
* If infoReader.Length <> 0 Then
* contadorz = -1
* Do Until EOF(3)
* contadorz = contadorz + 1
* ReDim Preserve solx(contadorz)
* ReDim Preserve soly(contadorz)
* ReDim Preserve solz(contadorz)
* ReDim Preserve contadorxyz(contadorz)
* Input(3, solx(contadorz))
* Input(3, soly(contadorz))
* Input(3, solz(contadorz))
* Input(3, contadorxyz(contadorz))
* Input(3, contadorz)
* If contadorxyz(contadorz) = 1 Or contadorxyz(contadorz) = 0 Then
* valor += 1
* ReDim Preserve numeroinici(valor)
* numeroinici(valor) = contadorz
* If contadorxyz(contadorz) = 0 Then
* valor2 += 1
* ReDim Preserve numerofinal(valor2)
* numerofinal(valor2) = contadorz
* End If
* End If
* Loop
* calculateparameters()
* End If
* Calculatepos()
* If EstatForm4 = True Then 'Operacio Form4
* Select Case iform4
* Case 0
* ReDim Preserve asolxform4(contadorz)
* ReDim Preserve asolyform4(contadorz)
* ReDim Preserve asolzform4(contadorz)
* ReDim Preserve acontadorxyzform4(contadorz)
* ReDim traza1(contadorobjecte)
* solx.CopyTo(asolxform4, 0)
* soly.CopyTo(asolyform4, 0)
* solz.CopyTo(asolzform4, 0)
* contadorxyz.CopyTo(acontadorxyzform4, 0)
* acontadorz = contadorz
* trazelength.CopyTo(traza1, 0)
* Array.Resize(trazelength, 0)
* Case 1
* ReDim Preserve bsolxform4(contadorz)
* ReDim Preserve bsolyform4(contadorz)
* ReDim Preserve bsolzform4(contadorz)
* ReDim Preserve bcontadorxyzform4(contadorz)
* ReDim traza2(contadorobjecte)
* solx.CopyTo(bsolxform4, 0)
* soly.CopyTo(bsolyform4, 0)
* solz.CopyTo(bsolzform4, 0)
* contadorxyz.CopyTo(bcontadorxyzform4, 0)
* bcontadorz = contadorz
* trazelength.CopyTo(traza2, 0)
* Array.Resize(trazelength, 0)
* Case 2
* ReDim Preserve csolxform4(contadorz)
* ReDim Preserve csolyform4(contadorz)
* ReDim Preserve csolzform4(contadorz)
* ReDim Preserve ccontadorxyzform4(contadorz)
* ReDim traza3(contadorobjecte)
* solx.CopyTo(csolxform4, 0)
* soly.CopyTo(csolyform4, 0)
* solz.CopyTo(csolzform4, 0)
* contadorxyz.CopyTo(ccontadorxyzform4, 0)
* ccontadorz = contadorz
* trazelength.CopyTo(traza3, 0)
* Array.Resize(trazelength, 0)
* End Select
* End If
* FileClose(3)
* FileOpen(3, DireccioFichResPix(seles), OpenMode.Append)
* FileClose(4)
* FileOpen(4, DireccioFichCoord(seles), OpenMode.Append)
* Exit Sub
* ErrorHandler:
* MsgBox("Error on Ground Coordinates file path -----> " + DireccioFichCoord(seles), MsgBoxStyle.Exclamation, "Warming: Error")
* Form1.Close()
* End Sub
* Sub lecturaCalibracioPhoto()
* Dim skew As Integer
* Dim dummy1 As String = ""
* ' Dim dummy2 As String = ""
* ' Dim inputRecord As String = Nothing
* Dim myPoints() As String
* Dim LastNonEmpty As Integer = -1
* ' Dim inReader As StreamReader = File.OpenText(direccioCalib(seles, 0))
* Dim xmldoc As New XmlDataDocument()
* Dim xmlnode As XmlNodeList
* Dim fs As New FileStream(direccioCalib(seles, 0), FileMode.Open, FileAccess.Read)
* xmldoc.Load(fs)
* xmlnode = xmldoc.GetElementsByTagName("structure")
* f = CDbl(xmlnode(0).ChildNodes(seles).ChildNodes(0).ChildNodes.Item(0).InnerText)
* xpp = CDbl(xmlnode(0).ChildNodes(seles).ChildNodes(0).ChildNodes.Item(2).InnerText)
* ypp = CDbl(xmlnode(0).ChildNodes(seles).ChildNodes(0).ChildNodes.Item(3).InnerText)
* skew = CDbl(xmlnode(0).ChildNodes(seles).ChildNodes(0).ChildNodes.Item(4).InnerText)
* k1 = CDbl(xmlnode(0).ChildNodes(seles).ChildNodes(0).ChildNodes.Item(5).InnerText)
* k2 = CDbl(xmlnode(0).ChildNodes(seles).ChildNodes(0).ChildNodes.Item(6).InnerText)
