Journal of Computers Vol. 28, No. 5, 2017, pp. 202-209
doi:10.3966/199115992017102805018

A New Algorithm for Medical Image Window Transformation

Zhen-Huan Zhou[1*], Xiao-Jun Wen[1]

[1] School of Computer Engineering, Shenzhen Polytechnic, Shenzhen, China, 518055
zhouzhenhuan@szpt.edu.cn, wxjun@szpt.edu.cn

Received 25 October 2016; Revised 13 June 2017; Accepted 26 June 2017

Abstract. Most medical images are 12-bit grayscale images, Window transformation is an algorithm that can map 12-bits raw data(0-4095) to 8-bits displaying data (0-255). This paper presents a new mapping table, first, suppose ww representatives window width, wl representatives window level, then the windows top (top) = wl+ww/2, the window bottom (bottom) = wl-ww/2. we use the window level (wl) divided raw data (0-4095) into low zone (0-wl), the upper zone (wl-4095). The low zone (bottom-wl) in raw data is mapped to the low zone (0-128) in the displaying data, the upper zone (wl-top) is mapped to the upper zone (128-255), in this way, we may make displaying image equalization. Secondly, we use shift instead of the division in mapping operations, have overcome the division caused by loss of accuracy, and improve images clarity significantly. Finally, without knowing the value of the window, we can calculate the histogram of the image to get a more ideal window value. The experimental results show that these methods can improve the clarity significantly in window transformation..

Keywords: grayscale image, medical image, shift operation, window transformation

1 Introduction

Most of raw data of medical image are 12-bit grayscale images [1-2], such as CT, MRI and so on, and cannot be displayed directly on the computer screen [3-5]. To display medical images, you must map 12-bit raw image data into 8-bit displaying data, that is mapping the original range (0-4095) gray level to the display range (0-255) gray level, this mapping called the window transformation [6-7].

Let x be a value of the original range, y is the corresponding value in the range of displaying, ww and wl represents window width and window level, ym represents the maximum value (255) in the displaying range. Window conversion formula is commonly as follows [8-9]:

$$y(x)=\begin{cases} 0 & x < wl - ww/2 \\ \dfrac{ym}{ww}(x+\dfrac{ww}{2}-wl) & wl-\dfrac{ww}{2} < x < wl+\dfrac{ww}{2} \\ ym & x > wl+\dfrac{ww}{2} \end{cases} \quad \textbf{(1)}$$

This formula is theoretically no errors, but in the actual calculation, this formula would be a loss of precision because of the division [10], and will reduce image sharpness significantly. This formula can be improved in two ways:

Original range will be divided into two parts: the lower zone (bottom-wl) and high zone (lw-4095). Let 128 be the center of display range, display range will be divided into two parts: the lower zone (0-128) and the high zone (128-255) also. Original zone (bottom-wl) is mapped to the display zone (0-128), the original zone (wl-top) is mapped to the display zone (128-255). Before the division operation, the denominator ym is left-shifted 12, after the division operation, the denominator ym is right-shifted 12, in

* Corresponding Author

this way, we may avoid the loss of precision greatly, and improve the clarity of the image displayed.

2 Mapping from the Original Image Data to the Display Image Data

Let ww be the window width and wl be the window level, top be the windows top then $top = wl + ww/2$, $bottom$ be the window bottom and then $bottom = wl - ww/2$. mapping from the original image range to display image can be expressed in Fig. 1.

