```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: MDIProject.cpp

#include <vcl.h>
#pragma hdrstop

USERES("MDIProject.res");
USEFORM("ParentUnit.cpp", ParentForm);
USEFORM("TextChildUnit.cpp", TextChildForm);
USEFORM("ImageChildUnit.cpp", ImageChildForm);
USEFORM("Histogram.cpp", frmHistogram);
USEFORM("TopologicalView.cpp", frmTopologicalView);
USEFORM("hvProjection.cpp", frmHVP);

//---------------------------------------------------------------------------

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
  try
  {
    Application->Initialize();
    Application->CreateForm(__classid(TParentForm), &ParentForm);
    Application->CreateForm(__classid(TfrmHistogram), &frmHistogram);
    Application->CreateForm(__classid(TfrmTopologicalView), &frmTopologicalView);
    Application->CreateForm(__classid(TfrmHVP), &frmHVP);
    Application->Run();
  }
  catch (Exception &exception)
  {
    Application->ShowException(&exception);
  }
  return 0;
}
```

```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: Histogram.cpp

#include <vcl.h>
#pragma hdrstop

#include "Histogram.h"
#include "ImageChildUnit.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmHistogram *frmHistogram;
//---------------------------------------------------------------------------
__fastcall TfrmHistogram::TfrmHistogram(TComponent* Owner)
  : TForm(Owner)
{
  this->Caption = "Histogram View";
  this->Width = 255;
  this->Height = 280;
  this->Color = clWhite;

  THRESHOLD_VALUE = 255 / 2;
}
//---------------------------------------------------------------------------

void __fastcall TfrmHistogram::FormClose(TObject *Sender,
      TCloseAction &Action)
{
   Action = caFree;
}

void __fastcall TfrmHistogram::Histogram( unsigned int Histogram[] )
{
  // code here ...
  for( int i=0; i<256; i++ )
  {
    this->Canvas->Pen->Color = RGB( 0, 0, 0 );
    this->Canvas->MoveTo( i, 250 );
    this->Canvas->LineTo( i, 250 - Histogram[i]/4 );
  }
}

void __fastcall TfrmHistogram::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState
 Shift, int X, int Y)
{
  THRESHOLD_VALUE = X;  // Set the threshold value to be the position of the mouse clicked!
}

void __fastcall TfrmHistogram::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  this->Caption = "Location: ";
  this->Caption = this->Caption + "[" + X + "," + Y + "]";
}

void __fastcall TfrmHistogram::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState
 Shift, int X, int Y)
{
  // code here ...
}

// Return the Threshold Value;
int  __fastcall TfrmHistogram::Threshold( )
{
  return THRESHOLD_VALUE;
}
```

```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: Histogram.h

#ifndef HistogramH
#define HistogramH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TfrmHistogram : public TForm
{
__published:  // IDE-managed Components

  void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
  void __fastcall FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X,
 int Y);
  void __fastcall FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y);
  void __fastcall FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift, int X,
 int Y);

private:  // User declarations

  int  THRESHOLD_VALUE;

public:   // User declarations

  __fastcall TfrmHistogram(TComponent* Owner);

  void __fastcall Histogram( unsigned int Histogram[] );
  int  __fastcall Threshold( );

};
//---------------------------------------------------------------------------
extern PACKAGE TfrmHistogram *frmHistogram;
//---------------------------------------------------------------------------
#endif
```

```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: hvProjection.cpp

#include <vcl.h>
#pragma hdrstop

#include "hvProjection.h"
#include "ImageChildUnit.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmHVP *frmHVP;
//---------------------------------------------------------------------------
__fastcall TfrmHVP::TfrmHVP(TComponent* Owner)
  : TForm(Owner)
{
  this->Height = 300;
  this->Width = 550;
}
//---------------------------------------------------------------------------

void __fastcall TfrmHVP::HVP( unsigned int H[], unsigned int V[], int w, int h )
{
  for( int i=0; i<h; i++ )
  {
    imageH->Canvas->Pen->Color = RGB( 0, 0, 0 );
    imageH->Canvas->MoveTo( i, imageV->Width );
    imageH->Canvas->LineTo( i, imageV->Width - H[i] );
  }

  for( int i=0; i<w; i++ )
  {
    imageV->Canvas->Pen->Color = RGB( 0, 0, 0 );
    imageV->Canvas->MoveTo( 0, i );
    imageV->Canvas->LineTo( V[i], i );
  }
}
```

```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: hvProjection.h

#ifndef hvProjectionH
#define hvProjectionH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TfrmHVP : public TForm
{
__published:  // IDE-managed Components
  TImage *imageH;
  TImage *imageV;
  TLabel *Label1;
  TLabel *Label2;
private:  // User declarations
public:   // User declarations
  __fastcall TfrmHVP(TComponent* Owner);

  void __fastcall HVP( unsigned int H[], unsigned int V[], int w, int h );
};
//---------------------------------------------------------------------------
extern PACKAGE TfrmHVP *frmHVP;
//---------------------------------------------------------------------------
#endif
```

```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: ImageChildUnit.cpp

#include <vcl.h>
#pragma hdrstop

#include "ImageChildUnit.h"
#include "ParentUnit.h"
#include "Histogram.h"
#include "TopologicalView.h"
#include "hvProjection.h"

#include <jpeg.hpp>  // for JPEG support :-)

//---------------------------------------------------------------------------//
#pragma package(smart_init)
#pragma resource "*.dfm"

int Lowpass(BYTE Image[][256], int x, int y)
{
        int Mask[3][3] = {{1, 1, 1,}, {1, 1, 1}, {1, 1, 1}};
        int i, j, sum=0;

        for (j=-1; j<=1; j++)
                for (i=-1; i<=1; i++)
                      sum += Image[x+i][y+j]*Mask[i+1][j+1];
        return INT(sum/9.0);
}

int Edge(BYTE Image[][256], int x, int y)
{
        int Mask[3][3] = {{-1, -1, -1,}, {-1, 9, -1}, {-1, -1, -1}};
        int i, j, sum=0;

        for (j=-1; j<=1; j++)
                for (i=-1; i<=1; i++)
                      sum += Image[x+i][y+j]*Mask[i+1][j+1];
        return INT(sum/9.0);
}

TImageChildForm *ImageChildForm;
//---------------------------------------------------------------------------//
__fastcall TImageChildForm::TImageChildForm(TComponent* Owner)
        : TForm(Owner)
{
  Image->AutoSize = true;
}
//---------------------------------------------------------------------------//

void __fastcall TImageChildForm::FormClose(TObject *Sender,
      TCloseAction &Action)
{
   Action = caFree;
}
//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//

void __fastcall TImageChildForm::Close1Click(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------//

void __fastcall TImageChildForm::SaveAs1Click(TObject *Sender)