* P1 = CDbl(xmlnode(0).ChildNodes(seles).ChildNodes(0).ChildNodes.Item(7).InnerText)
* dummy1 = xmlnode(0).ChildNodes(seles).ChildNodes(1).ChildNodes(0).InnerText
* myPoints = dummy1.Split(" ")
* For ii As Integer = 0 To myPoints.Length - 1
* If myPoints(ii) <> " " Then
* LastNonEmpty += 1
* myPoints(LastNonEmpty) = myPoints(ii)
* End If
* Next ii
* m11 = CDbl(myPoints(0))
* m21 = CDbl(myPoints(1))
* m31 = CDbl(myPoints(2))
* Xcam = CDbl(myPoints(3))
* m12 = CDbl(myPoints(4))
* m22 = CDbl(myPoints(5))
* m32 = CDbl(myPoints(6))
* Ycam = CDbl(myPoints(7))
* m13 = CDbl(myPoints(8))
* m23 = CDbl(myPoints(9))
* m33 = CDbl(myPoints(10))
* Zcam = CDbl(myPoints(11))
* FileClose(1)
* k1 = k1 / (Math.Pow(f, 2))
* k2 = k2 / (Math.Pow(f, 4))
* f = (psx \* f) / 1000
* f = f / 1000 'distancia focal en metros
* xpp = (xpp - (Image1.Width / 2)) ' Punt Principal x (xpp-(nºpixels/2))en pixels\*micras
* ypp = (ypp - (Image1.Height / 2)) ' Punt Principal y (ypp-(nºpixels/2))en pixels\*micras
* psx = psx / 1000000 '15 mida de pixel x (micras/1000000)en metres
* psy = psy / 1000000 '16 mida de pixel y (micras/1000000)en metres
* FileClose(1)
* ' Calculs angle camera
* ' omega = omega \* ((2 \* Math.PI) / 360)
* ' phi = phi \* ((2 \* Math.PI) / 360)
* ' kappa = kappa \* ((2 \* Math.PI) / 360)
* ' Calculs angle camera
* ' m11 = Math.Cos(phi) \* Math.Cos(kappa)
* ' m21 = -Math.Cos(phi) \* Math.Sin(kappa)
* ' m31 = Math.Sin(phi)
* ' m12 = (Math.Cos(omega) \* Math.Sin(kappa)) + (Math.Sin(omega) \* Math.Sin(phi) \* Math.Cos(kappa))
* ' m22 = (Math.Cos(omega) \* Math.Cos(kappa)) - (Math.Sin(omega) \* Math.Sin(phi) \* Math.Sin(kappa))
* ' m32 = -Math.Sin(omega) \* Math.Cos(phi)
* ' m13 = (Math.Sin(omega) \* Math.Sin(kappa)) - (Math.Cos(omega) \* Math.Sin(phi) \* Math.Cos(kappa))
* ' m23 = (Math.Sin(omega) \* Math.Cos(kappa)) + (Math.Cos(omega) \* Math.Sin(phi) \* Math.Sin(kappa))
* ' m33 = Math.Cos(omega) \* Math.Cos(phi)
* Dim infoReader As System.IO.FileInfo
* Dim dummy2 As String = ""
* Dim dummy3 As String = ""
* Dim position1 As Integer
* Dim position2 As Integer
* 'Dim position3 As String
* On Error GoTo ErrorHandler
* FileOpen(1, direccioCalib(seles, 1), OpenMode.Input)
* infoReader = My.Computer.FileSystem.GetFileInfo(direccioCalib(seles, 1))
* If infoReader.Length <> 0 Then
* Input(1, dummy1)
* ' position1 = InStrRev(dummy1, " ")
* ' position2 = InStrRev(dummy1, " ", position1 - 1)
* position1 = CDec(dummy1.Substring(11, 1))
* dummy1 = LineInput(1)
* For i = 0 To seles
* dummy1 = LineInput(1)
* position2 = CDec(InStrRev(dummy1, " "))
* m33 = -CDec(dummy1.Substring(position2, dummy1.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* m32 = -CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* m31 = -CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* m23 = -CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* m22 = -CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* m21 = -CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* m13 = CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* m12 = CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* m11 = CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* dummy3 = CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* dummy3 = CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* dummy3 = CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* Zcam = CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* Ycam = CDec(dummy2.Substring(position2, dummy2.Length - position2))
