//DetecBlackFrame.cpp

#include "stdafx.h"

#include "DetecBlackFrame.h"

#include "MAUtil.h"

#include "iplFrameEx.h"

#include "math.h"

#define BLACK\_TIME 60000000

#ifdef \_OPENMP

#include "omp.h"

#endif

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

#ifdef \_LOGDATA

#include "apputil.h"

#include <time.h>

#include "tchar.h"

#include <mmsystem.h>

#pragma comment(lib,"winmm.lib")

CString g\_szLogPath;

#define \_LOGPATH \_T("\\CQ\_API\\BlackFrame")

void init\_log(void)

{

//g\_szLogPath = CAppUtil::GetMainPath();

g\_szLogPath = CAppUtil::GetUserDataPath();

// g\_szLogPath.Delete(g\_szLogPath.GetLength()-1,1);

// while(g\_szLogPath.Right(1)!=\_T("\\") && g\_szLogPath.GetLength() )

// g\_szLogPath.Delete(g\_szLogPath.GetLength()-1,1);

g\_szLogPath += \_LOGPATH;

CAppUtil::CreateFullDir(g\_szLogPath);

g\_szLogPath += \_T("\\");

CTime time = CTime::GetCurrentTime();

CString szFile;

szFile.Format(\_T("%d\_%d\_%d\_%d\_%d\_%d.txt"), time.GetYear(), time.GetMonth(), time.GetDay(),

time.GetHour(), time.GetMinute(), time.GetSecond());

g\_szLogPath += szFile;

}

void LOGMSG(TCHAR \*szMsg, ... )

{

FILE \*fFile = NULL;

if( (fFile = \_tfopen(g\_szLogPath, \_T("a+") )) == NULL )

return;

va\_list marker;

TCHAR szTmp[2048];

memset(szTmp,0,2048\*sizeof(TCHAR));

va\_start( marker ,szMsg );

\_vstprintf(szTmp,szMsg, marker);

va\_end( marker );

char szMessage[2048];

memset(szMessage,0,2048\*sizeof(char));

#ifdef \_UNICODE

int AnsiStrSize = WideCharToMultiByte(CP\_ACP, 0, szTmp, -1, NULL, 0, NULL, NULL);

WideCharToMultiByte(CP\_ACP, 0, szTmp, -1, szMessage, AnsiStrSize, 0, 0);

#else

strcpy\_s(szMessage, szTmp);

#endif

fseek(fFile,0,SEEK\_END);

fwrite(szMessage,strlen(szMessage),1,fFile);

fwrite("\r\n",strlen("\r\n"),1,fFile);

fclose(fFile);

}

#endif

**BOOL ipl\_RGBToGray(OUT LPBYTE pGrayBuffer, IN const LPBYTE pSrc, int nSrcWidth, int nSrcHeight, int nBytes)**

**{**

**if(nBytes<3) return -1;**

**LPBYTE IpDst= pGrayBuffer;**

**LPBYTE IpSrc= pSrc;**

**register int nByteGap = nBytes-3;**

**if(nByteGap<0) nByteGap=0;**

**int nSize = nSrcHeight\*nSrcWidth;**

**LPBYTE IpDstEnd = IpDst + nSize;**

**BYTE rgbBlue, rgbGreen, rgbRed;**

**while(IpDst<IpDstEnd)**

**{**

**rgbRed = \*(IpSrc++);**

**rgbGreen= \*(IpSrc++);**

**rgbBlue= \*(IpSrc++);**

**\*IpDst++=(BYTE)(0.2125f\*(float)rgbRed + .7154f\*(float)rgbGreen + 0.0721f\*(float)rgbBlue);**

**//\*IpDst++=(\*(IpSrc++) + \*(IpSrc++) + \*(IpSrc++))/3 ;**

**IpSrc+=nByteGap;**

**}**

**return 1;**

**}**

**BOOL ipl\_CpyyGrayToRGB(OUT LPBYTE pDstRGB, IN const LPBYTE pSrcGray, int nSrcWidth, int nSrcHeight, int nBytes)**

**{**

**if(nBytes<3) return -1;**

**LPBYTE IpDst= pDstRGB;**

**LPBYTE IpSrc= pSrcGray;**

**register int nByteGap = nBytes-3;**

**if(nByteGap<0) nByteGap=0;**

**int nSize = nSrcHeight\*nSrcWidth;**

**LPBYTE IpSrcEnd = IpSrc + nSize;**

**while(IpSrc<IpSrcEnd)**

**{**

**\*IpDst++= \*IpSrc;**

**\*IpDst++= \*IpSrc;**

**\*IpDst++= \*IpSrc;**

**IpSrc++;**

**IpDst+=nByteGap;**

**}**

**return 1;**

**}**

CDetectionBlackFrame::CDetectionBlackFrame(void)