{
   if(ParentForm->SavePictureDialog1->Execute())
   {
      AnsiString Extension = ExtractFileExt(ParentForm->SavePictureDialog1->FileName);

      TGraphic* Graphic;
```

```
      try
      {

         if(Extension == ".bmp")
         {
            if(!dynamic_cast<Graphics::TBitmap*>(Image->Picture->Graphic))
            {
               Graphic = new Graphics::TBitmap();
               Graphic->Assign(Image->Picture->Graphic);
            }
            else Graphic = Image->Picture->Graphic;
         }
         else if(Extension == ".ico")
         {
            if(!dynamic_cast<TIcon*>(Image->Picture->Graphic))
            {
               Graphic = new TIcon();
               Graphic->Assign(Image->Picture->Graphic);
            }
            else Graphic = Image->Picture->Graphic;
         }
         else if(Extension == ".wmf" || Extension == ".emf")
         {
            if(!dynamic_cast<TMetafile*>(Image->Picture->Graphic))
            {
               Graphic = new TMetafile();
               Graphic->Assign(Image->Picture->Graphic);
            }
            else Graphic = Image->Picture->Graphic;
         }
         else if(Extension == ".jpg" || Extension == ".jpeg")
         {
            if(!dynamic_cast<TJPEGImage*>(Image->Picture->Graphic))
            {
               Graphic = new TJPEGImage();
               Graphic->Assign(Image->Picture->Graphic);
            }
            else Graphic = Image->Picture->Graphic;
         }
         Graphic->SaveToFile(ParentForm->SavePictureDialog1->FileName);

      }
      __finally
      {
         delete Graphic;
      }
   }
}

//---------------------------------------------------------------------------

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>

//---------------------------------------------------------------------------//

void __fastcall TImageChildForm::GetPixelFormat1Click(TObject *Sender)
{
  // Write it to a file for a test
  ofstream dataFile( "pwgFile.pgm", ios::out );

  dataFile<<"P2"<<endl;
  dataFile<<"#Vahe Karamian"<<endl;
  dataFile<<this->Width<<" "<<this->Height<<endl;
  dataFile<<"255"<<endl;

  for( int i=0; i<this->Height; i++ )
  {
    for( int j=0; j<this->Width; j++ )
    {
```

```
      dataFile<< (WORD) Image->Canvas->Pixels[j][i] << " ";
    }
    dataFile<<endl;
  }
}

//---------------------------------------------------------------------------

void __fastcall TImageChildForm::Histogram1Click(TObject *Sender)
{
  unsigned int histogram[256];
  unsigned int transform[256];


  BYTE *LinePtr;

  int  x, y, i, j, sum;

  int  width  = Image->Picture->Width;
  int  height = Image->Picture->Height;

  if( width > 256 )
    width = 256;
  if( height > 256 )
    height = 256;

  // Copy the image to ImageData
  for( y=0; y<height; y++ )
  {
    LinePtr = (BYTE *) Image->Picture->Bitmap->ScanLine[y];
    for( x=0; x<width; x++ )
      ImageData[x][y] = LinePtr[x];
  }

  // Initialize Histogram
  for( i=0; i<256; i++ )
    histogram[i] = 0;

  // Construct the histogram
  for( y=0; y<height; y++ )
  {
    for( x=0; x<width; x++ )
      histogram[ ImageData[x][y] ]++;
  }

  frmHistogram->Histogram( histogram );

  // Compute the transform
  /*
  for( i=0; i<height; i++ )
  {
    sum = 0;
    for( j=0; j<i; j++ )
    {
      sum += histogram[j];
      transform[i] = INT( 255.0 * sum / (256 * 256) );
    }
  }

  // Transform the image
  for( y=0; y<height; y++ )
  {
    for( x=0; x<width; x++ )
    {
      ImageData[x][y] = transform[ImageData[x][y]];
    }
  }

  // Copy the image back to display
  for( y=0; y<height; y++ )
  {
```

```
    LinePtr = (BYTE *) Image->Picture->Bitmap->ScanLine[y];
    for( x=0; x<width; x++ )
    {
      LinePtr[x] = ImageData[x][y];
    }
  }

  Image->Refresh( );
  */
}
//---------------------------------------------------------------------------

void __fastcall TImageChildForm::TopographicView1Click(TObject *Sender)
{
  BYTE ImageData[256][256];
  BYTE *LinePtr;

  int  x, y, i, j, sum;

  int  width  = Image->Picture->Width;
  int  height = Image->Picture->Height;

  if( width > 256 )
    width = 256;
  if( height > 256 )
    height = 256;

  // Copy the image to ImageData
  for( y=0; y<height; y++ )
  {
    LinePtr = (BYTE *) Image->Picture->Bitmap->ScanLine[y];
    for( x=0; x<width; x++ )
      ImageData[x][y] = LinePtr[x];
  }
  frmTopologicalView->Image( ImageData, width, height );
}
//---------------------------------------------------------------------------

void __fastcall TImageChildForm::VerticalHorizontalProjection1Click(
      TObject *Sender)
{
  unsigned int horizontal[256];
  unsigned int vertical[256];

  BYTE *LinePtr;

  int  x, y, i, j, sum;

  int  width  = Image->Picture->Width;
  int  height = Image->Picture->Height;

  if( width > 256 )
    width = 256;
  if( height > 256 )
    height = 256;


  // Copy the image to ImageData
  for( y=0; y<height; y++ )
  {
    LinePtr = (BYTE *) Image->Picture->Bitmap->ScanLine[y];
    for( x=0; x<width; x++ )
      ImageData[x][y] = LinePtr[x];
  }

  // Initialize horizontal & vertical Projection
  for( i=0; i<256; i++ )
    horizontal[i] = vertical[i] = 0;

  unsigned int rowAverage = 0;
```

```
  // Construct Vertical Projection
  for( y=0; y<height; y++ )
  {
    rowAverage = 0;
    for( x=0; x<width; x++ )
    {
      rowAverage = rowAverage + ImageData[x][y];
    }
    horizontal[ y ] = rowAverage/width;
  }

  unsigned int colAverage = 0;
  // Construct Horizontal Projection
  for( x = 0; x<width; x++ )
  {
    colAverage = 0;
    for( y=0; y<height; y++ )
    {
      colAverage = colAverage + ImageData[x][y];
    }
    vertical[x] = colAverage/height;
  }

 frmHVP->HVP( horizontal, vertical, width, height );
}
//---------------------------------------------------------------------------

void __fastcall TImageChildForm::Threshold1Click(TObject *Sender)
{
  int THRESHOLD = frmHistogram->Threshold( );

  this->Caption = "Threshold is: ";
  this->Caption = this->Caption + THRESHOLD;

  // Enter code here to perform the threshold stuff
  BYTE *linePtr;
  int  width  = Image->Picture->Width;
  int  height = Image->Picture->Height;

  if( width > 256 )
    width = 256;
  if( height > 256 )
    height = 256;

  for( int y=0; y<height; y++ )
  {
    linePtr = (BYTE*) Image->Picture->Bitmap->ScanLine[y];
    for( int x=0; x<width; x++ )
    {
      if( linePtr[x] > THRESHOLD )
        linePtr[x] = 255;
      else
        linePtr[x] = 0;
    }
  }
  Image->Refresh( );
}
//---------------------------------------------------------------------------

void __fastcall TImageChildForm::EdgeDetect1Click(TObject *Sender)
{
  BYTE ImageData[256][256], *linePtr;
  BYTE Output[256][256];

  int  width  = Image->Picture->Width;
  int  height = Image->Picture->Height;

  if( width > 256 )
    width = 256;
