รายการอ้างอิง

1. Nick Efford, Digital Image Processing: a practical introduction using Java, United States of America: Pearson Education Limited, 2000.
2. Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, พิมพ์ครั้งที่2, New Jersey: Prentice Hall, 2002.
3. John R. Smith, Shih-Fu Chang, Multi-stage Classification of Images from Features and Related Text, Fourth DELOS Workshop, San Miniato, Italy, August, 1997 : pp.2
4. Pethuru (Peter) Raj, Component Technology (Software Components), <http://www.peterindia.com/ComponentOverview.html>, 2002.
5. Using Library Databases and Dynamic-Link Libraries, MSDN: Microsoft, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/office97/html/usinglibrarydatabasesdynamic-linklibraries.asp>.
6. Thomas J. Olson, Nicholas G. Klop, Mark R. Hyett, and Shawn M. Carnell. MAVIS: A Visual Environment for Active Computer Vision. IEEE Workshop on Visual Languages, September 1992.
7. Ad Oculos 4.0, DBS Digitale Bildverarbeitung und Systementwicklung GmbH, Germany, <http://dbs-imaging.com/>, 1998.
8. Timothy A. Budd, Classic Data Structures in JAVA, United States of America: Addison Wesley Longman, 2001.
9. Gregory L. Heileman, Data Structure, Algorithms, and Object-Oriented Programming, Singapore: McGRAW-HILL, 1996.
10. Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, <http://www.imageprocessingbook.com/>.

ภาคผนวก

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ก
ตัวอย่างรหัสโปรแกรม

บทนี้เป็นการแสดงตัวอย่างรหัสโปรแกรม (Source code) โครงสร้างข้อมูล และรหัสโปรแกรมการใช้งานโครงสร้างข้อมูล ซึ่งมีรายละเอียดดังต่อไปนี้

ก.1 รหัสโปรแกรมโครงสร้างข้อมูล

รหัสโปรแกรมในหัวข้อนี้ประกอบด้วยรหัสโปรแกรมโครงสร้างข้อมูลตามข้อกำหนดในหัวข้อ 4.5.1 ที่เขียนขึ้นจากโปรแกรมภาษา Delphi โปรแกรมภาษา BCB และโปรแกรมภาษา VC++ ซึ่งมีรายละเอียดแสดงในรูปที่ ก.1 ถึง ก.2

```
type
TPixel_Data = record
 case ShortInt of
   0: (scVal: ^ShortInt);
   1: (ucVal: ^Byte);
   2: (ssVal: ^SmallInt);
   3: (usVal: ^Word);
   4: (siVal: ^Integer);
   5: (uiVal: ^Cardinal);
   6: (bVal: ^Boolean);
   7: (slVal: ^LongInt);
   8: (ulVal: ^Cardinal);
   9: (fltVal: ^Single);
   10: (sdVal: ^Double);
   11: (sldVal: ^Extended);
 end;
type PData_Layer   = ^TPixel_Data;
type PPData_Layer  = ^PData_Layer;
type
TData_Header = record
          SystemType: Integer;
          PixelType: Integer;
          Width,Height,Depth: Integer;
 end;
type PData_Header=^TData_Header;

type
TData = record
   Header: TData_Header;
   Layer: PPData_Layer;
 end;
type PData =^TData;
```

รูปที่ ก.1 รหัสโปรแกรมภาษา Delphi ของโครงสร้างข้อมูล

```
struct TPixel_Data
{
   union
   {
      char*              scVal;
      unsigned char*     ucVal;
      signed short*      ssVal;
      unsigned short*    usVal;
      signed int*        siVal;
      unsigned int*      uiVal;
      bool*              bVal;
      signed long*       slVal;
      unsigned long*     ulVal;
      float*             fltVal;
      double*            sdVal;
      long double*       sldVal;
   };
};
typedef TPixel_Data* PData_Layer;
typedef TPixel_Data** PPData_Layer;
struct TData_Header{
         int SystemType;
         int PixelType;
         int Width,Height,Depth;
};
typedef TData_Header* PData_Header;
struct TData{
   TData_Header Header;
   PPData_Layer Layer;
};
typedef TData* PData;
```

รูปที่ ก.2 รหัสโปรแกรมภาษา BCB และภาษา VC++ ของโครงสร้างข้อมูล

ก.2 รหัสโปรแกรมการใช้งานโครงสร้างข้อมูล

ตัวอย่างรหัสโปรแกรมในหัวข้อนี้ประกอบด้วย รหัสโปรแกรมสำหรับจองหน่วยความจำให้กับโครงสร้างข้อมูล และรหัสโปรแกรมสำหรับคืนหน่วยความจำโครงสร้างข้อมูล ที่เขียนขึ้นจากโปรแกรมภาษา Delphi โปรแกรมภาษา BCB และโปรแกรมภาษา VC++ ซึ่งมีรายละเอียดแสดงในรูปที่ ก.3 ถึง ก.4

```
function NewData(SystemType,PixelType: Integer; Width,Height,Depth: Integer): PData;
var
   Band: Integer;
   NewItem: PData;
```

รูปที่ ก.3 รหัสโปรแกรมภาษา Delphi ของการใช้งานโครงสร้างข้อมูล

```
    d,h: Integer;
    tmpLayer: PPData_Layer;
    tmpRow: PData_Layer;
begin

    case SystemType of
      1: Band := 1; //BW
      2: Band := 1; //Gray
      3: Band := 4; //RGB
      4: Band := 4; //CMY
      5: Band := 4; //HSI
      6: Band := 2; //Complex
      7: Band := 1; //Label
      8: Band := 1; //Histogram
    else
      Band := 1;
    end;

    NewItem := AllocMem(SizeOf(TData));

    NewItem^.Header.SystemType   := SystemType;
    NewItem^.Header.PixelType    :=PixelType;
    NewItem^.Header.Width        :=Width;
    NewItem^.Header.Height       :=Height;
    NewItem^.Header.Depth        :=Depth;

    NewItem^.Layer := AllocMem(Depth * SizeOf(PData_Layer));
    tmpLayer := NewItem^.Layer;
    for d := 0 to Depth-1 do
    begin
      tmpLayer^ := AllocMem(Height * SizeOf(TPixel_Data));
      tmpRow := tmpLayer^;
      Inc(tmpLayer);
      for h := 0 to Height-1 do
      begin

        case PixelType of
          0: tmpRow^.scVal := AllocMem(Width * Band * SizeOf(ShortInt));
          1: tmpRow^.ucVal := AllocMem(Width * Band * SizeOf(Byte));
          2: tmpRow^.ssVal := AllocMem(Width * Band * SizeOf(SmallInt));
          3: tmpRow^.usVal := AllocMem(Width * Band * SizeOf(Word));
          4: tmpRow^.siVal := AllocMem(Width * Band * SizeOf(Integer));
          5: tmpRow^.uiVal := AllocMem(Width * Band * SizeOf(Cardinal));
          6: tmpRow^.bVal := AllocMem(Width * Band * SizeOf(Boolean));
          7: tmpRow^.slVal := AllocMem(Width * Band * SizeOf(LongInt));
          8: tmpRow^.ulVal := AllocMem(Width * Band * SizeOf(Cardinal));
          9: tmpRow^.fltVal := AllocMem(Width * Band * SizeOf(Single));
          10: tmpRow^.sdVal := AllocMem(Width * Band * SizeOf(Double));
          11: tmpRow^.sldVal := AllocMem(Width * Band * SizeOf(Extended));
        end;
        Inc(tmpRow);

      end;
    end;

    NewData := NewItem;
end;
```

รูปที่ ก.3 รหัสโปรแกรมภาษา Delphi ของการใช้งานโครงสร้างข้อมูล (ต่อ)

```
function DeleteData(Data: PData): Integer;
var
   Height: Integer;
   Depth: Integer;
   PixelType: Integer;
   d,h: Integer;

   tmpLayer: PPData_Layer;
   tmpRow: PData_Layer;

begin
   if Data = Nil then
   begin
      DeleteData := -2; //No Data
      Exit;
   end;

   Height := Data^.Header.Height;
   Depth := Data^.Header.Depth;
   PixelType := Data^.Header.PixelType;

   tmpLayer := Data^.Layer;
   for d := 0 to Depth-1 do
   begin

      tmpRow := tmpLayer^;
      for h := 0 to Height-1 do
      begin

         case PixelType of
            0: Dispose(tmpRow^.scVal);
            1: Dispose(tmpRow^.ucVal);
            2: Dispose(tmpRow^.ssVal);
            3: Dispose(tmpRow^.usVal);
            4: Dispose(tmpRow^.siVal);
            5: Dispose(tmpRow^.uiVal);
            7: Dispose(tmpRow^.slVal);
            8: Dispose(tmpRow^.ulVal);
            9: Dispose(tmpRow^.fltVal);
            10: Dispose(tmpRow^.sdVal);
            11: Dispose(tmpRow^.sldVal);
         end;

         Inc(tmpRow);
      end;

      Dispose(tmpLayer^);
      Inc(tmpLayer);
   end;

   Dispose(Data^.Layer);
   Dispose(Data);
   DeleteData := 0;

end;
```