:m\_rgbBlack(0,0,0)//,m\_BeforeRGB(255,255,255)

{

m\_nCurFrame =0 ;

m\_nStartFrame =0 ;

m\_nEndFrame =0 ;

m\_lpCallback =NULL;

m\_lpUserData =NULL;

m\_hCVideo =NULL;

memset( &m\_AVResult, 0, sizeof(AVRESULT) );

m\_dwDetecedFlag = CQ\_QC\_FS\_BLACK;

m\_hCommonDLL = NULL;

memset( &m\_VideoApi, 0, sizeof(CVIDEOAPI) );

LoadCommonDLL();

m\_bInitFlag = TRUE;

// m\_nStopProcessFrame= 0 ;

m\_byteNoiseLevel = 0 ;

m\_nNoneDetectStart = 3 ;//%

m\_nNoneDetectEnd = 5 ;//%

m\_dStartTime = 0.0 ;

// m\_dEndTime = 0.0 ;

m\_bDetectFalg = FALSE;

m\_bAlram = FALSE;

m\_nAlramTime = 0 ;

m\_tSystemStart = 0 ;

m\_tSystemEnd = 0 ;

**m\_pGrayBuffer = NULL;**

**m\_nSizeOfGray = 0 ;**

}

CDetectionBlackFrame::~CDetectionBlackFrame(void)

{

if(m\_hCommonDLL)

FreeLibrary(m\_hCommonDLL);

**if( m\_pGrayBuffer )**

**delete[] m\_pGrayBuffer;**

}

HRESULT CDetectionBlackFrame::SetOption( LPSTR\_MAP lpmapOption )

{

STR\_MAP::iterator itor;

CString szValue;

itor = lpmapOption->find(\_T("Noise level"));

if (itor != lpmapOption->end())

{

szValue = (\*itor).second.data();

m\_byteNoiseLevel = (BYTE)\_ttoi( (LPCTSTR)szValue );

// ">>"연산을 쓰기 위한 비트 수 계산.

//-연산은 0-

m\_byteNoiseLevel = (BYTE)( log10( (double)m\_byteNoiseLevel ) / log10(2.0) );

}

itor = lpmapOption->find(\_T("None detect start ratio"));

if (itor != lpmapOption->end())

{

szValue = (\*itor).second.data();

m\_nNoneDetectStart = (UINT)\_ttoi( (LPCTSTR)szValue );

}

itor = lpmapOption->find(\_T("None detect end ratio"));

if (itor != lpmapOption->end())

{

szValue = (\*itor).second.data();

m\_nNoneDetectEnd = (UINT)\_ttoi( (LPCTSTR)szValue );

}

itor = lpmapOption->find(\_T("Alram time"));

if (itor != lpmapOption->end())

{

szValue = (\*itor).second.data();

m\_nAlramTime = (UINT)\_ttoi( (LPCTSTR)szValue );

m\_nAlramTime \*= UINT\_REFERENCE\_TIME;

}

return S\_OK;

}

HRESULT CDetectionBlackFrame::CheckStream(LPAVDATA lpAVData)

{

if( m\_hCommonDLL == NULL ) return E\_FAIL;

if( lpAVData == NULL ) return E\_FAIL;

if( lpAVData->eType != \_AL\_MEDIA\_Video ) return E\_FAIL;

if( m\_bInitFlag )

{

m\_nNoneDetectStart = (UINT)(((float)m\_nNoneDetectStart/100.f) \* lpAVData->video.dwTotalFrames);

m\_nNoneDetectEnd = lpAVData->video.dwTotalFrames - (UINT)( ((float)m\_nNoneDetectEnd/100.f) \* lpAVData->video.dwTotalFrames );

// if( lpAVData->video.dwTotalFrames/lpAVData->video.dbFrameRate > 6200 )//1시간보다 길다면 마지막 10% 프로세싱 중지.