  if( height > 256 )
    height = 256;
```

```
  // Copy the image
  for( int y=0; y<height; y++ )
  {
    linePtr = (BYTE*) Image->Picture->Bitmap->ScanLine[y];
    for( int x=0; x< width; x++ )
      ImageData[x][y] = linePtr[x];
  }

  // Calculate the edge image
  for( int y=1; y<height-1; y++ )
    for( int x=1; x<width-1; x++ )
      Output[x][y] = Edge( ImageData, x, y );

  // Copy the image back for display
  for( int y=0; y<height; y++ )
  {
    linePtr = (BYTE*) Image->Picture->Bitmap->ScanLine[y];
    for( int x=0; x<width; x++ )
      linePtr[x] = Output[x][y];
  }
  Image->Refresh( );
}
//---------------------------------------------------------------------------

void __fastcall TImageChildForm::Smoothing1Click(TObject *Sender)
{
  BYTE ImageData[256][256], *linePtr;
  BYTE Output[256][256];

  int  width  = Image->Picture->Width;
  int  height = Image->Picture->Height;

  if( width > 256 )
    width = 256;
  if( height > 256 )
    height = 256;

  // Copy the image
  for( int y=0; y<height; y++ )
  {
    linePtr = (BYTE*) Image->Picture->Bitmap->ScanLine[y];
    for( int x=0; x< width; x++ )
      ImageData[x][y] = linePtr[x];
  }

  // Calculate the edge image
  for( int y=1; y<height-1; y++ )
    for( int x=1; x<width-1; x++ )
      Output[x][y] = Lowpass( ImageData, x, y );

  // Copy the image back for display
  for( int y=0; y<height; y++ )
  {
    linePtr = (BYTE*) Image->Picture->Bitmap->ScanLine[y];
    for( int x=0; x<width; x++ )
      linePtr[x] = Output[x][y];
  }
  Image->Refresh( );

}
//---------------------------------------------------------------------------
```

```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: ImageChildUnit.h

#ifndef ImageChildUnitH
#define ImageChildUnitH
//---------------------------------------------------------------------------//
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Menus.hpp>
//---------------------------------------------------------------------------//
class TImageChildForm : public TForm
{
__published:  // IDE-managed Components
  TScrollBox *ScrollBox1;
  TMainMenu *MainMenu1;
  TMenuItem *File1;
  TMenuItem *N2;
  TMenuItem *Close1;
  TMenuItem *CloseAll1;
  TMenuItem *N3;
  TMenuItem *Exit1;
  TImage *Image;
  TMenuItem *SaveAs1;
  TMenuItem *N4;
  TMenuItem *OpenImage1;
  TMenuItem *Tools1;
  TMenuItem *TopographicView1;
  TMenuItem *Threshold1;
  TMenuItem *EdgeDetect1;
  TMenuItem *VerticalHorizontalProjection1;
  TMenuItem *GetPixelFormat1;
  TMenuItem *N1;
  TMenuItem *Histogram1;
  TMenuItem *Smoothing1;

  void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
  void __fastcall Close1Click(TObject *Sender);
  void __fastcall SaveAs1Click(TObject *Sender);
  void __fastcall GetPixelFormat1Click(TObject *Sender);
  void __fastcall Histogram1Click(TObject *Sender);
  void __fastcall TopographicView1Click(TObject *Sender);
  void __fastcall VerticalHorizontalProjection1Click(TObject *Sender);
  void __fastcall Threshold1Click(TObject *Sender);
  void __fastcall EdgeDetect1Click(TObject *Sender);
  void __fastcall Smoothing1Click(TObject *Sender);

private:  // User declarations

  int  pixelFormat;
  BYTE ImageData[256][256];

public:   // User declarations
        __fastcall TImageChildForm(TComponent* Owner);
};

//---------------------------------------------------------------------------//
extern PACKAGE TImageChildForm *ImageChildForm;
//---------------------------------------------------------------------------//
#endif
```

```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: ParentUnit.cpp

#include <vcl.h>
#pragma hdrstop

#include "ParentUnit.h"
#include "TextChildUnit.h"
#include "ImageChildUnit.h"
#include <jpeg.hpp>  // for JPEG support :-)
#include <memory>    // for auto_ptr<>

//---------------------------------------------------------------------------//

#pragma package(smart_init)
#pragma resource "*.dfm"
TParentForm *ParentForm;

// Constructor
__fastcall TParentForm::TParentForm(TComponent* Owner)
        : TForm(Owner)
{
   this->Caption = "Vahe Karamian - CS 519 - Project 1";
   this->WindowState = wsMaximized;

   NewClientWndProc      = 0;
   OriginalClientWndProc = 0;

   NewClientWndProc = MakeObjectInstance(ClientWndProc);

   OriginalClientWndProc
    = reinterpret_cast<void*>
      ( SetWindowLong(ClientHandle,
                      GWL_WNDPROC,
                      reinterpret_cast<LONG>(NewClientWndProc)) );

}

// Window Procedure for the MDI parent's client window
// Perform Message Handling Here
void __fastcall TParentForm::ClientWndProc(TMessage& Message)
{
   switch(Message.Msg)
   {
      case WM_ERASEBKGND: { break; }

      // refresh the image upon scrolling
      case WM_HSCROLL:
      case WM_VSCROLL:
      {
         InvalidateRect(ClientHandle, NULL, true);
         break;
      }
   }


   // Then call the original WndProc() for the other messages
   if(OriginalClientWndProc != 0)
   {
      Message.Result
       = CallWindowProc(reinterpret_cast<FARPROC>(OriginalClientWndProc),
                        ClientHandle,
                        Message.Msg,
                        Message.WParam,
                        Message.LParam);
   }
}

// Close all child windows
void __fastcall TParentForm::FormClose(TObject *Sender,
      TCloseAction &Action)
```

```
{
   CloseAllChildWindows();

   // Reset the original WndProc() for the client window
   SetWindowLong(ClientHandle,
                 GWL_WNDPROC,
                 reinterpret_cast<LONG>(OriginalClientWndProc));

   OriginalClientWndProc = 0;
   FreeObjectInstance(NewClientWndProc);
}

// Intercept WM_ERASEBKGROUND messages to the form itself and ignore
// them since we are going to handle the painting manually ourselves
void __fastcall TParentForm::WMEraseBkgnd(TWMEraseBkgnd& Message)
{
   Message.Result = true; // We didn't erase the background but we tell Windows we did
}

//    These functions that follow are for general MDI features such as closing
//    all child windows
void __fastcall TParentForm::CloseAllChildWindows()
{
   // Back-to-front because MDIChildCount will be decremented by one each time
   for(int i=MDIChildCount-1; i>=0; --i)
   {
      MDIChildren[i]->Close();
   }
}

void __fastcall TParentForm::OpenImageActionExecute(TObject *Sender)
{
   if(OpenPictureDialog1->Execute())
   {
      AnsiString Extension = ExtractFileExt(OpenPictureDialog1->FileName);


      if( Extension == ".bmp" )
      {
         Application->CreateForm (__classid(TImageChildForm), &ImageChildForm);
         ImageChildForm->Image->Picture->Bitmap->LoadFromFile(OpenPictureDialog1->FileName);
         ImageChildForm->Width = ImageChildForm->Image->Picture->Width;
         ImageChildForm->Height = ImageChildForm->Image->Picture->Height;
         ImageChildForm->Caption = OpenPictureDialog1->FileName;
         ImageChildForm->Show();