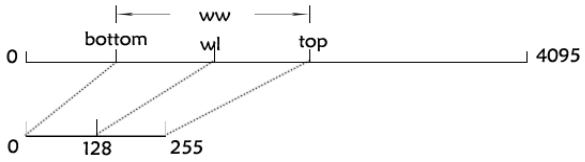

Fig. 1. Mapping from the original image to the display image

3 Mapping and Shift Operations

We use Studio Visual 2010 to create a dialog based application, mapping from the original data to the display data and shift operations C++ code is as follows:

```
#define PIX_BITS       12
#define D_HIGH        255
#define D_LOW         0
#define D_MID                    ((D_HIGH+D_LOW+1)/2)
#define FACTOR_H
         ((D_HIGH+1-D_MID) << PIX_BITS)
#define FACTOR_L
         ((D_MID-D_LOW) << PIX_BITS)
#define MIN_BOTTOM     0
#define MAX_TOP       4095
#define set_window_h(bot,top,fact,x) (x >= top) ? D_HIGH : \
((x <= bot) ? D_MID : ((fact * (x - bot)) >> PIX_BITS) + D_MID)
#define set_window_l(bot,top,fact,x) (x >= top) ? D_MID : \
((x <= bot) ? D_LOW : ((fact * (x - bot)) >> PIX_BITS) + D_LOW)
void WindowTransform(int ww,int wl,unsigned short *win_table)
{
int top = wl + ww/2 ;
int bottom = wl - ww/2 ;
if(top>MAX_TOP){
top = MAX_TOP ;
bottom = MAX_TOP - ww;
}
if(bottom<MIN_BOTTOM)     {
bottom = MIN_BOTTOM ;
top = ww ;
}
int scale_factorh, scale_factorl;
if(top == wl && bottom == wl){
scale_factorh = FACTOR_H;
scale_factorl = FACTOR_L;
}
else if(top == wl){
scale_factorh = FACTOR_H;
scale_factorl = FACTOR_L / (wl - bottom) ;
}
else if(bottom == wl){
scale_factorh = FACTOR_H / (top - wl);
scale_factorl = FACTOR_L;
```

```
}
else{
scale_factorh = FACTOR_H / (top - wl);
scale_factorl = FACTOR_L / (wl - bottom) ;
}
for(int i = 0 ; i <= wl ; i++)
vi_win_table[i] = set_window_l(bottom, wl, scale_factorl, i);
for(int i = wl ; i <= MAX_TOP ; i++)
vi_win_table[i] = set_window_h(wl, top, scale_factorh, i);
}
```

4 Medical image File Structure and Display

Medical image files are generally composed of two parts of the file header and image data cannot display directly, in order to display image, we must build the Device Independent Bitmap(DIB) by the use of image data in memory, the DIB consists of a bitmap information header, color table, and pixel arrays.

4.1 Building Bitmap Information Header and Color Table

The DIB is a 8 bit grayscale image with 256 gray levels, so the color table consists of 256 gray level from 0 to 255. It's just the value of the pixel and the RGB of each pixel are equal, that is R=G=B. We use LocalAlloc () function to allocate memory in the heap for the bitmap information header and color table, use the LocalLock () lock this memory, so as to build bitmap information header and color table. The C++ source code as shown below:

```
HANDLE hpbmi;
PBITMAPINFO pbmi;
//////////////////////////////
unsigned int iNumColor=256;
RGBQUAD argbq[256];
for (unsigned int ii=0;ii<iNumColor;ii++)
{
 argbq[ii].rgbRed=ii;
 argbq[ii].rgbGreen=ii;
 argbq[ii].rgbBlue=ii;
 argbq[ii].rgbReserved=0;
}
         if(hpbmi=::LocalAlloc(LMEM_MOVEABLE|LMEM_ZEROINIT,sizeof(BIT
MAPINFOHEADER)+sizeof(RGBQUAD)*iNumColor))
         {
          if(pbmi=(PBITMAPINFO)::LocalLock(hpbmi)){}
          else AfxMessageBox(L"Could not lock memory
block!",MB_OK,0);
         }
         else AfxMessageBox(L"Could not allock memory
block!",MB_OK,0);
         pbmi->bmiHeader.biSize=40L;
         pbmi->bmiHeader.biPlanes=1L;
         pbmi->bmiHeader.biBitCount=8L;
         pbmi->bmiHeader.biSizeImage=(long)256*256;
         pbmi->bmiHeader.biCompression=0L;
         pbmi->bmiHeader.biXPelsPerMeter=0xbc;
         pbmi->bmiHeader.biYPelsPerMeter=0xbc;
         pbmi->bmiHeader.biClrUsed=256;
         pbmi->bmiHeader.biClrImportant=0;
         pbmi->bmiHeader.biWidth=256;
         pbmi->bmiHeader.biHeight=256;
         memcpy(pbmi->bmiColors,argbq,sizeof(RGBQUAD)*iNumColor);
```

4.2 Opening the Medical Image

We use MFC application program base on dialog and open an original image:

```
CFileDialog dlg(TRUE, NULL, "*.img");
if (dlg.DoModal() == IDOK)
{
  filepath=dlg.GetPathName();
  if(mfile.Open(filepath,
CFile::modeRead|CFile::typeBinary|CFile::shareExclusive,&fe1)==NULL)
  {
    AfxMessageBox("Failed to open the image files!");
    return;
  }
}
```