รูปที่ ก.3 รหัสโปรแกรมภาษา Delphi ของการใช้งานโครงสร้างข้อมูล (ต่อ)

```
function NewCopyData(out Destination: PData; Source: PData): Integer;
var
   SystemType,PixelType: Integer;
   Depth,Height,Width: Integer;

   Band: Integer;
   NewItem: PData;

   d,h: Integer;
   tmpDLayer: PPData_Layer;
   tmpDRow: PData_Layer;
   tmpSLayer: PPData_Layer;
   tmpSRow: PData_Layer;

begin
   if Source = Nil then
   begin
      NewCopyData := -2; //No Data
      Exit;
   end;

   SystemType := Source^.Header.SystemType;
   PixelType := Source^.Header.PixelType;
   Depth := Source^.Header.Depth;
   Height := Source^.Header.Height;
   Width := Source^.Header.Width;

   case SystemType of
    1: Band := 1; //BW
    2: Band := 1; //Gray
    3: Band := 4; //RGB
    4: Band := 4; //CMY
    5: Band := 4; //HSI
    6: Band := 2; //Complex
    7: Band := 1; //Label
    8: Band := 1; //Histogram
   else
    Band := 1;
   end;

   NewItem := AllocMem(SizeOf(TData));

   NewItem^.Header.SystemType   :=Source^.Header.SystemType;
   NewItem^.Header.PixelType    :=Source^.Header.PixelType;
   NewItem^.Header.Width        :=Source^.Header.Width;
   NewItem^.Header.Height       :=Source^.Header.Height;
   NewItem^.Header.Depth        :=Source^.Header.Depth;

   NewItem^.Layer := AllocMem(Depth * SizeOf(PData_Layer));

   tmpDLayer := NewItem^.Layer;
   tmpSLayer := Source^.Layer;

   for d := 0 to Depth-1 do
   begin
```

รูปที่ ก.3 รหัสโปรแกรมภาษา Delphi ของการใช้งานโครงสร้างข้อมูล (ต่อ)

```
      tmpDLayer^ := AllocMem(Height * SizeOf(TPixel_Data));

      tmpDRow := tmpDLayer^;
      tmpSRow := tmpSLayer^;

      Inc(tmpDLayer);
      Inc(tmpSLayer);

      for h := 0 to Height-1 do
      begin

        case PixelType of
          0: tmpDRow^.scVal := AllocMem(Width * Band * SizeOf(ShortInt));
          1: tmpDRow^.ucVal := AllocMem(Width * Band * SizeOf(Byte));
          2: tmpDRow^.ssVal := AllocMem(Width * Band * SizeOf(SmallInt));
          3: tmpDRow^.usVal := AllocMem(Width * Band * SizeOf(Word));
          4: tmpDRow^.siVal := AllocMem(Width * Band * SizeOf(Integer));
          5: tmpDRow^.uiVal := AllocMem(Width * Band * SizeOf(Cardinal));
          6: tmpDRow^.bVal := AllocMem(Width * Band * SizeOf(Boolean));
          7: tmpDRow^.slVal := AllocMem(Width * Band * SizeOf(LongInt));
          8: tmpDRow^.ulVal := AllocMem(Width * Band * SizeOf(Cardinal));
          9: tmpDRow^.fltVal := AllocMem(Width * Band * SizeOf(Single));
          10: tmpDRow^.sdVal := AllocMem(Width * Band * SizeOf(Double));
          11: tmpDRow^.sldVal := AllocMem(Width * Band * SizeOf(Extended));
        end;

        case PixelType of
          0: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(ShortInt));
          1: Move(tmpSRow^.scVal^,tmpDRow^.scVal^,Width * Band * SizeOf(Byte));
          2: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(SmallInt));
          3: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(Word));
          4: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(Integer));
          5: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(Cardinal));
          6: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(Boolean));
          7: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(LongInt));
          8: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(Cardinal));
          9: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(Single));
          10: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(Double));
          11: Move(tmpSRow^.scVal,tmpDRow^.scVal,Width * Band * SizeOf(Extended));
        end;

        Inc(tmpDRow);
        Inc(tmpSRow);

      end;

    end;

    Destination := NewItem;

    NewCopyData := 0;

end;
```

รูปที่ ก.3 รหัสโปรแกรมภาษา Delphi ของการใช้งานโครงสร้างข้อมูล (ต่อ)

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

```
PData NewData(int SystemType, int PixelType, int Width, int Height, int Depth)
{
   int Band;
   switch(SystemType)
   {
      case 1:   Band=1;   break; //Black & White
      case 2:   Band=1;   break; //Gray
      case 3:   Band=4;   break; //RGB
      case 4:   Band=4;   break; //CMY
      case 5:   Band=4;   break; //HSI
      case 6:   Band=2;   break; //Complex
      case 7:   Band=1;   break; //Label
      case 8:   Band=1;   break; //Histogram
      default:  Band=1;
   }
   PData NewData=new TData;
   NewData->Header.SystemType=SystemType;
   NewData->Header.PixelType=PixelType;
   NewData->Header.Width=Width;
   NewData->Header.Height=Height;
   NewData->Header.Depth=Depth;
   NewData->Layer=new PData_Layer[Depth];
   for(int d=0;d<Depth;d++)
   {
      NewData->Layer[d]=new TPixel_Data[Height];
      for(int h=0;h<Height;h++)
         switch (PixelDepth)
         {
            case 0: NewData->Layer[d][h].scVal=new char[Width*Band];
                    break;
            case 1: NewData->Layer[d][h].ucVal=new unsigned char[Width*Band];
                    break;
            case 2: NewData->Layer[d][h].ssVal=new signed short[Width*Band];
                    break;
            case 3: NewData->Layer[d][h].usVal=new unsigned short[Width*Band];
                    break;
            case 4: NewData->Layer[d][h].siVal=new signed int[Width*Band];
                    break;
            case 5: NewData->Layer[d][h].uiVal=new unsigned int[Width*Band];
                    break;
            case 6: NewData->Layer[d][h].bVal=new bool[Width*Band];
                    break;
            case 7: NewData->Layer[d][h].slVal=new signed long[Width*Band];
                    break;
            case 8: NewData->Layer[d][h].ulVal=new unsigned long[Width*Band];
                    break;
            case 9: NewData->Layer[d][h].fltVal=new float[Width*Band];
                    break;
            case 10: NewData->Layer[d][h].sdVal=new double[Width*Band];
                    break;
            case 11: NewData->Layer[d][h].sldVal=new long double[Width*Band];
                    break;
         }
   }
   return NewData;
}
```

รูปที่ ก.4 รหัสโปรแกรมภาษา BCB และภาษา VC++ ของการใช้งานโครงสร้างข้อมูล

```
int DeleteData(PData &Data)
{
   if(Data==NULL) return -2;
   int Height=Data->Header.Height;
   int Depth=Data->Header.Depth;

   for(int d=0;d<Depth;d++)
   {
      for(int h=0;h<Height;h++)
         switch (Data->Header.PixelDepth)
         {
            case 0: delete Data->Layer[d][h].scVal;
                    break;
            case 1: delete Data->Layer[d][h].ucVal;
                    break;
            case 2: delete Data->Layer[d][h].ssVal;
                    break;
            case 3: delete Data->Layer[d][h].usVal;
                    break;
            case 4: delete Data->Layer[d][h].siVal;
                    break;
            case 5: delete Data->Layer[d][h].uiVal;
                    break;
            case 6: delete Data->Layer[d][h].bVal;
                    break;
            case 7: delete Data->Layer[d][h].slVal;
                    break;
            case 8: delete Data->Layer[d][h].ulVal;
                    break;
            case 9: delete Data->Layer[d][h].fltVal;
                    break;
            case 10: delete Data->Layer[d][h].sdVal;
                    break;
            case 11: delete Data->Layer[d][h].sldVal;
                    break;
         }
      delete[] Data->Layer[d];
   }
   delete[] Data->Layer;
   delete Data;
   Data=NULL;

   return 0;
}

int NewCopyData(PData &Destination,PData Source)
{
   if(Source==NULL) return -2;

   int Band;
   switch(Source->Header.System)
   {
      case 1:   Band=1;   break; //Black & White
      case 2:   Band=1;   break; //Gray
      case 3:   Band=4;   break; //RGB
      case 4:   Band=4;   break; //CMY
```

รูปที่ ก.4 รหัสโปรแกรมภาษา BCB และภาษา VC++ ของการใช้งานโครงสร้างข้อมูล (ต่อ)

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

```
    case 5:   Band=4;   break; //HSI
    case 6:   Band=2;   break; //Complex
    case 7:   Band=1;   break; //Label
    case 8:   Band=1;   break; //Histogram
    default:  Band=1;
}

Destination=new TData;
int Width = Source->Header.Width;
int Height = Source->Header.Height;
int Depth = Source->Header.Depth;

Destination->Header.SystemType = Source->Header.SystemType;
Destination->Header.PixelType  = Source->Header.PixelType;
Destination->Header.Width   = Source->Header.Width;
Destination->Header.Height  = Source->Header.Height;
Destination->Header.Depth   = Source->Header.Depth;

Destination->Layer=new PData_Layer[Depth];
for(int d=0;d<Source->Header.Depth;d++)
{
   Destination->Layer[d]=new TPixel_Data[Height];
   for(int h=0;h<Height;h++)
      switch (Source->Header.PixelType)
      {
         case 0: Destination->Layer[d][h].scVal=new char[Width*Band];
                 memcpy(Destination->Layer[d][h].scVal,
                 Source->Layer[d][h].scVal,Width*Band*sizeof(char));
                 break;
         case 1: Destination->Layer[d][h].ucVal=new unsigned char[Width*Band];
                 memcpy(Destination->Layer[d][h].ucVal,
                 Source->Layer[d][h].ucVal,Width*Band*sizeof(char));
                 break;
         case 2: Destination->Layer[d][h].ssVal=new signed short[Width*Band];
                 memcpy(Destination->Layer[d][h].ssVal,
                 Source->Layer[d][h].ssVal,Width*Band*sizeof(short));
                 break;
         case 3: Destination->Layer[d][h].usVal=new unsigned short[Width*Band];
                 memcpy(Destination->Layer[d][h].usVal,
                 Source->Layer[d][h].usVal,Width*Band*sizeof(short));
                 break;
         case 4: Destination->Layer[d][h].siVal=new signed int[Width*Band];
                 memcpy(Destination->Layer[d][h].siVal,
                 Source->Layer[d][h].siVal,Width*Band*sizeof(int));
                 break;
         case 5: Destination->Layer[d][h].uiVal=new unsigned int[Width*Band];
                 memcpy(Destination->Layer[d][h].uiVal,
                 Source->Layer[d][h].uiVal,Width*Band*sizeof(int));
                 break;
         case 6: Destination->Layer[d][h].bVal=new bool[Width*Band];
                 memcpy(Destination->Layer[d][h].bVal,
                 Source->Layer[d][h].bVal,Width*Band*sizeof(bool));
                 break;
         case 7: Destination->Layer[d][h].slVal=new signed long[Width*Band];
                 memcpy(Destination->Layer[d][h].slVal,
                 Source->Layer[d][h].slVal,Width*Band*sizeof(long));
                 break;
```