      }
      if( Extension == ".ico"
          || Extension == ".wmf"
          || Extension == ".emf"
          || Extension == ".jpg"
          || Extension == ".jpeg" )
      {
         Application->CreateForm (__classid(TImageChildForm), &ImageChildForm);
         ImageChildForm->Image->Picture->LoadFromFile(OpenPictureDialog1->FileName);
         ImageChildForm->Width = ImageChildForm->Image->Picture->Width;
         ImageChildForm->Height = ImageChildForm->Image->Picture->Height;
         ImageChildForm->Caption = OpenPictureDialog1->FileName;
         ImageChildForm->Show();
      }
   }
}

//---------------------------------------------------------------------------//

void __fastcall TParentForm::CloseAllActionExecute(TObject *Sender)
{
   CloseAllChildWindows();
}

//---------------------------------------------------------------------------//
```

```
void __fastcall TParentForm::ExitActionExecute(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------//
```

```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: ParentUnit.h

#ifndef ParentUnitH
#define ParentUnitH

//---------------------------------------------------------------------------//

#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
#include <Dialogs.hpp>
#include <ExtDlgs.hpp>
#include <ActnList.hpp>
#include <ImgList.hpp>
#include <StdActns.hpp>

//---------------------------------------------------------------------------//

class TParentForm : public TForm
{
__published:  // IDE-managed Components
  TMainMenu *MainMenu1;
  TMenuItem *File1;
  TMenuItem *Exit1;
  TMenuItem *Window1;
  TMenuItem *TileHorizontally1;
  TMenuItem *TileVertically1;
  TMenuItem *Cascade1;
  TMenuItem *N2;
  TOpenPictureDialog *OpenPictureDialog1;
  TSavePictureDialog *SavePictureDialog1;
  TColorDialog *ColorDialog1;
  TMenuItem *ArrangeMinimizedWindows1;
  TImageList *ActionListImageList;
  TActionList *ActionList1;
  TWindowTileHorizontal *WindowTileHorizontal1;
  TWindowTileVertical *WindowTileVertical1;
  TWindowCascade *WindowCascade1;
  TWindowArrange *WindowArrange1;
  TOpenDialog *OpenDialog1;
  TMenuItem *OpenImage1;
  TAction *OpenAction;
  TAction *OpenImageAction;
  TAction *CloseAction;
  TAction *CloseAllAction;
  TAction *ExitAction;
  TAction *SaveAction;
  TAction *SaveAsAction;

  void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
  void __fastcall OpenImageActionExecute(TObject *Sender);
  void __fastcall CloseAllActionExecute(TObject *Sender);
  void __fastcall ExitActionExecute(TObject *Sender);

private:  // User declarations

   //---------------------------------------------------------------------------//
   // Window Procedure Functions and Variables ------------------------------//

   void* OriginalClientWndProc;
   void* NewClientWndProc;
   void __fastcall ClientWndProc(TMessage& Message);
   void __fastcall WMEraseBkgnd(TWMEraseBkgnd& Message);

public:   // User declarations
```

```
   __fastcall TParentForm(TComponent* Owner);

   // MESSAGE_MAP for WM_ERASEBKGND
   BEGIN_MESSAGE_MAP
   VCL_MESSAGE_HANDLER(WM_ERASEBKGND, TWMEraseBkgnd, WMEraseBkgnd)
   END_MESSAGE_MAP(TForm)

   // General MDI Helper Functions
   void __fastcall CloseAllChildWindows();
};
//---------------------------------------------------------------------------//
extern PACKAGE TParentForm *ParentForm;
//---------------------------------------------------------------------------//
#endif
```

```
//---------------------------------------------------------------------------//
#include <vcl.h>
#pragma hdrstop

#include "TextChildUnit.h"
#include "ParentUnit.h"
//---------------------------------------------------------------------------//
#pragma package(smart_init)
#pragma resource "*.dfm"
TTextChildForm *TextChildForm;
//---------------------------------------------------------------------------//
__fastcall TTextChildForm::TTextChildForm(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------////

void __fastcall TTextChildForm::Close1Click(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------//

void __fastcall TTextChildForm::FormClose(TObject *Sender,
      TCloseAction &Action)
{
   Action = caFree;
}
//---------------------------------------------------------------------------//

void __fastcall TTextChildForm::Save1Click(TObject *Sender)
{
   RichEdit1->Lines->SaveToFile(Caption);
}
//---------------------------------------------------------------------------//

void __fastcall TTextChildForm::SaveAll1Click(TObject *Sender)
{
   if(SaveDialog1->Execute())
   {
      AnsiString Extension = ExtractFileExt(SaveDialog1->FileName);

      if(Extension == ".rtf") RichEdit1->PlainText = false;
      else RichEdit1->PlainText = true;

      RichEdit1->Lines->SaveToFile(SaveDialog1->FileName);
      Caption = SaveDialog1->FileName;
   }
}
//---------------------------------------------------------------------------//
```

```
//---------------------------------------------------------------------------

#ifndef TextChildUnitH
#define TextChildUnitH
//---------------------------------------------------------------------------//
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include <ActnList.hpp>
#include <ComCtrls.hpp>
#include <ImgList.hpp>
#include <StdActns.hpp>
#include <Dialogs.hpp>
//---------------------------------------------------------------------------//
class TTextChildForm : public TForm
{
__published:  // IDE-managed Components
        TMainMenu *MainMenu1;
        TMenuItem *File1;
        TMenuItem *Open1;
        TMenuItem *N2;
        TMenuItem *Close1;
        TMenuItem *CloseAll1;
        TMenuItem *N3;
        TMenuItem *Exit1;
        TRichEdit *RichEdit1;
        TMenuItem *Edit1;
        TActionList *ActionList1;
        TImageList *ImageList1;
        TMenuItem *Cut1;
        TMenuItem *Copy1;
        TMenuItem *Paste1;
        TMenuItem *SelectAll1;
        TMenuItem *Delete1;
        TEditCut *EditCut1;
        TEditCopy *EditCopy1;
        TEditPaste *EditPaste1;
        TEditSelectAll *EditSelectAll1;
        TEditDelete *EditDelete1;
        TMenuItem *Undo1;
        TMenuItem *N4;
        TEditUndo *EditUndo1;
        TMenuItem *Save1;
        TMenuItem *SaveAll1;
        TMenuItem *N5;
        TMenuItem *OpenImage1;
        TSaveDialog *SaveDialog1;
        void __fastcall Close1Click(TObject *Sender);
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
        void __fastcall Save1Click(TObject *Sender);
        void __fastcall SaveAll1Click(TObject *Sender);
private:  // User declarations
public:   // User declarations
        __fastcall TTextChildForm(TComponent* Owner);
};
//---------------------------------------------------------------------------//
extern PACKAGE TTextChildForm *TextChildForm;
//---------------------------------------------------------------------------//
#endif
```