4.3 Reading Medical Image Data and Setting a Mapping Table

Medical images are made of header and pixels. The header is a structure with 1024 bytes, for example, called image_header. We read the file header first to get the window width (m_width) and window level (m_level), then use the WindowTransform() to transforms the image window, to get the mapping table from raw date to displaying data. Finally, we read the raw data rawdata [j] [k], the source codes are as follow:

```
struct image_header image_head;
mfile.Read(&image_head,sizeof(struct image_header));
m_width=image_head.window_width;
m_level=image_head.window_level;
WindowTransform (m_width,m_level,image_table);
mfile.Seek(1024,CFile::begin);
for(int j=0;j<256;j++)
{
  mfile.Read((unsigned short*)rawdata[j],256*sizeof(unsigned short));
}.
mfile.Close();
```

4.4 Conversing from Raw Data to Displaying Data

Raw rawdata [j] [k] are mapped to intermediate data3d [j] [k] by window transformation image_table []. Pixels in the medical image coordinate system are from bottom to top, but DIB images pixels are exactly the opposite, from top to bottom, so data3d [j] [k] is an upside-down image. So we make an upside-down operation, get displaying data pImageSetShowData[256*256] finally.

```
for(int i=0;i<256*256;i++)
   pImageSetShowData[i]=0;
 for(int j=0;j<256;j++)
   for(int k=0;k<256;k++)
   {
        data3d[j][k]=(unsigned
char)image_table[rawdata[j][k]&0x0fff];
    pImageSetShowData[(255-j)*256+k]=data3d[j][k];
   }
```

4.5 Using StretchDIBits () to Display the Image

In order to display the image in the dialog, we drag picture control IDC_STATIC_IMAGE on the dialog. Let window class pointer be CWnd * pImageSetWnd, the device context class pointer for the CDC * pImageSetDC, the rectangle class variables be CRect m_Rectangle, the main codes are as follows:

```
CWnd* pImageSetWnd;
CDC* pImageSetDC;
CRect m_Rectangle;
pImageSetWnd=CWnd::GetDlgItem(IDC_STATIC_IMAGE);
pImageSetDC=pImageSetWnd->GetDC();
pImageSetWnd->GetWindowRect(&m_Rectangle);
::StretchDIBits(pImageSetDC->m_hDC,0,0,  m_Rectangle.right-
m_Rectangle.left-0, m_Rectangle.bottom-m_Rectangle.top-0, 0,0, 256,
 256, pImageSetShowData, pbmi, DIB_RGB_COLORS, SRCCOPY);
pImageSetWnd->ReleaseDC(pImageSetDC);
```

5 Finding Better Window Width and Window Level Values Automatically

In some cases, such as, video captured images, we have no way to get their window width and level values, cannot use window transform to get displaying images. The usual way is to give a value of experience, for example, window width = 800, window level = 100. Of course, we can also use the whole image histogram to obtain better window width and level values, the code as follows:

```
void Autowindow(unsigned short** rawdata, int width, int height)
{
   int hist[4096];
   for(int wi = 0; wi < 4096 ; wi++)
         hist[wi] = 0 ;
         for(int n=0;n<256;n++)
         for(int p=0;p<256;p++)
              hist[rawdata[n][p]&0x0fff] ++ ;
          int win_bot=0,win_top=0;
         int histmax=0,subscript=0;
         for(int q=0;q<4096;q++)
         {
              if(hist[q]>histmax)
              {
          histmax=hist[q];
          subscript=q;
               }
         }
          int mintemp=histmax;
         for(int r=subscript;r<4096;r++)
         {
              if(hist[r]!=0)
              {
          if(hist[r]<=mintemp)
          {
                   mintemp=hist[r];
              win_bot=r;
          }
          else
              break ;
           }
          }
```

```
            int maxtemp=0;
            for(int s=win_bot;s<4096;s++)
            {
           if(hist[s]!=0)
           {
               maxtemp=hist[s];
               win_top=s;
             }
            }
           if(1 == nFir)
          {
             usFirWW = (unsigned short)(win_top - win_bot);
             usFirWL = (unsigned short)(win_top + win_bot)/2;
           }
           else if(2 == nFir)
          {
            usSecWW = (unsigned short)(win_top - win_bot);
            usSecWL = (unsigned short)(win_top + win_bot)/2;
           }
}
```

6 Experimental Results

We use the Lenovo ThinkCentre PC, Windows7 operating system, Visual Studio 2010, to create a dialog-based application. Fig. 2 is the results of window transform without shift operations, Fig. 3 is the results of window transformation after the shift operations. Experimental results show that shift operations avoid the loss of precision caused division and make the image more clearer.

In the case of the values of experience, such as window width =800, window level = 100, may be no good result. Fig. 4 is the window transformation result with the values of experience. Fig. 5 is the result of automatic computation by the whole image histogram. Fig. 5 is much more clearer than Fig. 4.