// m\_nStopProcessFrame = lpAVData->video.dwTotalFrames - (UINT)(lpAVData->video.dwTotalFrames\*0.08);

m\_nSizeOfGray = lpAVData->dwDataSize / lpAVData->video.dwBytes;

m\_pGrayBuffer = new BYTE[m\_nSizeOfGray];

m\_bInitFlag = FALSE;

}

// if(lpAVData->video.dwCurrentFrame==0) return S\_OK;

//Histogram을 구하면서 간단한 비트 작업을 같이 하게 함으로서 연산 량을 줄인다.

//비트연산(point pixel 간 xor(^)결과를 CIplFrame의 맴버러 등록 하고 여기서는 그 결과에 대한 분석만으로 콜백함수로 결과 전달.)

//아래는 사용방법에 대한 샘플로서의 가치를 갖는다.

if( lpAVData->video.dwBytes != sizeof(tagRGB) ) return E\_FAIL;

m\_nCurFrame = lpAVData->video.dwCurrentFrame;

m\_hCVideo = lpAVData->video.hCVideo;

//if( m\_nStopProcessFrame && m\_nStopProcessFrame<m\_nCurFrame) return S\_OK;

if(lpAVData->video.dwTotalFrames !=0)

{

if( m\_nNoneDetectStart>m\_nCurFrame ) return S\_OK;

if( m\_nNoneDetectEnd <m\_nCurFrame ) return S\_OK;

}

// CIplFrameEx\* pCurFrame = m\_VideoApi.lpGetFrame( lpAVData->video.hCVideo, m\_nCurFrame );

LPBYTE lpSrc = lpAVData->lpData;

ipl\_RGBToGray( m\_pGrayBuffer, lpSrc, lpAVData->video.dwWidth, lpAVData->video.dwHeight, lpAVData->video.dwBytes);

ipl\_CpyyGrayToRGB( lpSrc, m\_pGrayBuffer, lpAVData->video.dwWidth, lpAVData->video.dwHeight, lpAVData->video.dwBytes);

#ifdef \_LOGDATA

LOGMSG(\_T("%08d %03d %03d %03d"), lpAVData->video.dwCurrentFrame, \*lpR,\*lpG,\*lpB);

#endif

SendErrorFrame( lpAVData, m\_nCurFrame );

return S\_OK;

}

HRESULT CDetectionBlackFrame::SendAlramFrame(LPAVDATA lpAVData, const int& nTargetFrame)

{

//이전 프레임을 얻어 오는 방법.

CIplFrameEx\* pFrame = m\_VideoApi.lpGetFrame( lpAVData->video.hCVideo, nTargetFrame );

#ifdef \_LOGDATA

LOGMSG(\_T("CDetectionBlackFrame::SendErrorFrame() frame=%08d m\_lpCallback=%08x, pFrame=%08x"), lpAVData->video.dwCurrentFrame-1, m\_lpCallback, pFrame);

#endif

if(pFrame==NULL) return S\_FALSE;

m\_nEndFrame = m\_nCurFrame - 1 ;

m\_AVResult.nFrameStart = m\_nStartFrame ;

m\_AVResult.nFrameEnd = m\_nEndFrame ;

m\_AVResult.lpData = pFrame->GetPixelData () ;

m\_AVResult.dwDataSize = pFrame->GetDataSize () ;

m\_AVResult.dwDetectFlag = m\_dwDetecedFlag ;

m\_AVResult.nFrameNumber = pFrame->GetFrameNo () ;

m\_AVResult.rtStartTime = m\_nStartFrame;

m\_AVResult.rtEndTime = m\_nEndFrame;

m\_AVResult.eResultCondition = eCONDITION\_ALARM;

m\_AVResult.tSystemTimeStart = m\_tSystemStart;

time(&m\_AVResult.tSystemTimeEnd);

m\_AVResult.rtStartTime = (REFERENCE\_TIME)( m\_dStartTime\*UINT\_REFERENCE\_TIME );

m\_AVResult.rtEndTime = (REFERENCE\_TIME)(pFrame->GetSampleTime()\*UINT\_REFERENCE\_TIME);

if((m\_AVResult.rtEndTime - m\_AVResult.rtStartTime) > m\_nAlramTime && m\_bAlram)

{

if(m\_lpCallback) m\_lpCallback( &m\_AVResult, m\_lpUserData);

m\_bAlram = FALSE;

}

return S\_OK;

}

HRESULT CDetectionBlackFrame::SendErrorFrame(LPAVDATA lpAVData, const int& nTargetFrame)

{

//이전 프레임을 얻어 오는 방법.