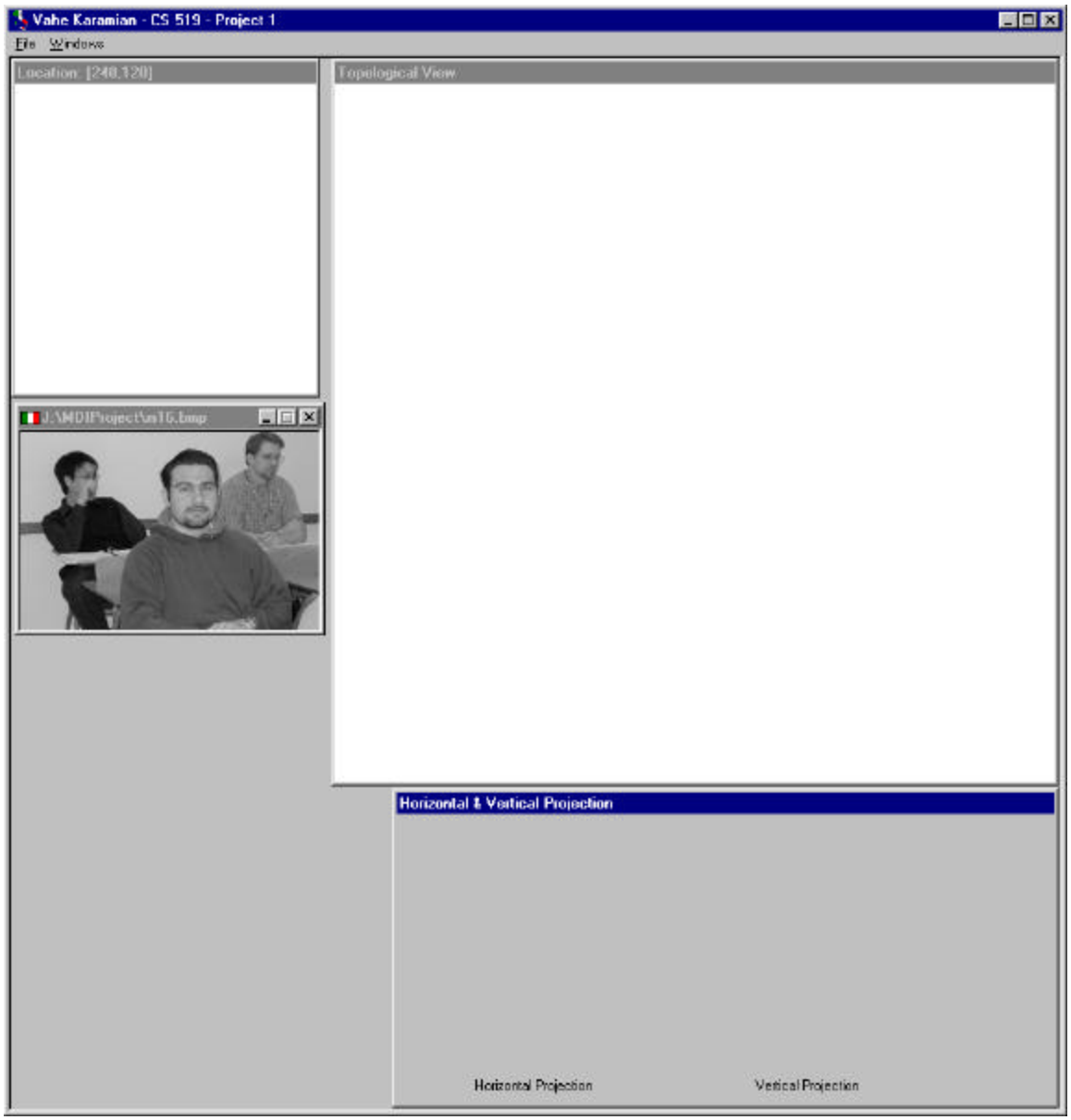
```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: TopologicalView.cpp

#include <vcl.h>
#pragma hdrstop

#include "TopologicalView.h"
#include "ImageChildUnit.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmTopologicalView *frmTopologicalView;
//---------------------------------------------------------------------------
__fastcall TfrmTopologicalView::TfrmTopologicalView(TComponent* Owner)
  : TForm(Owner)
{
  this->Caption = "Topological View";
  this->Width = 600;
  this->Height = 600;
  this->Color = clWhite;

  topologyImage->Width = this->Width;
  topologyImage->Height = this->Height;
}
//---------------------------------------------------------------------------

void __fastcall TfrmTopologicalView::FormClose(TObject *Sender,
      TCloseAction &Action)
{
   Action = caFree;
}

void __fastcall TfrmTopologicalView::Image( BYTE ImageData[256][256],
                                            int width, int height )
{
  for( int x=0; x<256; x++ )
    for( int y=0; y<256; y++ )
      Data[x][y] = ImageData[x][y];

  int u = 0;
  int v = 0;
  int a, b;

  for( b=0; b<height; b++ )
  {
    for( a=0; a<width; a++ )
    {
      u = a + 0.5 * b;
      v = Data[a][b] + 0.5 * b;
      if( v > this->Width )
        this->Width = v;
      if( u > this->Height )
        this->Height = u;
    }
  }

  u = 0;
  v = 0;

  for( b=height-1; b>=0; b-- )
  {
    for( a=width-1; a>=0; a-- )
    {
      u = a + 1.5 * b;
      v = Data[a][b] + 1.5 * b;
      topologyImage->Canvas->Pixels[u][v] = RGB( Data[a][b], Data[a][b], Data[a][b] );
    }
  }

}
```

```
// Vahe Karamian - CS 519 - Computer Vision - Project 1 - Dr. Lee
// Filename: TopologicalView.h

#ifndef TopologicalViewH
#define TopologicalViewH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TfrmTopologicalView : public TForm
{
__published:  // IDE-managed Components
  TImage *topologyImage;

  void __fastcall FormClose(TObject *Sender, TCloseAction &Action);

private:  // User declarations

  BYTE Data[256][256];

  int  HEIGHT;
  int  WIDTH;

public:   // User declarations

  __fastcall TfrmTopologicalView(TComponent* Owner);

  void __fastcall Image( BYTE Image[256][256], int width, int height );

};
//---------------------------------------------------------------------------
extern PACKAGE TfrmTopologicalView *frmTopologicalView;
//---------------------------------------------------------------------------
#endif
```