รูปที่ ก.4 รหัสโปรแกรมภาษา BCB และภาษา VC++ ของการใช้งานโครงสร้างข้อมูล (ต่อ)

```
            case 8: Destination->Layer[d][h].ulVal=new unsigned long[Width*Band];
                    memcpy(Destination->Layer[d][h].ulVal,
                    Source->Layer[d][h].ulVal,Width*Band*sizeof(long));
                    break;
            case 9: Destination->Layer[d][h].fltVal=new float[Width*Band];
                    memcpy(Destination->Layer[d][h].fltVal,
                    Source->Layer[d][h].fltVal,Width*Band*sizeof(float));
                    break;
            case 10: Destination->Layer[d][h].sdVal=new double[Width*Band];
                    memcpy(Destination->Layer[d][h].sdVal,
                    Source->Layer[d][h].sdVal,Width*Band*sizeof(double));
                    break;
            case 11: Destination->Layer[d][h].sldVal=new long double[Width*Band];
                    memcpy(Destination->Layer[d][h].sldVal,
                    Source->Layer[d][h].sldVal,Width*Band*sizeof(long double));
                    break;
        }
    }

    return 0;
}
```

รูปที่ ก.4 รหัสโปรแกรมภาษา BCB และภาษา VC++ ของการใช้งานโครงสร้างข้อมูล (ต่อ)

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ข
คู่มือการใช้เครื่องมือประมวลผลภาพดิจิทัล VisDIP-Tool

ภาคผนวก ข นี้เป็นการอธิบายการใช้เครื่องมือสำหรับทดลองประมวลผลภาพดิจิทัล เพื่อให้ผู้ใช้เข้าใจและสามารถนำไปใช้ในการออกแบบและทดสอบขั้นตอนวิธีในการประมวลผลภาพได้ ดังนั้นในหัวข้อนี้จะกล่าวถึงการจัดเตรียมเครื่องมือก่อนการใช้งาน ลักษณะโครงสร้างของหน้าจอและขั้นตอนการใช้เครื่องมือนี้

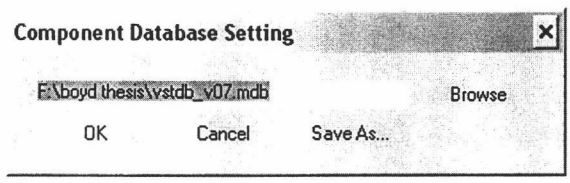
ข.1 การจัดเตรียมเครื่องมือก่อนการใช้งาน

เนื่องจากเครื่องมือประมวลผลภาพดิจิทัล VisDIP-Tool นี้ประกอบด้วยส่วนต่าง ๆ หลายส่วน จึงต้องมีการจัดเตรียมเครื่องมือก่อนการใช้งานเพื่อปรับแต่งระบบให้สามารถทำงานได้ การใช้งานเครื่องมือนี้ได้ต้องมีส่วนต่าง ๆ ดังนี้

1) ส่วนของระบบ ได้แก่แฟ้ม VisDIPTool.exe ซึ่งเป็นตัวโปรแกรม และแฟ้ม vstdb.mdb ซึ่งเป็นแฟ้มฐานข้อมูลของระบบ

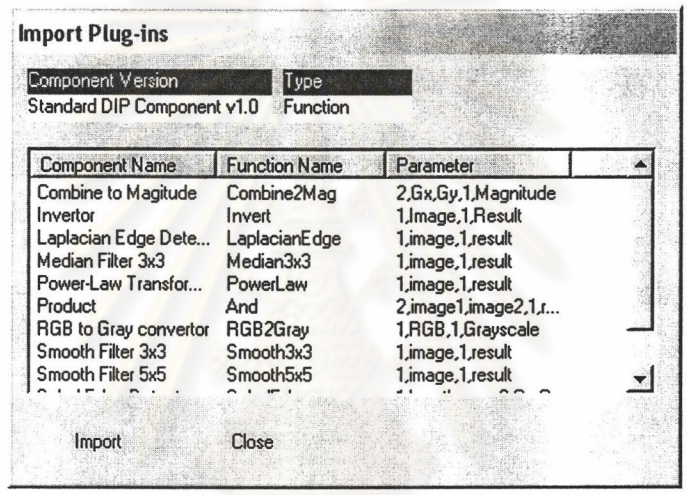
2) ส่วนของฟังก์ชันประมวลผลภาพ ซึ่งสร้างเป็นส่วนประกอบซอฟต์แวร์ตามข้อกำหนดในบทที่ 3 ของวิทยานิพนธ์นี้ โดยส่วนประกอบซอฟต์แวร์ที่ใช้กับเครื่องมือนี้จะอยู่ในรูปแฟ้ม .dll

เมื่อมีครบทั้งสองส่วนแล้วให้ทำการเปิดโปรแกรม VisDIP-Tool เพื่อจัดเตรียมการใช้งาน ซึ่งเมื่อเปิดโปรแกรมครั้งแรกอาจมีการแจ้งเตือนว่าหาฐานข้อมูลของระบบไม่เจอ ต้องทำการกำหนดฐานข้อมูลให้ระบบโดยเลือกจากเมนู "Setting->Component Database..." จะปรากฏหน้าต่างสำหรับเลือกแฟ้มฐานข้อมูลดังแสดงในรูปที่ ข.1 เมื่อเลือกแฟ้มฐานข้อมูลแล้วให้กดปุ่ม OK เพื่อติดตั้งแฟ้มฐานข้อมูลให้ระบบ



รูปที่ ข.1 หน้าต่างสำหรับเลือกฐานข้อมูล

เมื่อติดตั้งฐานข้อมูลให้ระบบเรียบร้อยแล้วขั้นตอนต่อไปต้องทำการเพิ่มฟังก์ชันประมวลผลภาพให้กับเครื่องมือก่อนใช้งานเนื่องจากเครื่องมือนี้เป็นระบบบรรณาธิกรที่ยังไม่มีฟังก์ชันประมวลผลภาพใด ๆ การเพิ่มฟังก์ชันประมวลผลภาพทำได้โดยเลือกเมนูย่อย Import Plug-ins... จากเมนูหลัก Component จะปรากฏหน้าต่างแบบผุดขึ้นให้เลือกส่วนประกอบซอฟต์แวร์ที่ต้องการจากนั้นให้กดปุ่ม OK จะปรากฏหน้าต่างนำเข้าส่วนประกอบดังแสดงในรูปที่ ข.2 หน้าต่างนี้จะแสดงรายการฟังก์ชันประมวลผลภาพทั้งหมดที่มีและสามารถเรียกใช้ได้จากส่วนประกอบซอฟต์แวร์ที่เลือก ให้ทำการเลือกฟังก์ชันที่ต้องการใช้งานแล้วกดปุ่ม Import ระบบจะทำการติดตั้งฟังก์ชันประมวลผลภาพให้กับเครื่องมือ



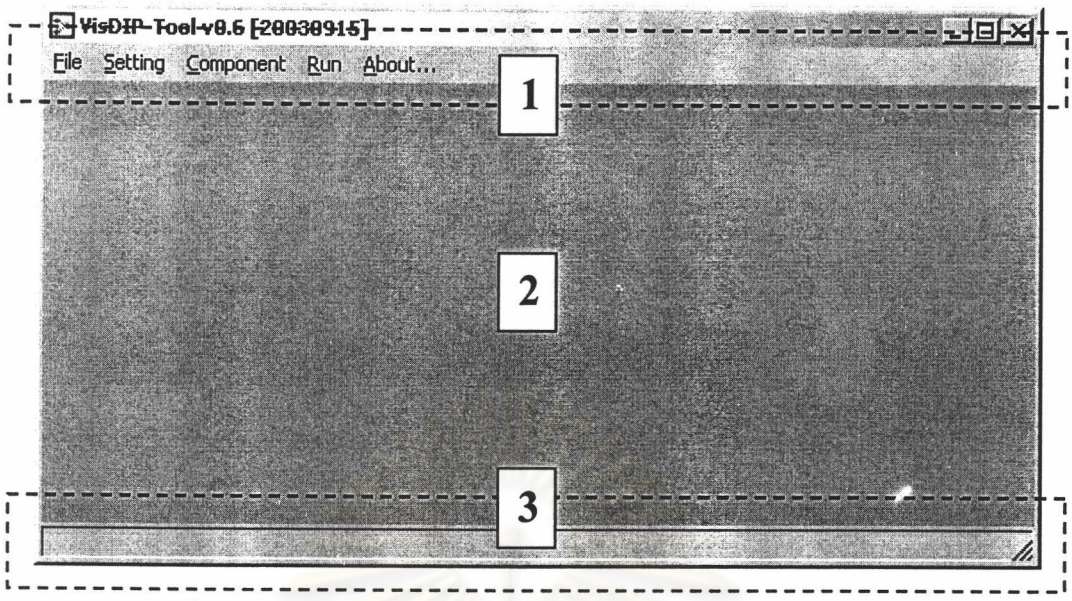
รูปที่ ข.2 หน้าต่างนำเข้าส่วนประกอบซอฟต์แวร์

ขั้นตอนทั้งสองเป็นการจัดเตรียมเครื่องมือก่อนการใช้งาน เมื่อทำเสร็จครบทุกขั้นตอนแล้วจึงสามารถใช้งานเครื่องมือได้อย่างสมบูรณ์

ข.2 หน้าจอหลักของระบบ

หน้าจอหลักของเครื่องมือประมวลผลภาพดิจิทัลนี้มีลักษณะดังรูปที่ ข.3 ซึ่งสามารถแบ่งโครงสร้างการทำงานได้เป็น 3 ส่วน ดังนี้

1. แถบเมนู
2. บริเวณบรรณาธิกร
3. แถบแสดงสถานะ



รูปที่ ข.3 หน้าจอหลักของเครื่องมือประมวลผลภาพดิจิทัล VisDIP-Tool

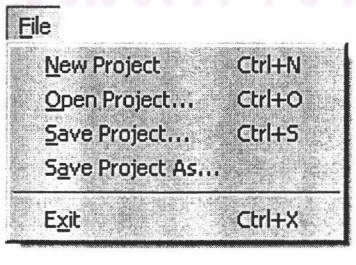
แถบเมนู

แถบเมนูนี้จะแสดงเมนูเพื่อเลือกการทำงาน ซึ่งจะประกอบด้วย 4 เมนูหลักคือ

1. เมนู File
2. เมนู Setting
3. เมนู Component
4. เมนู Run
5. เมนู About...