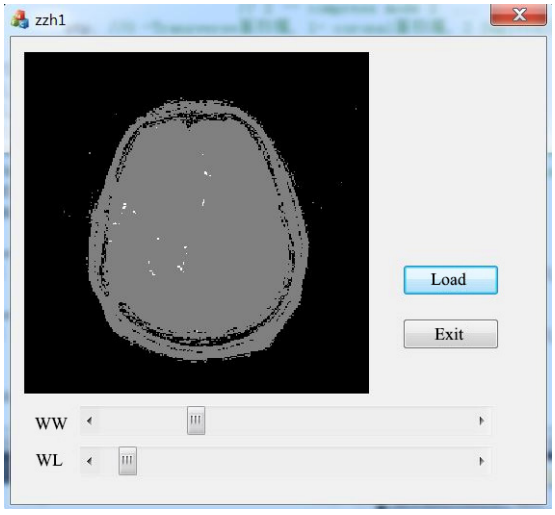

Fig. 2. Window transform without shift operations

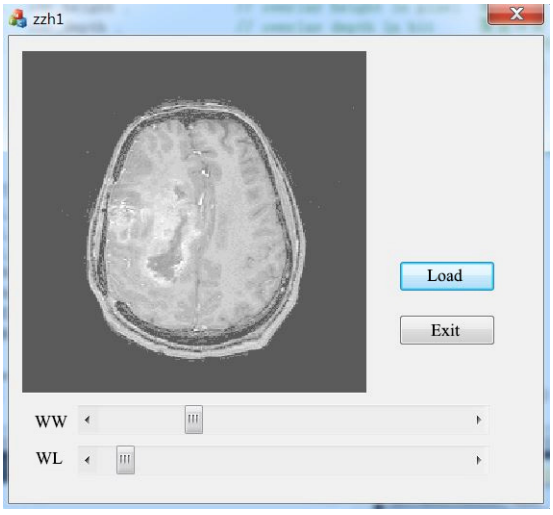

Fig. 3. Window transformation after the shift operations

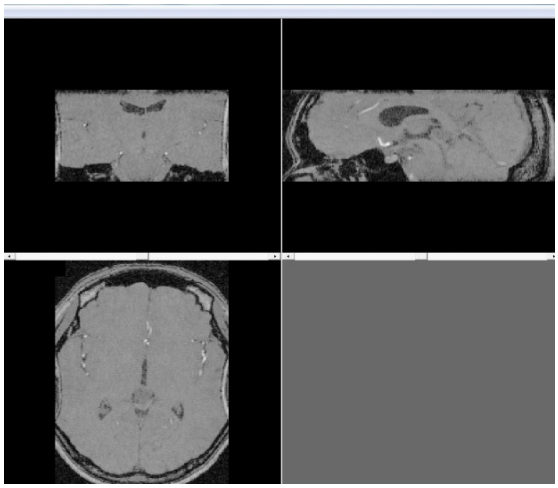

Fig. 4. Result of the values of experience

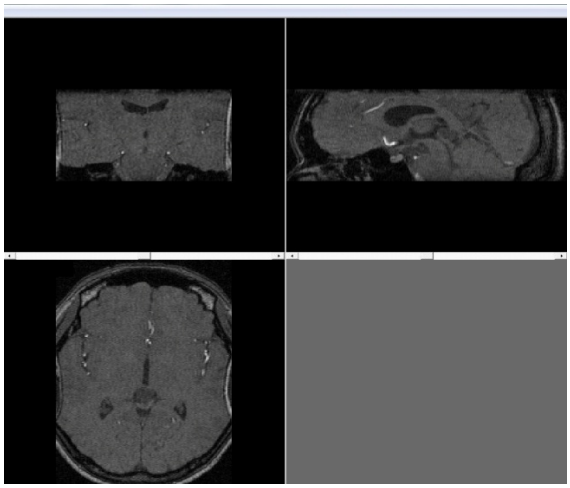

Fig. 5. Result of the whole image histogram

7 Conclusions

Medical image window transform is the basis of displaying, in this paper, the window level of the original image is mapped to the median of the display image and maximized image displaying equalization. Shift operation is used to overcome the division caused by loss of accuracy, so that the image becomes clearer. Followed by a dialog-based program, we discuss how to read medical image files in detail, map from raw data to displaying data, construct a bitmap header and color table, display images and so on. Finally, we demonstrate the above algorithm in C ++ code.

References

[1] L. Liu, Z.H. Jia, J. Yang, K. Nikola, A medical image enhancement method using adaptive thresholding in NSCT domain combined unsharp masking, International Journal of Imaging Systems & Technology 25(3)(2015) 199-205.