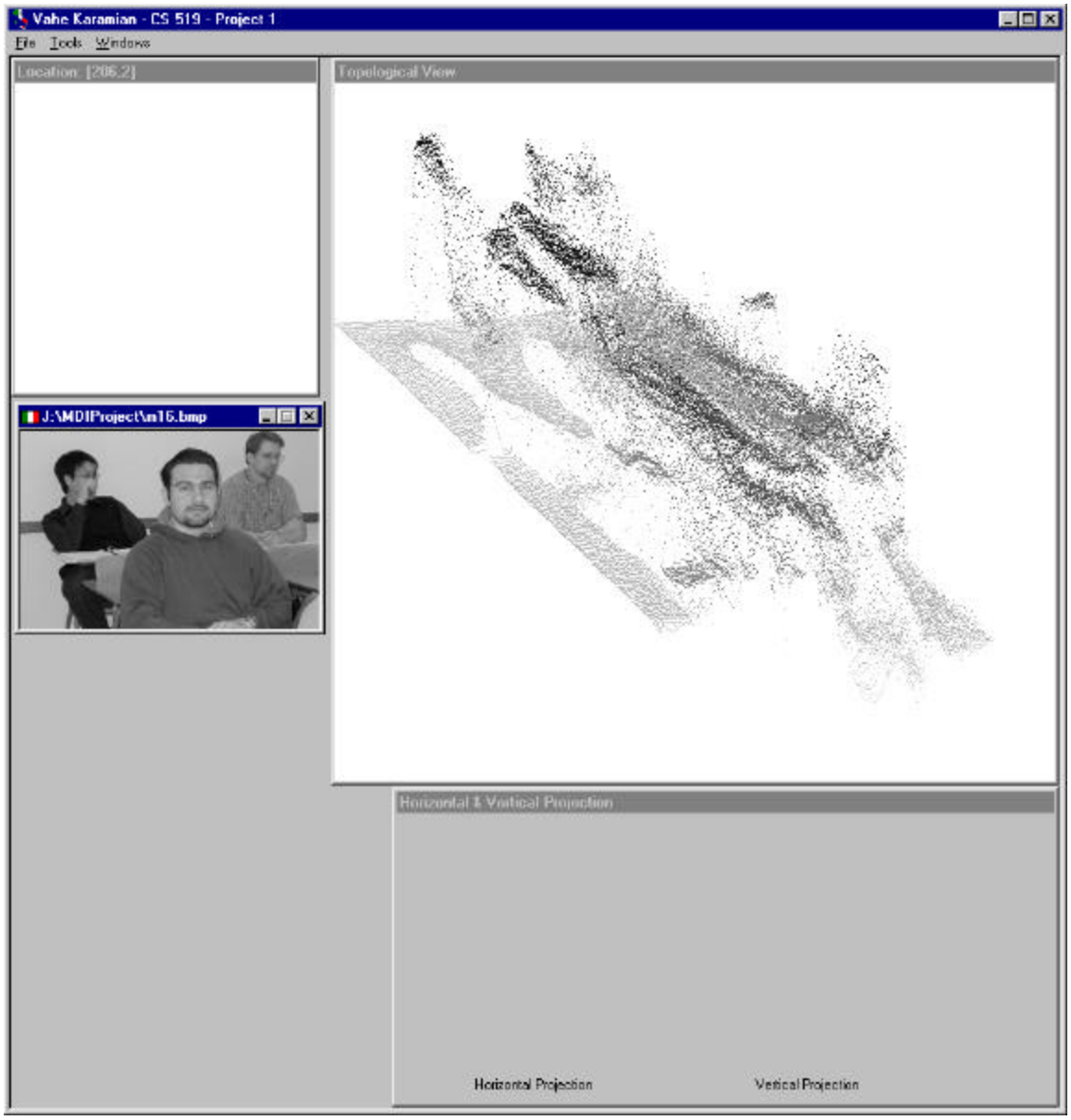
Vahe Karamian
Project 1
CS 519
Dr. Lee

Project I
Computer Vision

The purpose of this project was to practice early phase of computer vision, that is basic image processing. The tasks that had to be accomplished for the project where as follow: (1) capturing an image in any file format, in this project the application supports BMP, JPEG, and PGM file formats. The images can be opened regardless of their resolution or color depth. But for simplicity, the resolution of the image I am working to illustrate the program is 256 x 192 and it is grayscale. The following is the screen you see when running the program and opening the image file.

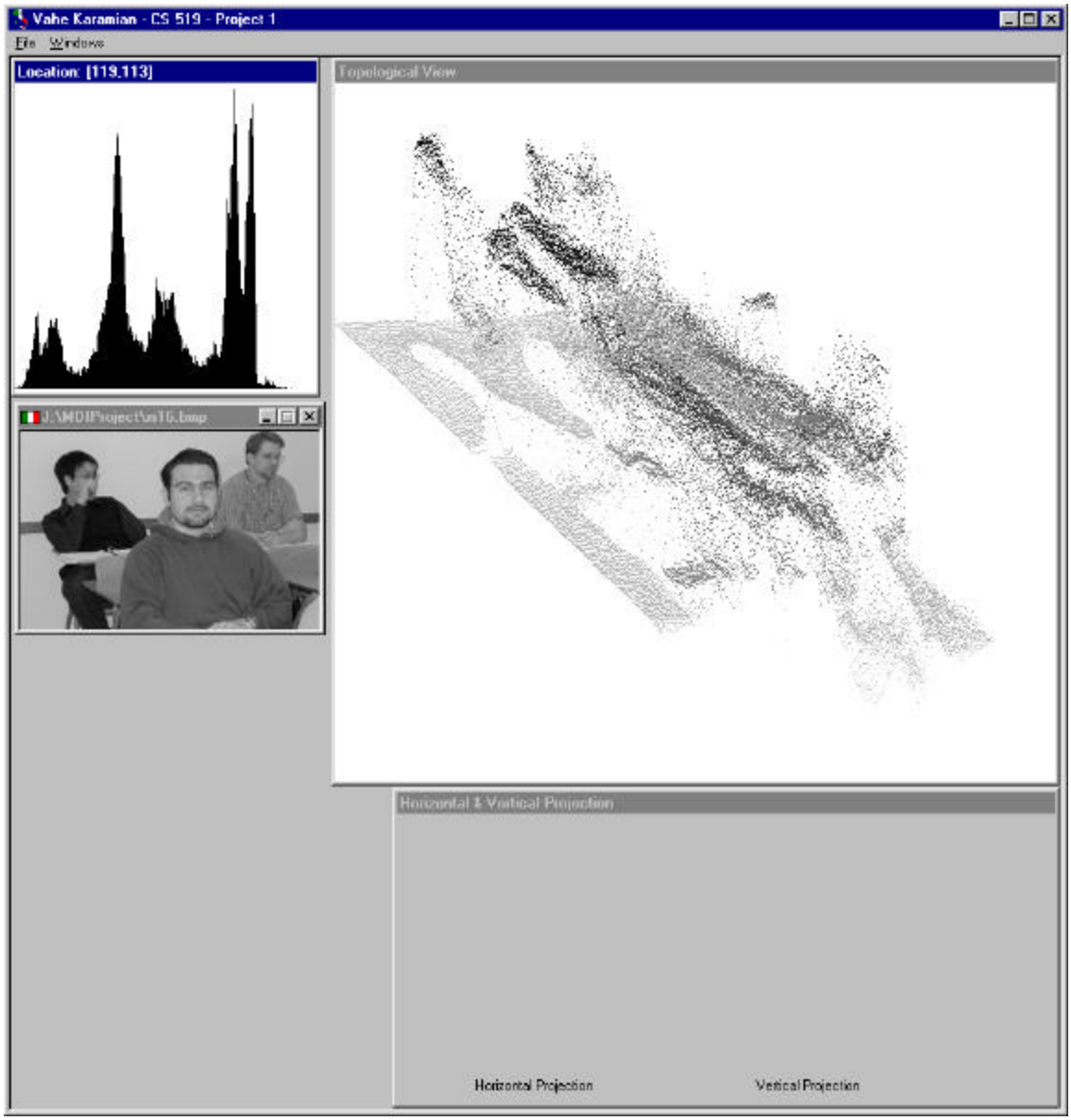


(2) We had to display a topographic representation of the image. This has been done by “Poorman’s 3D” algorithm with a little bit of modification, the following is the output:



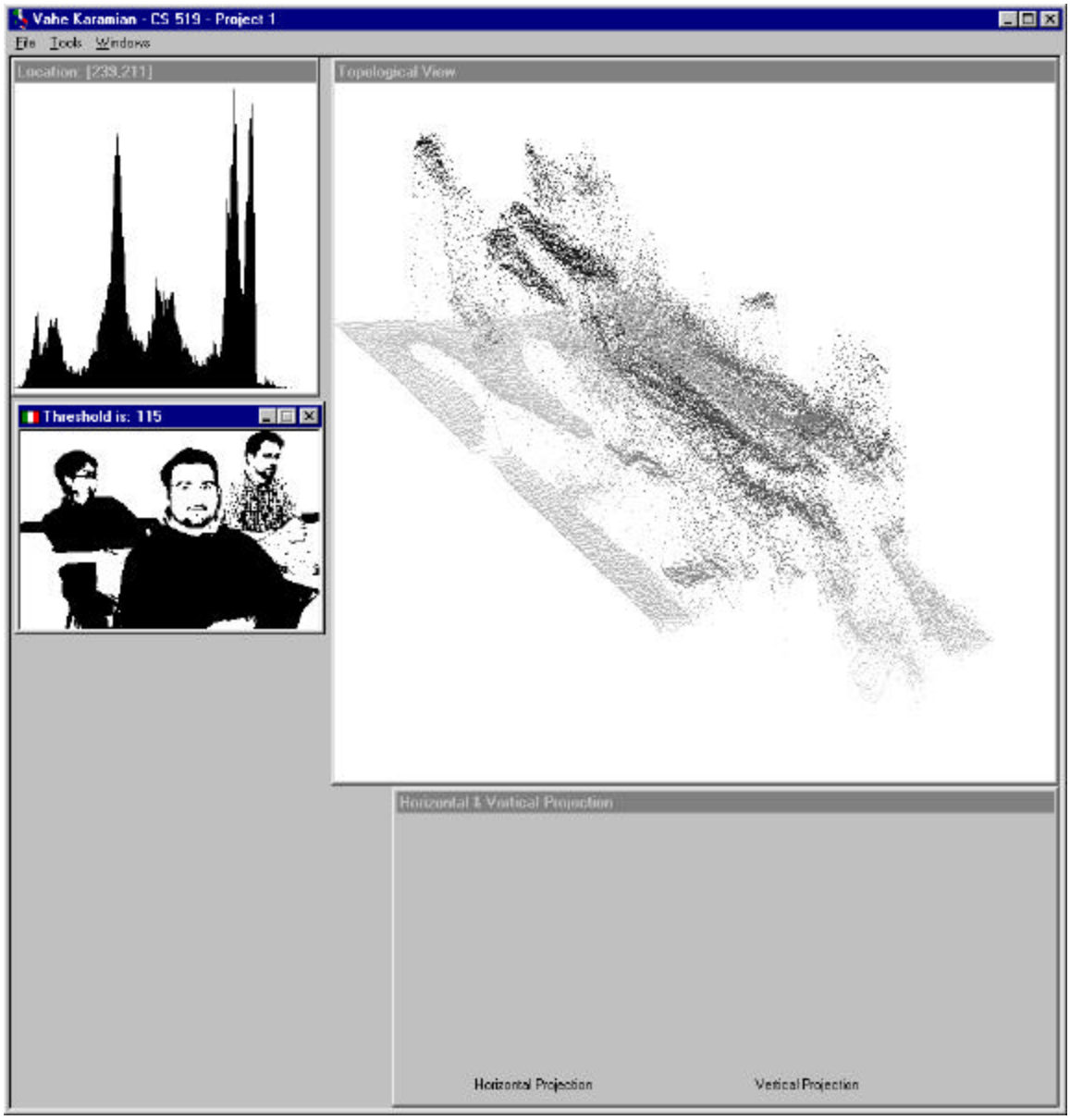
(4) Some image processing operations have been implemented, they are as follow: interactively determine the threshold value and convert the binary image, smoothing the image, edge detecting, displaying the vertical and horizontal projection of the image.