เมนู File

เป็นเมนูที่เกี่ยวกับการจัดการแฟ้มโครงการและการออกจากโปรแกรมเครื่องมือประมวลผลภาพ ซึ่งจะประกอบด้วยเมนูย่อยดังรูปที่ ข.4



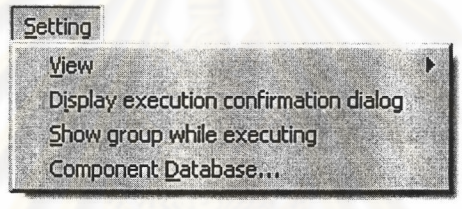
รูปที่ ข.4 เมนูย่อยของเมนู File

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

- เมนูย่อย New Project ใช้ในการสร้างแฟ้มโครงการใหม่
- เมนูย่อย Open Project... ใช้สำหรับเปิดแฟ้มโครงการ
- เมนูย่อย Save Project... ใช้สำหรับบันทึกกราฟกระแสข้อมูลเป็นแฟ้มโครงการเพื่อเก็บไว้ใช้ทดลองในครั้งต่อไปโดยจะบันทึกเป็นแฟ้ม .vtp (VisDIP-Tool Project) ซึ่งเป็นแฟ้มข้อความที่สามารถใช้โปรแกรม Notepad เปิดเพื่อดูรายละเอียดในแฟ้มได้
- เมนูย่อย Save Project As... ใช้สำหรับบันทึกแฟ้มโครงการเป็นชื่ออื่น
- เมนูย่อย Exit ใช้สำหรับออกจากโปรแกรมเครื่องมือประมวลผลภาพดิจิทัล

เมนู Setting

เป็นเมนูที่เกี่ยวกับการตั้งค่าต่าง ๆ ของระบบ ซึ่งจะประกอบด้วยเมนูย่อยดังรูปที่ ข.5

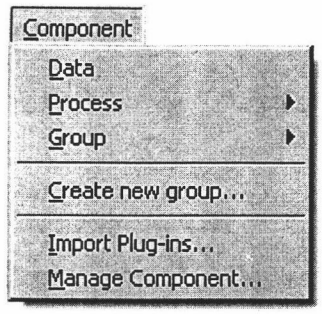


รูปที่ ข.5 เมนูย่อยของเมนู Setting

- เมนูย่อย View ใช้ในการแสดงหน้าต่างโต้ตอบของระบบ ได้แก่ Data Info, Process Info, Group Info
- เมนูย่อย Display execution confirmation dialog ใช้สำหรับกำหนดให้ระบบแสดงหน้าต่างยืนยันแบบผุดขึ้นเมื่อมีการสั่งงานให้เครื่องมือทำการประมวลผลภาพ
- เมนูย่อย Show group while executing ใช้สำหรับกำหนดให้ระบบแสดงการประมวลผลภาพของกราฟย่อยของกลุ่มฟังก์ชันประมวลผลภาพเมื่อกลุ่มฟังก์ชันประมวลผลภาพถูกสั่งให้ทำการประมวลผล
- เมนูย่อย Component Database... ใช้สำหรับกำหนดแฟ้มฐานข้อมูลให้ระบบ

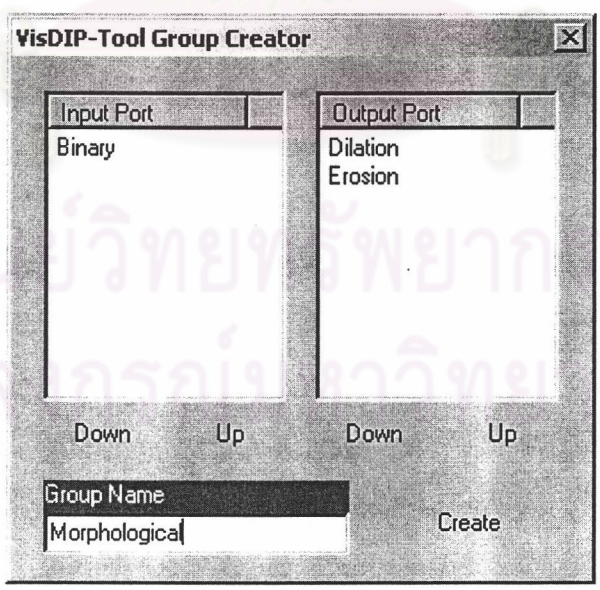
เมนู Component

เป็นเมนูที่เกี่ยวกับ การสร้างกราฟกระแสข้อมูล การสร้างกลุ่มฟังก์ชันประมวลผลภาพ และจัดการเกี่ยวกับส่วนประกอบซอฟต์แวร์ ซึ่งจะประกอบด้วยเมนูย่อยดังรูปที่ ข.6



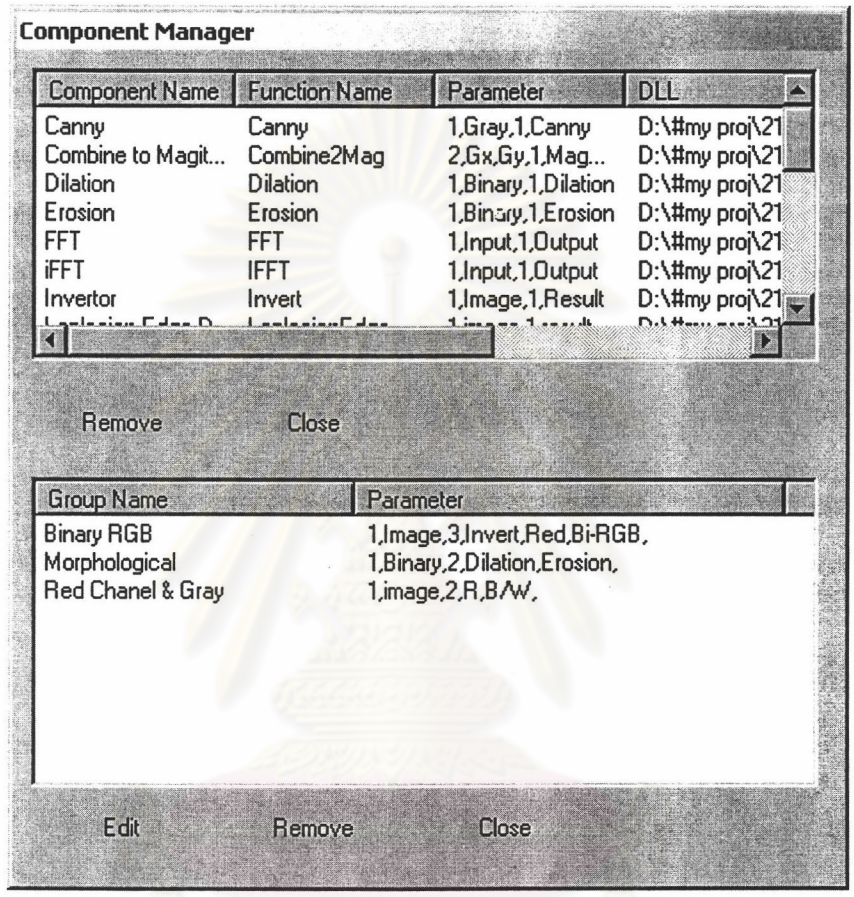
รูปที่ ข.6 เมนูย่อยของเมนู Component

- เมนูย่อย Data ใช้ในเลือกไอเท็มประเภทข้อมูลเพื่อนำมาวางบนบริเวณบรรณาธิกร
- เมนูย่อย Process ใช้ในเลือกไอเท็มประเภทฟังก์ชันประมวลผลภาพเพื่อนำมาวางบนบริเวณบรรณาธิกร โดยฟังก์ชันประมวลผลภาพจะปรากฏเป็นเมนูย่อยของเมนู Process
- เมนูย่อย Group ใช้ในเลือกไอเท็มประเภทกลุ่มฟังก์ชันประมวลผลภาพที่ผู้ใช้สร้างขึ้นเพื่อนำมาวางบนบริเวณบรรณาธิกร โดยกลุ่มฟังก์ชันประมวลผลภาพจะปรากฏเป็นเมนูย่อยของเมนู Group
- เมนูย่อย Create new group... ใช้สำหรับสร้างกลุ่มฟังก์ชันประมวลผลภาพซึ่งมีลักษณะหน้าจอการทำงานดังในรูปที่ ข.7 โดยก่อนเลือกเมนูนี้ต้องสร้างกราฟกระแสข้อมูลที่สมบูรณ์ก่อน



รูปที่ ข.7 หน้าต่างที่ใช้สร้างกลุ่มฟังก์ชันประมวลผลภาพ

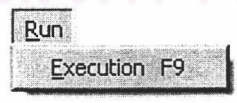
- เมนูย่อย Import Plug-ins ใช้ในการนำเข้าส่วนประกอบซอฟต์แวร์เพื่อเพิ่มฟังก์ชันประมวลผลภาพให้กับเครื่องมือ
- เมนูย่อย Manage Component... ใช้สำหรับจัดการไอเท็มของเครื่องมืออันได้แก่ Process Item และ Group Item ซึ่งมีหน้าจอการทำงานดังแสดงในรูปที่ ข.8