[2] W.Z.W. Ismail, K.S. Sim, Contrast enhancement dynamic histogram equalization for medical image processing application, International Journal of Imaging Systems & Technology 21(3)(2011) 280-289.

[3] Fred.N. Kiwanuka, M.H. Wilkinson, Automatic attribute threshold selection for morphological connected attribute filters, Pattern Recognition 53(2016) 59-72.

[4] M. Rodrigo, S. Örjan, Gradient-based enhancement of tubular structures in medical images, Medical Image Analysis 26(1)(2015) 19-29.

[5] H.-T. Wu, J.-W. Huang, Y.-Q. Shi, A reversible data hiding method with contrast enhancement for medical images, Journal of Visual Communication & Image Representation 31 (2015)146-153.

[6] C. Hariton, Recent trends in Medical Image Processing, Computer Science Journal of Moldova 22(2)(2014) 147-154.

[7] J.-J. Wang, Z.-H. Jia, K. Nikola, Medical image enhancement algorithm based on NSCT and the improved fuzzy contrast, International Journal of Imaging Systems & Technology 25(1)(2015) 7-14.

[8] E. Anders, D. Paul, F. Daniel, L. Stephen, Medical image processing on the GPU–Past, Medical Image Analysis 17(8)(2013) 1073-1094.

[9] K. Dagmar, V. Michal, H. Andreas, T. Thomas, Simplified implementation of medical image processing algorithms into a grid using a workflow management system, Future Generation Computer Systems 26(4)(2010) 681-684.

[10] M.S. Ehsan, A.-A. Alireza, R. Roohollah, F.-R., Shahrooz, Taimouri, Web-based interactive 2D/3D medical image

processing and visualization software, Computer Methods & Programs in Biomedicine 98(2)(2010) 172-182.

[11] A. Sam, N. George, L. Sergio, Stereotactic navigation for TAMIS-TME: opening the gateway to frameless, image-guided abdominal and pelvic surgery, Surgical Endoscopy 29(1)(2015) 207-211.

[12] X.Y. Sun, Y.K. Yoon, J. Li, F.D. McKenzie, Automated image-guided surgery for common and complex dental implants, Journal of Medical Engineering & Technology 38(5)(2014) 251-259.

[13] T.P. Kingham, M.A. Scherer, B.W. Neese, L.W. Clements, J.D. Stefansic, W.R. Jarnagin, Image-guided liver surgery: intraoperative projection of computed tomography images utilizing tracked ultrasound, HPB: The Official Journal of the International Hepato Pacreato Biliary Association 14(9)(2012) 594-603.

[14] T. Elserry, H. Anwer, I.N. Esene, Image guided surgery in the management of craniocerebral gunshot injuries, Surgical Neurology International 4(6)(2013) S448-S454.

[15] M. Nau-Hermes, R. Schmitt, M. Becker, W. El-Hakimi, S. Hansen, T. Klenzner, Quality assurance of multiport image-huided minimally invasive surgery at the lateral skull base, BioMed Research International (2014) 1-7.

[16] G. Li, H. Su, G.A. Cole, W.J Shang, K. Harrington, A. Camilo, J.G. Pilitsis, G.S. Fischer, Robotic system for MRI-guided stereotactic neurosurgery, IEEE Transactions on Biomedical Engineering 62(4)(2015) 1077-1088.

[17] L.M. Vigneron, R.C. Boman, J.P. Ponthot, P.A. Robe, S.K. Warfield, J.G. Verly, Enhanced FEM-based modeling of brain shift deformation in image-guided neurosurgery, Journal of Computational & Applied Mathematics 234(7)(2010) 2046-2053.

[18] J.H. Yan, J.C-S. Lim, D.W Townsend, MRI-guided brain PET image filtering and partial volume correction, Physics in Medicine & Biology 60(3)(2015) 1-7.

[19] R.J Zhang, J. Shen, F. Wei, X. Li, A.K. Sangaiah, Medical image classification based on multi-scale non-negative sparse coding, Artificial Intelligence in Medicine 3(2017) 1-8.

[20] Y.T. Chen, A novel approach to segmentation and measurement of medical image using level set methods, Magnetic Resonance Imaging 39(2017) 175-193.

[21] A.A. Kiaei, H. Khotanlou, Segmentation of medical images using mean value guided contour, Medical Image Analysis 17(40)(2017) 111-132.

[22] M. Tóth, L. Ruskó, B. Csébfalvi. Automatic recognition of anatomical regions in three-dimensional medical images, Computers in Biology and Medicine, 76(1)(2016)120-133.