In this window you will see the histogram of the image and the user than can click on the histogram image to select a threshold value.



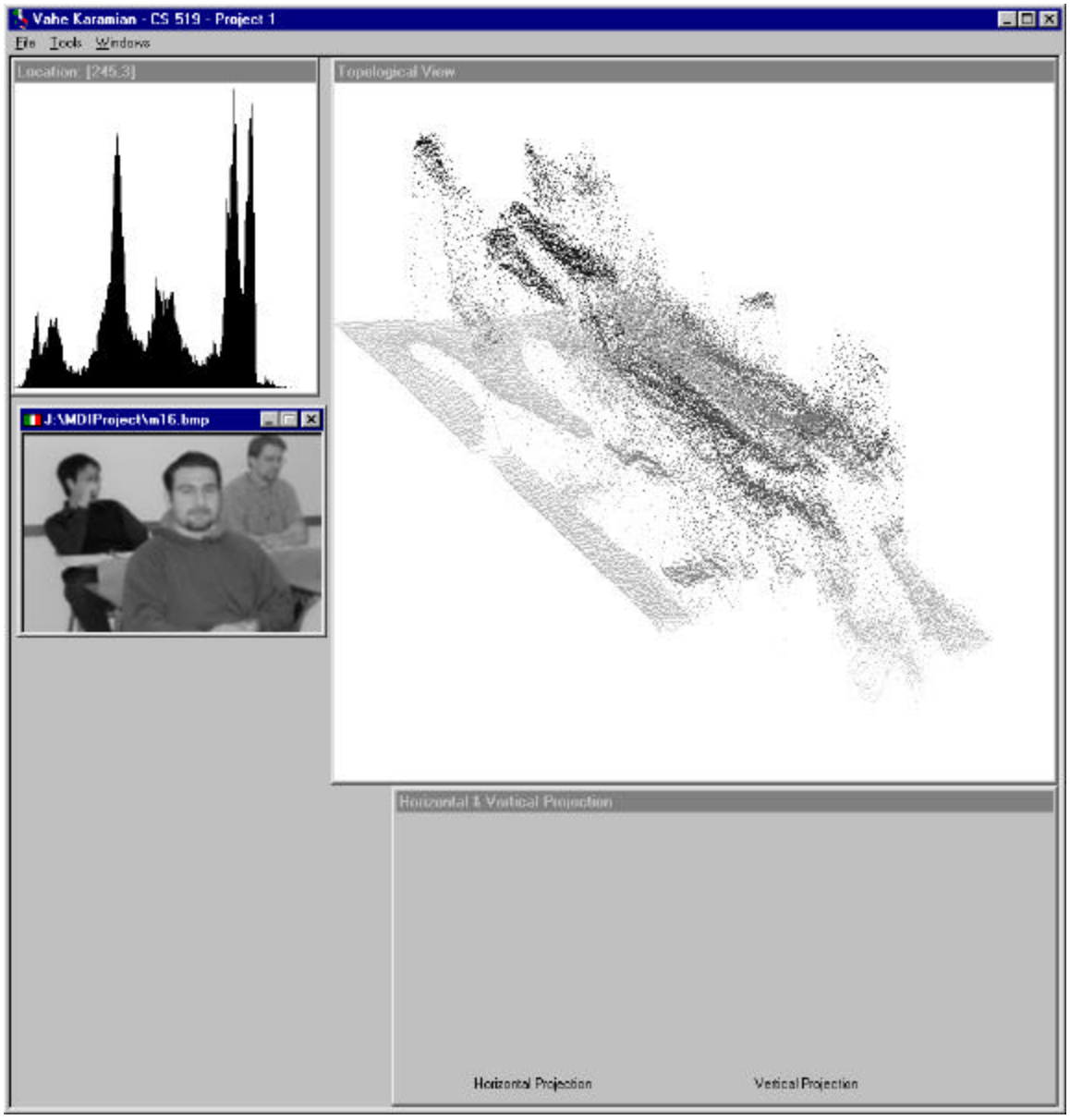
Here we get the histogram of the image, from the histogram data we can select a threshold value and proceed to convert the data into a binary picture.

The following display the image after the selection of the threshold value from the histogram window.



The threshold value for the given figure is 115.

Next we demonstrate smoothing of the image.