รูปที่ ข.8 หน้าต่างที่ใช้จัดการไอเท็มต่าง ๆ ของเครื่องมือ

เมนู Run

เป็นเมนูที่เกี่ยวกับการประมวลผลภาพซึ่งจะประกอบด้วยเมนูย่อย 1 เมนูดังรูปที่ ข.9



รูปที่ ข.9 เมนูย่อยของเมนู Run

- เมนูย่อย Execution ใช้ในการสั่งงานให้เครื่องมือทำการประมวลผลภาพตามกราฟกระแสข้อมูลบนบริเวณบรรณาธิกรที่ผู้ใช้ออกแบบ

บริเวณบรรณาธิกร

บริเวณบรรณาธิกรคือพื้นที่ว่างเปล่าตรงกลางหน้าต่างหลัก ซึ่งผู้ใช้จะใช้พื้นที่บริเวณส่วนนี้สำหรับวางและจัดเรียงไอเท็มต่าง ๆ เพื่อสร้างกราฟกระแสข้อมูลที่แสดงการทำงานของขั้นตอนวิธีในการประมวลผลภาพทั้งกระบวนการ

แถบแสดงสถานะ

แถบแสดงสถานะจะประกอบด้วยส่วนแสดงสถานะของเครื่องมือซึ่งใช้สำหรับแสดงสถานะการทำงานของเครื่องมือ

ข.3 ขั้นตอนการใช้เครื่องมือสำหรับออกแบบขั้นตอนวิธีในการประมวลผลภาพ

ในหัวข้อนี้เป็นการอธิบายขั้นตอนการใช้งานเครื่องมือสำหรับออกแบบขั้นตอนวิธีในการประมวลผลภาพโดยการวาดกราฟกระแสข้อมูล ซึ่งมีขั้นตอนดังนี้

1. เลือกฟังก์ชันประมวลผลภาพที่ต้องการจากเมนูย่อย Process ในเมนูหลัก Component แล้วคลิกเมาส์บนบริเวณบรรณาธิกรเพื่อวาง Process Item บนหน้าจอ
2. เลือกไอเท็มข้อมูลจากเมนูย่อย Data ในเมนูหลัก Component แล้วคลิกเมาส์บนบริเวณบรรณาธิกรเพื่อวาง Data Item ต้นทางบนหน้าจอเพื่อใช้เก็บภาพต้นฉบับ
3. คลิกเมาส์บริเวณพอร์ตส่งออกของ Data Item แล้วคลิกเมาส์บริเวณพอร์ตนำเข้าของ Process Item จะปรากฏเส้นเชื่อมต่อระหว่างพอร์ตที่คลิกเมาส์ ขั้นตอนนี้เป็นการกำหนดความสัมพันธ์ระหว่างไอเท็มสองภาพ
4. เลือกไอเท็มข้อมูลจากเมนูย่อย Data ในเมนูหลัก Component แล้วคลิกเมาส์บนบริเวณบรรณาธิกรเพื่อวาง Data Item ปลายทางบนหน้าจอเพื่อใช้เก็บภาพผลลัพธ์
5. คลิกเมาส์บริเวณพอร์ตส่งออกของ Process Item แล้วคลิกเมาส์บริเวณพอร์ตนำเข้าของ Data Item จะปรากฏเส้นเชื่อมต่อระหว่างพอร์ตที่คลิกเมาส์
6. ทำซ้ำข้อ 2 ถึง 3 และ 4 ถึง 5 จนครบทุกพอร์ตนำเข้าและส่งออกของ Process Item
7. โหลดภาพต้นฉบับไว้ใน Data Item ต้นทางโดยคลิกเมาส์ปุ่มขวาที่ Data Item ต้นทางจะปรากฏเมนูแบบผุดขึ้นสำหรับ Data Item ขึ้นดังรูปที่ ข.10 จากนั้นให้เลือกเมนู Load Data จะปรากฏหน้าต่างให้เลือกแฟ้มข้อมูลภาพที่ต้องการจะโหลดแล้วกดปุ่ม OK เมื่อโหลดภาพต้นฉบับแล้วสามารถตรวจดูภาพได้โดยการคลิกเมาส์สองครั้งบน Data Item จะมีหน้าต่างแสดงภาพปรากฏขึ้นบนหน้าจอ

8. ทำซ้ำข้อ 7 จนกระทั่งโหลดข้อมูลครบทุก Data Item ต้นทาง

9. ในขณะนี้จะได้กราฟกระแสข้อมูลที่สมบูรณ์ ผู้ใช้สามารถทดลองประมวลผลภาพได้โดยที่เลือกเมนูย่อย Execution จากเมนูหลัก Run แล้วดูผลลัพธ์ที่ได้โดยการคลิกเมาส์สองครั้งที่ไอเท็มปลายทาง

10. เมื่อต้องการบันทึกกราฟกระแสข้อมูลเก็บไว้ใช้ต่อไปในอนาคตให้เลือกเมนู "File -> Save Project..."

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ค
บทความที่ได้รับการคัดเลือกให้นำเสนอในงานประชุมวิชาการ

บทความเรื่อง "VisDIP-Tool: เครื่องมือประมวลผลภาพดิจิทัลบนสภาพแวดล้อมแบบวิชวล" ได้รับการคัดเลือกให้นำเสนอในงานประชุมวิชาการ The 7th National Computer Science and Engineering จัดที่มหาวิทยาลัยบูรพา บางแสน ชลบุรี ประเทศไทย ระหว่างวันที่ 28-30 ตุลาคม พ.ศ. 2546

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

VisDIP-Tool: เครื่องมือประมวลผลภาพดิจิทัลบนสภาพแวดล้อมแบบวิชวล
VisDIP-Tool: A Visual Environment DIP Tool

เจษฎา ชินอานุภาพ วิวัฒน์ วัฒนาวุฒิ และ นงลักษณ์ โควาวิสารัช
Jedsada Chin-anuparb, Wiwat Vatanawood and Nongluk Covavisaruch
Department of Computer Engineering, Faculty of Engineering
Chulalongkorn University, Bangkok 10330, Thailand
e-mail: Jedsada.C@Student.chula.ac.th, wiwat@chula.ac.th, nongluk.c@chula.ac.th

บทคัดย่อ

VisDIP-Tool คือเครื่องมือที่ใช้สำหรับทดลองเพื่อสนับสนุนการวิจัยและเพื่อการเรียนการสอนทางด้านการประมวลผลภาพดิจิทัล ผู้วิจัยสามารถทดสอบขั้นตอนวิธีในการประมวลผลภาพได้ง่ายและสะดวกรวดเร็วโดยการประกอบฟังก์ชันการประมวลผลภาพเป็นกราฟกระแสข้อมูลดีไอพี (DIP Dataflow graph) โดยใช้วิธีลากแล้วปล่อย ผู้วิจัยสามารถประมวลผลภาพและแสดงผลลัพธ์ตามกราฟกระแสข้อมูลดีไอพีที่ได้ออกแบบไว้ VisDIP-Tool พัฒนาขึ้นโดยใช้แนวคิดการพัฒนาเชิงส่วนประกอบซอฟต์แวร์ซึ่งรองรับการเพิ่มฟังก์ชันการประมวลผลภาพได้โดยสร้างเป็นส่วนประกอบซอฟต์แวร์แล้วนำมาประกอบเข้ากับเครื่องมือ

คำสำคัญ: การประมวลผลภาพดิจิทัล, วิชวล, ส่วนประกอบซอฟต์แวร์, เครื่องมือ

Abstract

VisDIP-Tool is a visual environment tool that supports Research and Instruction in digital image processing area. Researchers can quickly test their works on a series of image processing algorithms by drawing a DIP dataflow graph in a drag-and-drop manner. The VisDIP-Tool provides a command to execute the whole algorithms to show the results. In DIP dataflow graph, image processing functions and data storages are represented as function blocks and data blocks. This VisDIP-tool is designed and implemented using component-based software concept which supports the deployment of a new pluggable digital image processing components.

Key-Words: Digital image processing, visual, component, tool

1. บทนำ

การทำงานวิจัยทางด้านการประมวลผลภาพส่วนใหญ่จะเป็นการออกแบบขั้นตอนวิธีในการประมวลผลภาพ (Image Processing Algorithm) โดยมักมีวิธีการประมวลผลภาพ (Image Processing Method) เรียงกันแบบอนุกรมหรือขนานซึ่งความซับซ้อนขึ้นกับแต่ละขั้นตอนวิธี ดังนั้นการออกแบบขั้นตอนวิธีที่จะแก้ปัญหาหนึ่ง ๆ ในอดีตมาจนกระทั่งปัจจุบันจึงค่อนข้างลำบากและเสียเวลามากเนื่องจากต้องเขียนโปรแกรมและทดสอบขั้นตอนวิธีค่อนข้างมาก จากปัญหาข้างต้นจึงได้มีการพัฒนาเครื่องมือสำหรับใช้ในงานประมวลผลขึ้นโดยมุ่งที่จะพัฒนาเพื่อให้ใช้งานง่ายและสะดวกสบายมากขึ้น

ในปีค.ศ. 1992 Thomas J. Olson และคณะ ได้เสนอการใช้สภาพแวดล้อมแบบวิชวลเพื่อใช้สำหรับงานด้านทรรศนะคอมพิวเตอร์ (Computer Vision) โดยพัฒนาโครงงาน MAVIS [1] ขึ้น โดยมุ่งเน้นที่การสร้างและทดสอบขั้นตอนวิธีในการประมวลผลภาพเพื่อใช้กับโปรแกรมประยุกต์แบบทันกาล เช่นระบบยานพาหนะ

