

Standard Deviation:

It is common to find comparison of two bitmaps in Image Processing Development. Comparison of two bitmaps means how each pixel of image1 is different from corresponding pixel of image2 and vice-versa. This is called Standard Deviation. Standard Deviation tells us how many pixels are different in terms of its color in the given two images.

Many software are available on Internet to calculate Standard Deviation, Photoshop is one of them. But we don't need to install any software, it can be done programmatically using the simple, generic and dynamic algorithm given below, in an Windows Console.

Key points of Algorithm:

- It computes the Standard Deviation by calculating the difference of each channel (R,G,B and A) of a pixel.

- The below algorithm is tested with the Photoshop based Standard Deviation. Results are 95% matching.

Note:

The below algorithm is written for .bmp files only. So it supports only .bmp files.

Build Guide:

- Create a Windows Console application project in Microsoft Visual Studio 2003/2005.
- Project will be created with stdafx.h and main.cpp files.
- Copy the below stdafx.h code into the stdafx.h file
- Copy the below StdDev.cpp code into the main.cpp file
- Copy the bitmap files in where your .vcproj is created
- Open Project properties and then go Configuration Properties ---> Debugging ---> Command Arguments
- Write the names of the bitmap files with a space. For example...

TestFile1.bmp TestFile2.bmp
- Build the project and Run the project.
- Results will be displayed on the Console.

stdafx.h

```
#include <windows.h>
#include <stdio.h>
#include <tchar.h>
#include "stdlib.h"
#include "malloc.h"
#include "math.h"
#include "string.h"
```

```
#include "conio.h"
```

StdDev.cpp

```
#include "stdafx.h"
```

```
float StdDev24Bit(unsigned char* img1, unsigned char* img2, int width,
int height)
{
    if ( !img1 || !img2 || !width || !height )
        return -1.0f;

    int          index = 0;
    float sum      = 0;

    for( int y = 0; y < height; ++y )
    {
        for( int x = 0; x < width; ++x )
        {
            index = y * width + x;

            sum +=      pow( float(img1[index+0] - img2[index+0])
, int(2)) +
                    pow( float(img1[index+1] - img2[index+1])
, int(2)) +
                    pow( float(img1[index+2] - img2[index+2])
, int(2)) ;
        }

        // std dev
        long pixels = width*height*3;
        float stdDev = sqrt(float(sum/pixels));
        return stdDev;
    }
}
```

```
float StdDev32Bit(unsigned char* img1, unsigned char* img2, int width,
int height)
{
    if ( !img1 || !img2 || !width || !height )
        return -1.0f;

    int          index = 0;
    float sum      = 0;

    for( int y = 0; y < height; ++y )
    {
        for( int x = 0; x < width; ++x )
        {
            index = y * width + x;

            sum +=      pow( float(img1[index+0] - img2[index+0])
, int(2)) +
                    pow( float(img1[index+1] - img2[index+1])
, int(2)) +
                    pow( float(img1[index+2] - img2[index+2])
, int(2)) ;
        }

        // std dev
        long pixels = width*height*4;
        float stdDev = sqrt(float(sum/pixels));
        return stdDev;
    }
}
```

```

        pow( float(img1[index+2] - img2[index+2])
, int(2)) +
        pow( float(img1[index+3] - img2[index+3])
, int(2)) ;
    }

    // std dev
    long pixels = width*height*4;
    float stdDev = sqrt(float(sum/pixels));
    return stdDev;
}

```

```

int main(int argc, char* argv[])
{
    if ( argc < 2 )
        return 0;

    char drive[ _MAX_DRIVE ];
    char dir[ _MAX_DIR ];
    char fname1[ _MAX_PATH ] = {0};
    char fname2[ _MAX_PATH ] = {0};
    char ext1[ _MAX_EXT ];
    char ext2[ _MAX_EXT ];
    char fname[ _MAX_FNAME ];
    char ext[ _MAX_EXT ];

    _splitpath( argv[0], drive, dir, fname, ext );
    if ( argv[1] == NULL || argv[2] == NULL )
    {
        printf("\n\nOne of the input image file is missing.\n\n");
        printf("Press ENTER to exit...");
        getch();
    }

    _splitpath(argv[1], drive, dir, fname1, ext1);
    _splitpath(argv[2], drive, dir, fname2, ext2);

    bool isBMPImages = false;
    if ( _stricmp(".bmp", ext1)==0 && strcmp(".bmp", ext2)==0 )
        isBMPImages = true;

    if (isBMPImages == false)
        return 0;

    // image1
    FILE* fp1 = fopen(argv[1], "rb");
    if ( !fp1 )
    {
        printf("\n\nOne of the input image file is missing.\n\n");
        printf("Press ENTER to exit...");
        getch();
        return 0;
    }

    // Read image1 header

```

```

BITMAPFILEHEADER bmf1;
fread(&bmf1, sizeof(bmf1), 1, fp1);

// Read image1 info header
BITMAPINFOHEADER bmi1;
fread(&bmi1, sizeof(bmi1), 1, fp1);

// image2
FILE* fp2 = fopen(argv[2], "rb");
if ( !fp2 )
{
    fclose(fp1);
    printf("\n\nOne of the input image file is missing.\n\n");
    printf("Press ENTER to exit...");
    getch();
    return 0;
}

// Read image2 header
BITMAPFILEHEADER bmf2;
fread(&bmf2, sizeof(bmf2), 1, fp2);

// Read image2 info header
BITMAPINFOHEADER bmi2;
fread(&bmi2, sizeof(bmi2), 1, fp2);

if ( bmi1.biHeight != bmi2.biHeight || bmi1.biWidth !=
bmi2.biWidth)
{
    printf("\n\nWidth and Height of given bitmaps are not
matching.\n\n");
    fclose(fp1);
    fclose(fp2);
    return 0;
}

if (bmi1.biBitCount != bmi2.biBitCount)
{
    fclose(fp1);
    fclose(fp2);
    printf("\n\nBitDepth of given bitmaps are not
matching.\n\n");
    printf("\n\nPress ENTER to exit...\n\n");
    getch();
    return 0;
}

int imgSize = bmi1.biHeight * bmi1.biWidth * (bmi1.biBitCount /
8);

// read image1
unsigned char *img1 = (unsigned char*)malloc(imgSize);
if(!img1)
{
    fclose(fp1);
    fclose(fp2);
    printf("\n\nmemory allocation problem\n\n");

```

```

        printf("\n\nPress ENTER to exit...\n\n");
        getch();
        return 0;
    }
    fread( img1, imgSize, 1, fp1);

    // read image2
    unsigned char *img2 = (unsigned char*)malloc(imgSize);
    if(!img2)
    {
        free(img1);
        fclose(fp1);
        fclose(fp2);
        printf("\n\nmemory allocation problem\n\n");
        printf("\n\nPress ENTER to exit...\n\n");
        getch();
        return 0;
    }
    fread( img2, imgSize, 1, fp2);

    // close files
    fclose(fp1);
    fclose(fp2);

    // calculate std dev
    float stdDev = -1;

    if (bm1.biBitCount == 32)
        stdDev = StdDev32Bit(img1, img2, bm1.biWidth,
bm1.biHeight);
    else if (bm1.biBitCount == 24)
        stdDev = StdDev24Bit(img1, img2, bm1.biWidth,
bm1.biHeight);

    free(img1); img1 = NULL;
    free(img2); img2 = NULL;

    printf("\n\nStandard Deviation for %s and %s is: %2f \n\n",
argv[1], argv[2], stdDev);
    printf("\n\nPress ENTER to exit...\n\n");
    getch();
    exit(0);
}

```