* dummy2 = Left(dummy1, position2 - 1)
* position2 = CDec(InStrRev(dummy2, " "))
* Xcam = CDec(dummy2.Substring(position2, dummy2.Length - position2))
* Next
* End If
* Exit Sub
* ErrorHandler:
* MsgBox("Error on Photo Calibration file path -----> " + direccioCalib(seles, 1), MsgBoxStyle.Exclamation, "Warming: Error")
* Form1.Close()
* End Sub
* Sub lecturamdt()
* Dim dummy1 As Integer
* 'Dim midapixel As Double
* ' Dim oripixelx As Double
* ' Dim oripixely As Double
* ' Dim returnvalue As Color
* Dim segonadireccio As String
* Dim primeradireccio As String
* Dim aaa As Integer
* Dim bbb As Integer
* If projec = 3 Then
* segonadireccio = Right(ResponseStream, 3)
* primeradireccio = Left(ResponseStream, Len(ResponseStream) - 3)
* If segonadireccio = "jpg" Then
* segonadireccio = primeradireccio + "jgw"
* Else
* segonadireccio = primeradireccio + "tfw"
* End If
* On Error GoTo ErrorHandler
* FileOpen(1, segonadireccio, OpenMode.Input)
* Input(1, midapixel)
* Input(1, dummy1)
* Input(1, dummy1)
* Input(1, dummy1)
* Input(1, oripixelx)
* Input(1, oripixely)
* End If
* Dim numeropunts As Integer
* Dim aa As Double
* Dim bb As Double
* Dim cc As Double
* Dim dd As Double
* Dim ee As Double
* Dim inputRecord As String = Nothing
* Dim myPoints() As String
* Dim LastNonEmpty As Integer = -1
* Dim imReader As StreamReader = File.OpenText(direccioPIF(seles, 0))
* numeropunts = 0
* inputRecord = imReader.ReadLine()
* While (inputRecord IsNot Nothing)
* numeropunts += 1
* inputRecord = imReader.ReadLine()
* End While
* imReader.Close()
* ReDim Preserve pointx(numeropunts)
* ReDim Preserve pointy(numeropunts)
* ReDim Preserve pointz(numeropunts)
* ReDim Preserve resx(numeropunts)
* ReDim Preserve resy(numeropunts)
* Dim imReader1 As StreamReader = File.OpenText(direccioPIF(seles, 0))
* asa = -1
* inputRecord = imReader1.ReadLine()
* While (inputRecord IsNot Nothing)
* asa += 1
* 'ReDim Preserve pointx(asa)
* 'ReDim Preserve pointy(asa)
* 'ReDim Preserve pointz(asa)
* 'ReDim Preserve resx(asa)
* 'ReDim Preserve resy(asa)
* If inputRecord.Contains(" ") Then
* myPoints = inputRecord.Split
* ' For i As Integer = 0 To myPoints.Length - 1
* ' If myPoints(i) <> "" Then
* ' LastNonEmpty += 1
* ' myPoints(LastNonEmpty) = myPoints(i)
* ' End If
* ' LastNonEmpty = -1
* ' Next i
* pointx(asa) = myPoints(0)
* pointy(asa) = myPoints(1)
* pointz(asa) = myPoints(2)
* End If
* inputRecord = imReader1.ReadLine()
* ' ReDim Preserve dummy
* Select Case projec
* Case 2
* 'conversion a la S.R. camara 2D
* cc = (((pointx(asa) - Xcam) \* m11) + ((pointy(asa) - Ycam) \* m12) + ((pointz(asa) - Zcam) \* m13))
* dd = (((pointx(asa) - Xcam) \* m21) + ((pointy(asa) - Ycam) \* m22) + ((pointz(asa) - Zcam) \* m23))
* ee = (((pointx(asa) - Xcam) \* m31) + ((pointy(asa) - Ycam) \* m32) + ((pointz(asa) - Zcam) \* m33))
* aa = (((cc / ee) \* f) / psx) + (imagewidth / 2)
* bb = ((((dd / ee) \* f) / psy) + (imageheight / 2))
* If aa > 0 And bb > 0 Then
* resx(asa) = aa - 424
* resy(asa) = bb + 27
* If resx(asa) > pixmax Then pixmax = resx(asa)
* If resx(asa) < pixmin Then pixmin = resx(asa)
* If resy(asa) > piymax Then piymax = resy(asa)
* If resy(asa) < piymin Then piymin = resy(asa)
* Else
* resx(asa) = Nothing
* resy(asa) = Nothing
* End If
* Case 3
* aaa = CDec((pointx(asa) - oripixelx) / midapixel)
* bbb = Math.Abs((CDec(pointy(asa) - oripixely)) / midapixel)
* ' If aaa <= Image1.Width And bbb <= Image1.Height Then
* If pointx(asa) > oripixelx And pointx(asa) < (oripixelx + (Image1.Width \* midapixel)) Then
* If pointy(asa) > (oripixely - (Image1.Height \* midapixel)) And pointy(asa) < oripixely Then
* If aaa < Image1.Width And bbb < Image1.Height Then
* If aaa > 0 And bbb > 0 Then
* resx(asa) = aaa
* resy(asa) = bbb
* If resx(asa) > pixmax Then pixmax = resx(asa)
* If resx(asa) < pixmin Then pixmin = resx(asa)
* If resy(asa) > piymax Then piymax = resy(asa)
* If resy(asa) < piymin Then piymin = resy(asa)
* Else
* resx(asa) = Nothing
* resy(asa) = Nothing
* End If
* ' inputRecord = imReader1.ReadLine()
* Else
* ' inputRecord = imReader1.ReadLine()
* End If
* End If
* End If
* End Select
* End While
* imReader.Close()
* LoadArrayPixels()
* Exit Sub
* ErrorHandler:
* MsgBox("Error on GeoImage file path -----> " + segonadireccio, MsgBoxStyle.Exclamation, "Warming: Error")
* Form1.Close()
* End Sub
* Sub lecturatxt()
* Array.Resize(resx, 0)
* Array.Resize(resy, 0)
* Array.Resize(pointx, 0)
* Array.Resize(pointy, 0)
* Array.Resize(pointz, 0)
* Dim numeropunts As Integer
* 'Rotacio
* Dim aa As Double
* Dim bb As Double
* Dim cc As Double
* Dim dd As Double
* Dim ee As Double
* Dim inputRecord As String = Nothing
* Dim myPoints() As String
* Dim LastNonEmpty As Integer = -1
* Dim imReader As StreamReader = File.OpenText(direccioPIF(seles, 0))
* numeropunts = 0
* inputRecord = imReader.ReadLine()
* While (inputRecord IsNot Nothing)
* asa += 1
* inputRecord = imReader.ReadLine()
* End While
* imReader.Close()
* ReDim Preserve pointx(asa)
* ReDim Preserve pointy(asa)
* ReDim Preserve pointz(asa)
* ReDim Preserve resx(asa)
* ReDim Preserve resy(asa)
* Dim imReader1 As StreamReader = File.OpenText(direccioPIF(seles, 0))
* asa = -1
* inputRecord = imReader1.ReadLine()
* While (inputRecord IsNot Nothing)
* asa += 1
* If inputRecord.Contains(" ") Then
* myPoints = inputRecord.Split
* pointx(asa) = myPoints(0)
* pointy(asa) = myPoints(1)
* pointz(asa) = myPoints(2)
* End If
* inputRecord = imReader1.ReadLine()
* 'conversion a la S.R. camara 2D
* cc = (((pointx(asa) - Xcam) \* m11) + ((pointy(asa) - Ycam) \* m12) + ((pointz(asa) - Zcam) \* m13))
* dd = (((pointx(asa) - Xcam) \* m21) + ((pointy(asa) - Ycam) \* m22) + ((pointz(asa) - Zcam) \* m23))
* ee = (((pointx(asa) - Xcam) \* m31) + ((pointy(asa) - Ycam) \* m32) + ((pointz(asa) - Zcam) \* m33))
* aa = (((cc / ee) \* f) / psx) + (imagewidth / 2)
* bb = imageheight - ((((dd / ee) \* f) / psy) + (imageheight / 2))
* If aa > 0 And bb > 0 Then
* resx(asa) = aa
* resy(asa) = bb
* If resx(asa) > pixmax Then pixmax = resx(asa)
* If resx(asa) < pixmin Then pixmin = resx(asa)
* If resy(asa) > piymax Then piymax = resy(asa)
* If resy(asa) < piymin Then piymin = resy(asa)
* Else
* resx(asa) = Nothing
* resy(asa) = Nothing
* End If
* ' asa += 1
* End While
* FileClose(2)
* LoadArrayPixels()
* Exit Sub
* ErrorHandler:
* MsgBox("Error on scan txt file path -----> " + direccioPIF(seles, 0), MsgBoxStyle.Exclamation, "Warming: Error")
* Form1.Close()
* End Sub
* Private Function ByteToString(ByVal Expression() As Byte, ByVal StartIndex As Integer, ByVal Length As Integer) As String
* Return System.Text.Encoding.Default.GetString(Expression, StartIndex, Length)
* End Function
* End Module