อัตโนมัติและหุ่นยนต์ โดยผู้ใช้สามารถตรวจดูผลลัพธ์ที่ขั้นตอนต่าง ๆ ได้ โครงงานนี้มีส่วนติดต่อผู้ใช้แบบกราฟิกที่ใช้งานง่าย ผู้ใช้สามารถจัดเรียงฟังก์ชันตามแนวคิดโดยการใช้เมาส์เลือกฟังก์ชันประมวลผลภาพมาวาง แต่โครงงานนี้ยังขาดความยืดหยุ่นในการเพิ่มฟังก์ชันประมวลผลภาพให้กับซอฟต์แวร์และน่าจะมีการแสดงรายละเอียดของฟังก์ชันประมวลผลภาพบนหน้าจอที่มากกว่านี้ เช่น พารามิเตอร์ของฟังก์ชัน เป็นต้น

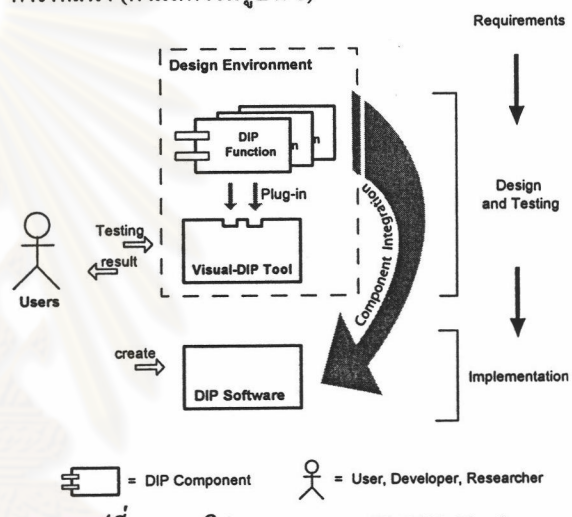
ในปีค.ศ. 1998 บริษัท The Image Source ได้ออกผลิตภัณฑ์ชื่อ Ad Oculos [2] ขึ้น ซึ่งมีจุดประสงค์หลักเพื่อใช้ในการเรียนการสอนโดยแสดงให้เห็นแต่ละขั้นตอนการทำงานของการวิเคราะห์ภาพ Ad Oculos เป็นเครื่องมือที่สามารถวาดแผนผังการประมวลผลภาพโดยนำฟังก์ชันประมวลภาพมาวางเรียงต่อกันแล้วสั่งให้เครื่องมือทำการประมวลผลภาพตามลำดับในแผนผัง และยังสามารถเพิ่มฟังก์ชันประมวลผลภาพได้ อย่างไรก็ดี เครื่องมือนี้มีส่วนติดต่อผู้ใช้ที่น่าจะปรับปรุงได้เพราะการแสดงทิศทางการเคลื่อนที่ของข้อมูลไม่ชัดเจน และภาพที่ใช้แทนฟังก์ชันไม่ได้แสดงพารามิเตอร์รวมทั้งเส้นเชื่อมต่อฟังก์ชันก็ไม่มีทิศทางที่เป็นระบบ

งานวิจัยนี้ได้เลือกนำข้อดีในงานของ Thomas J. Olson และคณะ [1] ที่ใช้สภาพแวดล้อมแบบวิชวล รวมถึงข้อดีของ Ad Oculos [2] ที่สามารถเพิ่มฟังก์ชันการประมวลผลภาพมาใช้ในการออกแบบและพัฒนา VisDIP-Tool เพื่อช่วยแก้ปัญหาและลดขั้นตอนยุ่งยากในการออกแบบและทดลองทางด้านการประมวลผลภาพด้วยสภาพแวดล้อมแบบวิชวลและความสามารถในการเพิ่มส่วนประกอบซอฟต์แวร์ในอนาคต

2. แนวทางการออกแบบ VisDIP-Tool

จากรูปที่ 1 ได้แสดงแนวคิดในการออกแบบ VisDIP-Tool เพื่อใช้งานในขั้นตอนการออกแบบและทดสอบโดยมีเป้าหมายให้ใช้งานง่ายและสามารถเพิ่มขีดความสามารถได้อีกในอนาคต (คือเพิ่มฟังก์ชันการประมวลผลภาพได้อีก) VisDIP-Tool จึงถูกออกแบบและสร้างบนพื้นฐานของส่วนประกอบซอฟต์แวร์ (Software component) [3] โดยเครื่องมือนี้จะทำหน้าที่เป็นตัวกลางเพื่อให้ผู้ใช้เรียกใช้งานฟังก์ชันประมวลภาพที่อยู่ในส่วนประกอบซอฟต์แวร์และดูผลลัพธ์ผ่านทางส่วนติดต่อผู้ใช้แบบกราฟิก

เมื่อทดสอบขั้นตอนวิธีตามที่ได้ออกแบบไว้จนได้ผลเป็นที่พอใจแล้ว ผู้ใช้ยังสามารถนำส่วนประกอบซอฟต์แวร์ที่ใช้กับ VisDIP-Tool ไปพัฒนารวมกันเป็นโปรแกรมประยุกต์ต่อไปได้ซึ่งช่วยลดภาระในขั้นตอนการพัฒนา (ดังแสดงในรูปที่ 1)

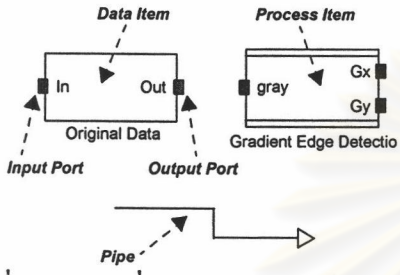


รูปที่ 1. แนวคิดการออกแบบ VisDIP-Tool

2.1 สภาพแวดล้อมแบบวิชวล

งานวิจัยนี้ได้เสนอกราฟกระแสข้อมูลดีไอพีเพื่อใช้แสดงขั้นตอนวิธีในการประมวลผลภาพในส่วนติดต่อผู้ใช้แบบวิชวลของ VisDIP-Tool โดยการแทนฟังก์ชันและข้อมูลต่าง ๆ เป็นรูปภาพบล็อกซึ่งเรียกว่า *ไอเท็ม* โดยข้อมูลจะถูกแทนด้วย *Data Item* และฟังก์ชันประมวลภาพจะถูกแทนด้วย *Process Item* ซึ่งมีข้อความแสดงชื่อของข้อมูลและฟังก์ชันประมวลภาพบริเวณด้านล่าง (ดังแสดงในรูปที่ 2) แต่ละไอเท็มจะมี *Port* ซึ่งแสดงด้วยรูปสี่เหลี่ยมเล็ก ๆ ติดอยู่ด้านข้างทั้งด้านซ้ายและด้านขวา โดยมีชื่อพอร์ตกำกับอยู่ด้านข้างเพื่อใช้แสดงพารามิเตอร์ของฟังก์ชันรวมทั้งเป็นช่องทางสื่อสารข้อมูลเข้า-ออก

โดย Port แบ่งออกเป็น 2 ประเภทคือ *พอร์ตนำเข้า (Input Port)* อยู่ทางด้านซ้ายของไอเท็ม และ *พอร์ตส่งออก (Output Port)* อยู่ทางด้านขวาของไอเท็ม การส่งข้อมูลระหว่างไอเท็มจะผ่านทาง Port โดยสามารถกำหนดเส้นทางการไหลของข้อมูลด้วย *Pipe* ซึ่งแสดงเป็นเส้นที่มีหัวลูกศรเพื่อบอกทิศทางการไหลของข้อมูล ดังแสดงในรูปที่ 2



รูปที่ 2. สัญกรณ์ที่ใช้ในกราฟกระแสข้อมูลดีไอพี

2.2 ส่วนประกอบซอฟต์แวร์

เนื่องจากงานวิจัยนี้ต้องการให้เครื่องมือสามารถเพิ่มเติมฟังก์ชันประมวลผลภาพได้ในอนาคต จึงออกแบบ VisDIP-Tool โดยใช้แนวคิดในการพัฒนาเชิงส่วนประกอบซอฟต์แวร์ [4] โดยออกแบบข้อกำหนดในการสร้างส่วนประกอบซอฟต์แวร์ทางด้านการประมวลผลภาพเพื่อใช้งานกับเครื่องมือเรียกว่าส่วนประกอบดีไอพี (ซึ่งจะกล่าวต่อไปในหัวข้อที่ 3.1) ดังนั้น ส่วนประกอบซอฟต์แวร์ที่สร้างขึ้นตามข้อกำหนดนี้สามารถนำมาประกอบเข้ากับ VisDIP-Tool แล้วใช้งานได้ทันที

ฟังก์ชันประมวลผลภาพบางฟังก์ชันต้องให้ผู้ใช้กำหนดค่าคงที่เพื่อกำหนดสมบัติบางอย่าง เช่น ค่าขีดแบ่งสำหรับฟังก์ชัน Thresholding ซึ่งค่าคงที่นี้ไม่เกี่ยวข้องกับระบบจึงกำหนดให้สร้างเป็นหน้าต่างโต้ตอบแบบผุดขึ้น (Pop-up window dialog) โดยสร้างรวมเป็นส่วนหนึ่งอยู่ในส่วนประกอบดีไอพี ฟังก์ชันประมวลผลภาพจะใช้หน้าต่างนี้เพื่อรับค่าคงที่จากผู้ใช้ (ดังรูปที่ 6 เมื่อฟังก์ชัน Thresholding ถูกสั่งให้ทำงานก็จะผุดหน้าต่างโต้ตอบขึ้นมารับค่าจากผู้ใช้)