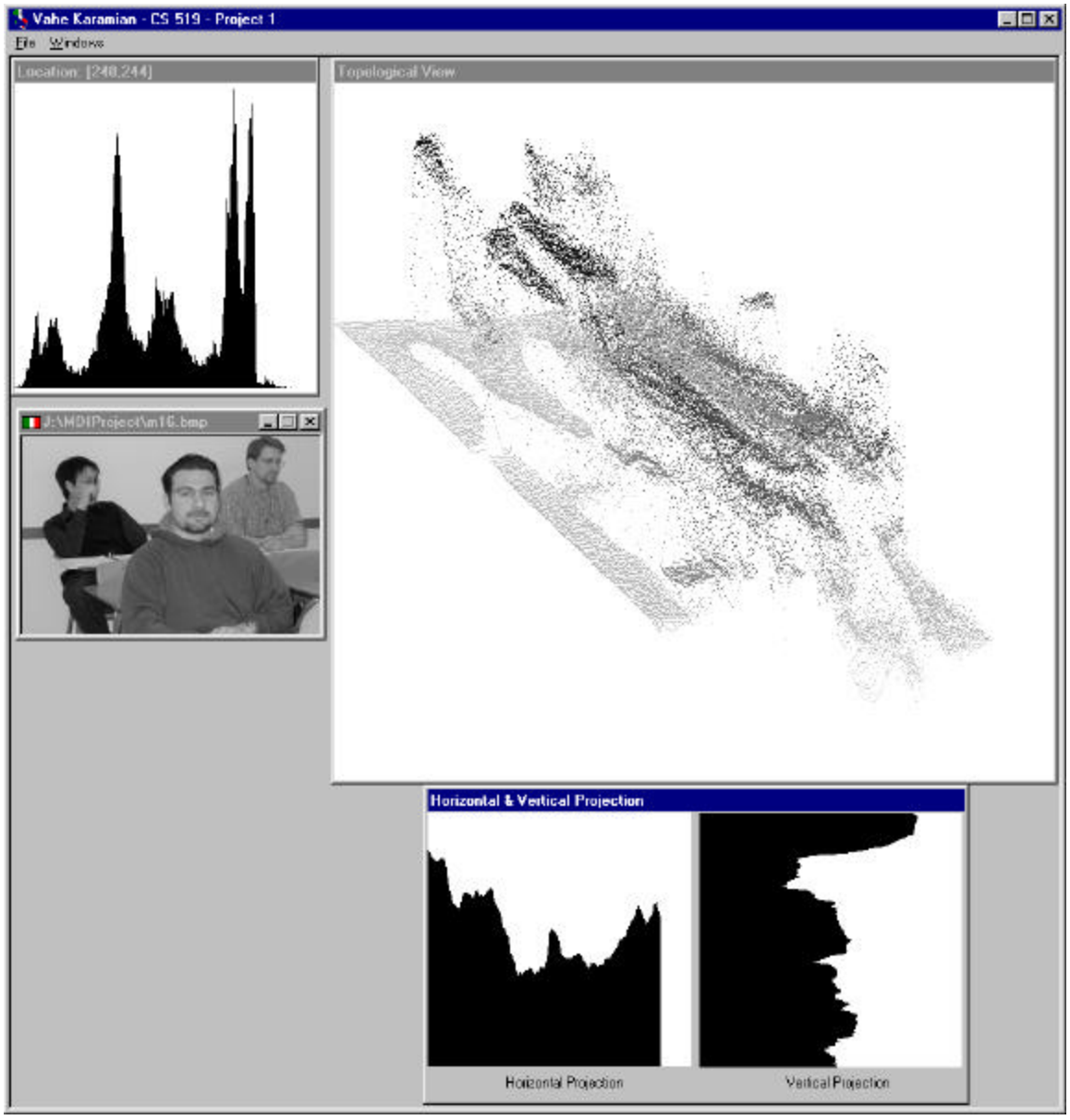


In Spatial operations, a mask is used to define the region of interest and the contribution of each pixel. Based on the mask, a newly calculated vale is then assigned to the pixel in the center. For practical reasons, the mask is usually square, with sizes such as 3x3 or 5x5. However, sometimes the size of a mask can be as large as 17x17, as in the case of a Mexican Hat function, which has some properties similar to those of the human eye's retina and is therefore used in computer vision systems. The mask used in this project is a simple smoothing filter that has a 3x3 mask, as follows:

[1/9][1/9][1/9]
[1/9][1/9][1/9]
[1/9][1/9][1/9]

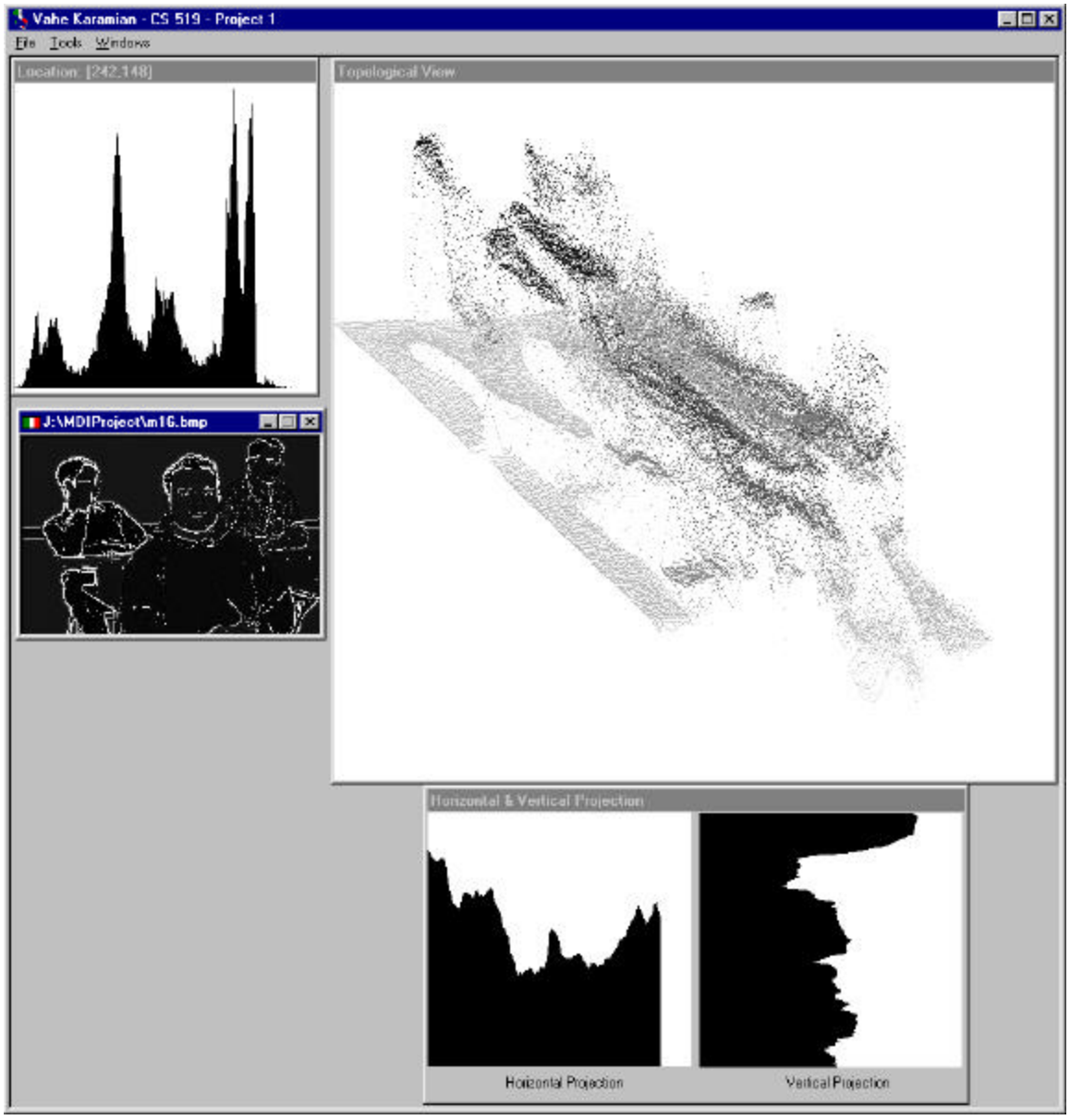
Effectively, a pixel value is replaced by the averaged pixel value of a 3x3 neighborhood.

This window displays the horizontal and vertical projection of the image.



Here you see the horizontal and vertical projection of the image. Summing all pixel values at each column and dividing them by the width to get the average for the horizontal, and summing all pixel values at each row and dividing by the height to get the average for the vertical projection have generated the data.

And finally we have the figure for edge detect algorithm.



Edge detection is a form of high-pass filtering because edges in an image are associated with high spatial frequencies, that is, areas with fast changing content. Edge detection is an important pre-processing technique often employed in computer vision system because it helps to define the boundary of an object – the first step to object recognition. The following high-pass filtering was used for this project:

[-1/9][-1/9][-1/9]
[-1/9][1][-1/9]
[-1/9][-1/9][-1/9]

However, there is an implementation complication that needs attention. When applying the mask to pixels at the edges of an image, the mask may extend beyond the image. To avoid this problem, the simplest solution is to leave the pixels on the edges initially and then copy the next available pixel value to it after processing. Interpolation can be employed for a better result.