CIplFrameEx\* pFrame = m\_VideoApi.lpGetFrame( lpAVData->video.hCVideo, nTargetFrame );

#ifdef \_LOGDATA

LOGMSG(\_T("CDetectionBlackFrame::SendErrorFrame() frame=%08d m\_lpCallback=%08x, pFrame=%08x"), lpAVData->video.dwCurrentFrame-1, m\_lpCallback, pFrame);

#endif

if(pFrame==NULL) return S\_FALSE;

m\_nEndFrame = m\_nCurFrame ;

m\_AVResult.nFrameStart = m\_nCurFrame ;

m\_AVResult.nFrameEnd = m\_nEndFrame ;

**m\_AVResult.lpData = lpAVData->lpData ;//pFrame->GetPixelData () ;**

**m\_AVResult.dwDataSize = lpAVData->dwDataSize ;//pFrame->GetDataSize () ;**

m\_AVResult.dwDetectFlag = m\_dwDetecedFlag ;

**m\_AVResult.nFrameNumber = lpAVData->video.dwCurrentFrame; //pFrame->GetFrameNo () ;**

m\_AVResult.eResultCondition = eCONDITION\_DETECTION;

m\_AVResult.tSystemTimeStart = m\_tSystemStart;

time(&m\_AVResult.tSystemTimeEnd);

m\_AVResult.rtStartTime = (REFERENCE\_TIME)( lpAVData->video.dbSampleTime \* UINT\_REFERENCE\_TIME );

m\_AVResult.rtEndTime = (REFERENCE\_TIME)( lpAVData->video.dbSampleTime \* UINT\_REFERENCE\_TIME );

if(m\_lpCallback) m\_lpCallback( &m\_AVResult, m\_lpUserData);

// m\_nStartFrame = 0 ;

// m\_nEndFrame = 0 ;

// m\_dStartTime = 0.0 ;

// m\_bDetectFalg = FALSE;

return S\_OK;

}

void CDetectionBlackFrame::SetCallback( LPFN\_AVRESULT lpCallback, LPVOID lpUserData )

{

m\_lpCallback = lpCallback;

m\_lpUserData = lpUserData;

}

void CDetectionBlackFrame::LoadCommonDLL(void)

{

CMAUtil::GetCommonVideoAPI( &m\_hCommonDLL, &m\_VideoApi);

}

HRESULT CDetectionBlackFrame::FinishedStream( void )

{

if( m\_nStartFrame )

{

//이전 프레임을 얻어 오는 방법.

CIplFrameEx\* pFrame = m\_VideoApi.lpGetFrame( m\_hCVideo, m\_nCurFrame );

#ifdef \_LOGDATA

// LOGMSG(\_T("%08d pFrame=%08x"), lpAVData->video.dwCurrentFrame, pFrame);

#endif

if(pFrame==NULL) return S\_FALSE;

m\_nEndFrame = m\_nCurFrame ;

m\_AVResult.nFrameStart = m\_nStartFrame ;

m\_AVResult.nFrameEnd = m\_nEndFrame ;

m\_AVResult.lpData = pFrame->GetPixelData () ;

m\_AVResult.dwDataSize = pFrame->GetDataSize () ;

m\_AVResult.dwDetectFlag = m\_dwDetecedFlag ;

m\_AVResult.nFrameNumber = pFrame->GetFrameNo () ;

//임시로 사용

m\_AVResult.rtStartTime = (REFERENCE\_TIME)( m\_dStartTime\*UINT\_REFERENCE\_TIME );

m\_AVResult.rtEndTime = (REFERENCE\_TIME)( pFrame->GetSampleTime() \* UINT\_REFERENCE\_TIME );

m\_AVResult.eResultCondition = eCONDITION\_DETECTION;

if(m\_lpCallback) m\_lpCallback( &m\_AVResult, m\_lpUserData);

m\_nStartFrame = 0;

m\_nEndFrame = 0;

}

return S\_OK;

}