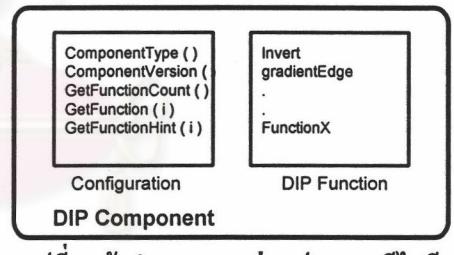
3. ระบบของ VisDIP-Tool

ระบบหลักของ VisDIP-Tool ถูกออกแบบและสร้างโดยใช้การโปรแกรมเชิงวัตถุ (OOP) เนื่องจากฟังก์ชันประมวลภาพแต่ละฟังก์ชันมีลักษณะการทำงานที่แยกกันโดยอิสระ การนำ OOP มาพัฒนา VisDIP-Tool ช่วยให้ง่ายต่อการออกแบบส่วนติดต่อผู้ใช้แบบวิชวลและออกแบบวิธีการประมวลผล

ระบบของ VisDIP-Tool แบ่งออกได้เป็น 4 ส่วนคือ ระบบจัดการส่วนประกอบ ระบบสร้างและแก้ไขกราฟ ระบบแสดงและแก้ไขข้อมูล และระบบการประมวลผล

3.1 ระบบจัดการส่วนประกอบ

VisDIP-Tool ใช้ฐานข้อมูลเชิงสัมพันธ์สำหรับจัดเก็บข้อมูลส่วนประกอบดีไอพีที่นำมาประกอบเข้ากับ VisDIP-Tool โดยกำหนดให้สร้างส่วนประกอบดีไอพีเป็นแฟ้ม DLL ซึ่งสามารถพัฒนาขึ้นได้จากหลายภาษา ได้แก่ Visual C++, Borland C/C++ Builder และ Delphi แต่ต้องสร้างตามข้อกำหนดเพื่อนำมาใช้งานกับ VisDIP-Tool



รูปที่ 3. ข้อกำหนดของส่วนประกอบดีไอพี

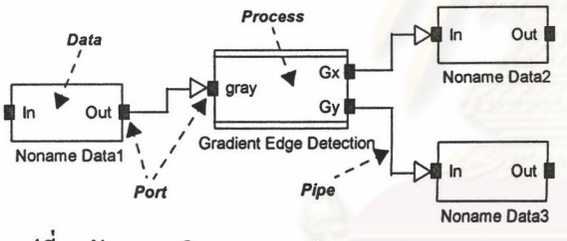
รูปที่ 3 แสดงโครงสร้างของส่วนประกอบดีไอพีที่สามารถนำมาใช้กับ VisDIP-Tool ซึ่งมีส่วนที่สำคัญคือ โครงแบบ (Configuration) โดยมีฟังก์ชันเพื่อระบุรายละเอียดต่าง ๆ ของส่วนประกอบดีไอพีได้แก่ *ComponentType* สำหรับระบุชนิดของส่วนประกอบ เช่น ชนิดฟังก์ชันประมวลผลภาพ *ComponentVersion* สำหรับบอกรุ่นของส่วนประกอบ *GetFunctionCount* และ *GetFunction* สำหรับแสดงฟังก์ชันประมวลผล

ภาพที่มีอยู่ในส่วนประกอบดีไอพี *GetFunctionHint* ใช้แสดงคำแนะนำการใช้ฟังก์ชัน

3.2 ระบบสร้างและแก้ไขกราฟ

ระบบสร้างและแก้ไขกราฟของ VisDIP-Tool ใช้สำหรับออกแบบขั้นตอนวิธีในการประมวลผลภาพ โดยจะมี*บริเวณบรรณาธิกร (Editor area)* สำหรับสร้างและแก้ไขกราฟกระแสข้อมูลดีไอพีซึ่งการสร้างจะเป็นแบบวิชวล ระบบนี้สามารถสร้างวัตถุต่าง ๆ ขึ้นมาวางบนบริเวณบรรณาธิกรเพื่อใช้แสดงฟังก์ชันประมวลผลภาพข้อมูล และ เส้นทางการไหลของข้อมูล

วัตถุที่วางบนบริเวณบรรณาธิกรมีทั้งหมด 4 ชนิด และแบ่งออกเป็น 2 กลุ่ม คือ *วัตถุไอเท็ม (Item Object)* ได้แก่ Process และ Data กับ *วัตถุช่องทาง (Channel Object)* ได้แก่ Port และ Pipe (ดังแสดงในรูปที่ 4) วัตถุเหล่านี้ถูกออกแบบให้มีคุณสมบัติในการแสดงภาพกราฟิกบนหน้าจอและสามารถส่งข้อมูลระหว่างกัน



รูปที่ 4. วัตถุบนบริเวณบรรณาธิกรของ VisDIP-Tool

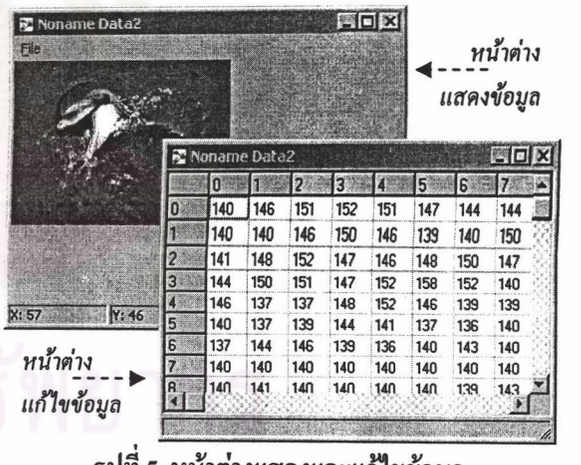
วัตถุไอเท็มถูกใช้เป็นจุดยอด (Vertices) ของกราฟ โดย Process ทำหน้าที่เป็นตัวแทนฟังก์ชันประมวลผลภาพ Data ใช้แทนข้อมูล

วัตถุช่องทางถูกใช้เพื่อเป็นช่องทางในการส่งข้อมูลระหว่างไอเท็มโดย Port ใช้แทนพารามิเตอร์ของฟังก์ชัน และ Pipe ใช้เป็นเส้นเชื่อม (Edge) ระหว่างไอเท็มเพื่อกำหนดและแสดงเส้นทางรวมทั้งทำหน้าที่ลำเลียงข้อมูลระหว่าง Port

ผู้ใช้สามารถสร้างกราฟกระแสข้อมูลดีไอพีได้โดยเลือกฟังก์ชันประมวลภาพที่ต้องการจากเมนู "Component -> Process" และเลือก Data จากเมนู "Component -> Data" แล้วนำมาวางบนบริเวณบรรณาธิกรโดย Port จะถูกสร้างขึ้นมาโดยอัตโนมัติ จากนั้นผู้ใช้ต้องสร้าง Pipe โดยคลิกเมาส์เพื่อสร้างเส้นเชื่อมระหว่างพอร์ตนำเข้าและพอร์ตส่งออกของแต่ละวัตถุไอเท็มเพื่อสร้างกราฟกระแสข้อมูลดีไอพีที่สมบูรณ์ ผู้ใช้สามารถลบ เคลื่อนย้าย รวมทั้งดูรายละเอียดวัตถุบนบริเวณบรรณาธิกรได้ และสามารถ บรรจุ (Load) บันทึก (Save) ดูและแก้ไขข้อมูลจาก Data ได้ ซึ่งกราฟกระแสข้อมูลดีไอพีทั้งหมดบนบริเวณบรรณาธิกรสามารถบันทึกไว้เป็นแฟ้มโครงการ (Project File) เพื่อเปิดใช้งานได้ในภายหลัง

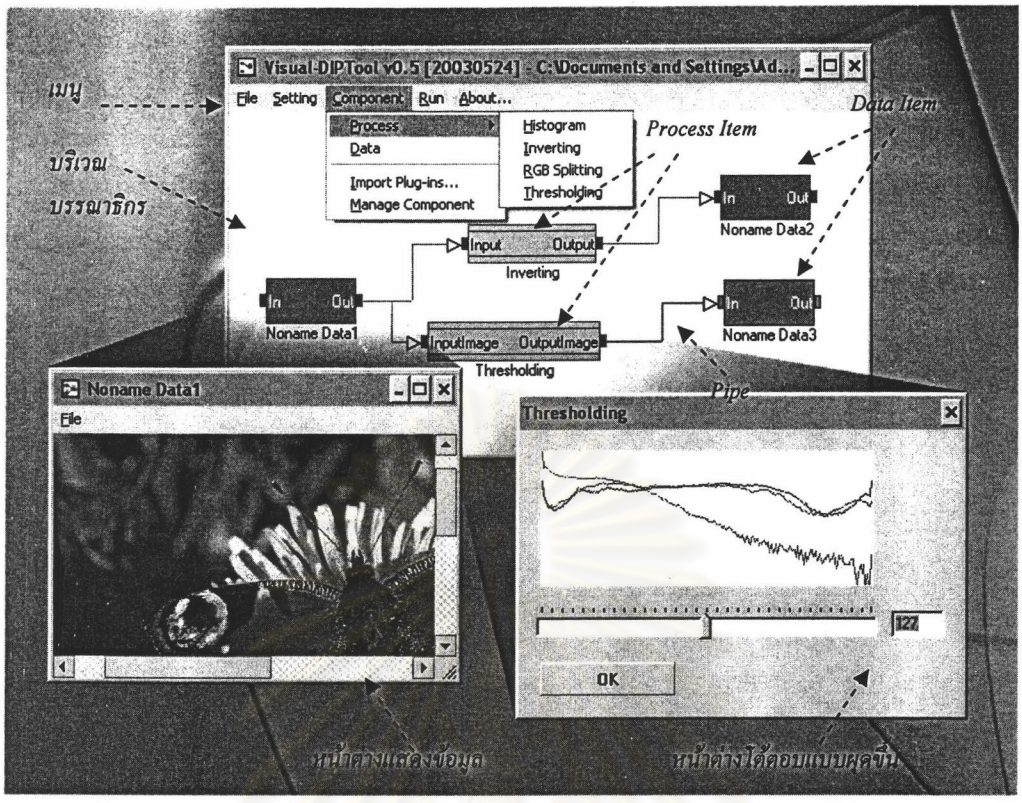
3.3 ระบบแสดงและแก้ไขข้อมูล

ด้วยเครื่องมือ VisDIP-Tool ผู้ใช้สามารถตรวจสอบผลลัพธ์จากการประมวลผลภาพได้ทันที โดยการดูและแก้ไขข้อมูลใน Data Object ด้วยหน้าต่างแสดงและแก้ไขข้อมูล (ดังรูปที่ 5) การแสดงผลมี 2 แบบ ได้แก่ การแสดงภาพบิตแมปและการแสดงตารางตัวเลข



รูปที่ 5. หน้าต่างแสดงและแก้ไขข้อมูล

การแสดงภาพบิตแมปคือการนำข้อมูลใน Data มาแสดงเป็นภาพบิตแมปบนหน้าต่างแสดงภาพโดยรองรับการแสดงภาพขาว/ดำ ภาพระดับเทาและภาพสี การแสดงตารางตัวเลขคือการนำข้อมูลใน Data มาแสดงผลเป็นตารางตัวเลข โดยแต่ละช่องจะแสดงเป็นตัวเลขแทนค่าความสว่างหรือค่าสีของจุดภาพ (Pixel) ผู้ใช้สามารถ



รูปที่ 6. ส่วนติดต่อผู้ใช้แบบวิชวลของ VisDIP-Tool

แก้ไขข้อมูลในตารางแสดงค่าตัวเลขนี้ได้โดยการป้อนตัวเลขแทนที่ตัวเลขเดิมในแต่ละช่อง เพื่อใช้ในการทดลองหรือทดสอบ Process ใดๆ

Data สามารถบรรจุข้อมูลจากแฟ้มประเภท JPEG และ BMP นอกจากนี้ยังสามารถเพิ่มชนิดแฟ้มข้อมูลประเภทอื่นด้วยการเพิ่มส่วนประกอบซอฟต์แวร์ให้กับ VisDIP-Tool จึงทำให้ Data Object สามารถบรรจุข้อมูลจากแฟ้มประเภทอื่นเพิ่มเติมได้อีก

VisDIP-Tool รองรับข้อมูลได้สูงสุดสามมิติจึงสามารถทำงานกับชุดข้อมูลภาพได้เช่น ภาพวีดิทัศน์ ชุดภาพตัดขวางเอ็มอาร์ไอ เป็นต้น และรองรับข้อมูลภาพในแบบจำลองสี [5] แบบต่างๆ เช่น ภาพสี RGB ภาพสี HIS เป็นต้น

3.4 ระบบการประมวลผล

การประมวลผลของเครื่องมือ VisDIP-Tool คือการประมวลผลภาพตามกราฟกระแสข้อมูลดีไอพีที่ผู้ใช้ออกแบบไว้โดยระบบจะเรียกใช้ฟังก์ชันประมวลผลภาพในส่วนประกอบดีไอพีให้ทำงาน ซึ่งการประมวลผลจะเกิดขึ้นที่ Process

3.4.1 การประมวลผลภาพ

เมื่อผู้ใช้สั่งระบบให้ประมวลผลภาพตามกราฟกระแสข้อมูลดีไอพีที่ออกแบบไว้บนบริเวณบรรณาธิกร ข้อมูลจะถูกลำเลียงและประมวลผลโดยอัตโนมัติ เนื่องจากวัตถุทุกชนิดที่เป็นส่วนของกราฟกระแสข้อมูลดีไอพีถูกกำหนดพฤติกรรมให้สามารถส่งข้อมูลและประมวลผลได้เองตามการเชื่อมต่อกันของกราฟกระแสข้อมูล โดยขณะที่ข้อมูลถูกส่งผ่านทาง Pipe ถ้ามี Data ต่ออยู่ ข้อมูลจะถูกดักเก็บไว้ที่ Data เพื่อใช้สำหรับตรวจสอบผลลัพธ์

ลำดับการทำงานเริ่มจากระบบสั่งให้ Data คัดลอกข้อมูลออกไปที่พอร์ตส่งออก จากนั้น Port จะส่งข้อมูลผ่านไปยัง Pipe (ซึ่งเป็นตัวเชื่อมต่อระหว่างพอร์ตส่งออก

และพอร์ตนำเข้า) ไปยัง Port ปลายทางซึ่งเป็นพอร์ตนำเข้าของ Process เมื่อ Port ได้รับข้อมูลก็จะสั่งให้ Process ตรวจสอบข้อมูลว่าได้รับครบทุกพอร์ตนำเข้าหรือไม่ ถ้าได้รับจนครบแล้ว Process จะดึงข้อมูลจากทุกพอร์ตนำเข้าไปประมวลผลโดยทำการโหลดส่วนประกอบดีไอพีเข้าสู่ในหน่วยความจำแล้วส่งข้อมูลให้กับฟังก์ชันเพื่อประมวลผลข้อมูล และเมื่อประมวลผลเสร็จ Process จะลบส่วนประกอบดีไอพีออกจากหน่วยความจำแล้วส่งผลลัพธ์ออกไปยังพอร์ตส่งออก และเป็นเช่นนี้ซ้ำไปเรื่อย ๆ จนกระทั่งการประมวลผลเสร็จครบทั้งกราฟ

3.4.2 การแสดงสถานะ

VisDIP-Tool สามารถแสดงสถานะของวัตถุต่าง ๆ ในขณะประมวลผลภาพได้โดยแสดงสัญญาณสีและข้อความเพื่อแสดงให้ผู้ใช้รู้ว่าวัตถุไอเท็มตัวใดกำลังทำงานอยู่ หรือได้พบข้อผิดพลาดที่วัตถุไอเท็มตัวใด ปัญหาและข้อผิดพลาดต่าง ๆ ของฟังก์ชันประมวลผลภาพในส่วนประกอบดีไอพีจะไม่ถูกพบจนกว่า Process จะถูกสั่งให้ทำงานในขณะประมวลผลเนื่องจากส่วนประกอบนั้นมีคุณสมบัติเป็นเสมือนกล่องดำซึ่งไม่สามารถทำการแก้ไขได้

VisDIP-Tool สามารถแสดงสถานะการทำงานและข้อผิดพลาดที่ได้รับฟังก์ชันประมวลผลภาพในส่วนประกอบดีไอพีในขณะกำลังประมวลผลดังนี้

1. Processing: ระบบจะแสดงไอเท็มที่กำลังทำงานโดยการเปลี่ยนสีขอบของไอเท็มเป็นสีแดง

2. Processing completed: ระบบจะแสดงไอเท็มที่ทำงานเสร็จแล้วโดยการเปลี่ยนสีขอบของไอเท็มเป็นสีเขียว

3. Processing error: ระบบจะแจ้งข้อความผิดพลาดขึ้นเมื่อ Process ไม่สามารถประมวลผลข้อมูลได้

4. Processing done: ระบบจะแสดงข้อความบอกผู้ใช้เมื่อการประมวลผลเสร็จสิ้น

4. การทดลอง

ในงานวิจัยนี้ได้ทำการทดลองนำฟังก์ชันประมวลผลภาพที่เขียนโดยใช้ภาษา Delphi และ C/C++ ที่สร้างเป็นส่วนประกอบซอฟต์แวร์ เช่น Sobel Edge Detection, Thresholding, RGB to Gray Convertor, Inverting มาประกอบเข้ากับ VisDIP-Tool เพื่อทำงานร่วมกันและได้ทดลองออกแบบขั้นตอนวิธีในการประมวลผลภาพโดยใช้เครื่องมือนี้สร้างกราฟกระแสข้อมูลดีไอพีขึ้นมาแล้วสั่งให้เครื่องมือทำการประมวลผลภาพ ซึ่งได้ผลลัพธ์ถูกต้องตามขั้นตอนที่ได้ออกแบบไว้

5. บทสรุปและวิจารณ์

ในงานวิจัยนี้ได้เสนอการพัฒนาเครื่องมือสำหรับช่วยสนับสนุนการทดลองสำหรับการวิจัยและการเรียนการสอนทางด้านการประมวลผลภาพดิจิทัลให้มีความสะดวกและง่าย เครื่องมือนี้ให้ผู้ใช้สามารถจัดเรียงฟังก์ชันประมวลภาพตามขั้นตอนวิธีในการประมวลผลภาพที่ได้ออกแบบไว้แล้วสั่งให้เครื่องมือประมวลผลเพื่อดูผลลัพธ์ เครื่องมือนี้สามารถเพิ่มเติมฟังก์ชันและแฟ้มข้อมูลภาพชนิดใหม่ได้โดยการสร้างเป็นส่วนประกอบดีไอพีแล้วนำมาประกอบเข้ากับเครื่องมือก็สามารถใช้งานได้ทันทีทำให้มีความยืดหยุ่นในการใช้งานสูง

6. เอกสารอ้างอิง

[1] Thomas J. Olson, Nicholas G. Klop, Mark R. Hyett, and Shawn M. Carnell, "MAVIS: A Visual Environment for Active Computer Vision", IEEE Workshop on Visual Languages, September 1992.

[2] Ad Oculos 4.0, DBS Digitale Bildverarbeitung und Systementwicklung GmbH, Germany, <http://dbs-imaging.com/>, 1998.

[3] Steve Robinson and Alex Krassel, "COMponents", Microsoft COM White Papers, 1997.

[4] Dmitrij V. Koznov, "Visual formalisms for component-based software design", TOOLS EE, 2000.

[5] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Prentice Hall, Upper Saddle River, New Jersey 07458, 2001.

ประวัติผู้เขียนวิทยานิพนธ์

นายเจษฎา ชินอานุภาพ เกิดวันอังคารที่ 14 พฤศจิกายน พ.ศ. 2521 สำเร็จการศึกษาระดับปริญญาตรีวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ จากมหาวิทยาลัยรังสิต ในปีการศึกษา 2543 และเข้าทำงานกับบริษัท วีไทย ดอท จำกัด ในตำแหน่งโปรแกรมเมอร์เป็นเวลา 8 เดือน แล้วลาศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต ที่คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย