Wyner-Ziv Video Codec with Unsupervised Motion Learning

David Chen[1], David Varodayan, Markus Flierl, Bernd Girod
Image, Video, and Multimedia Systems Group
Information Systems Laboratory
Stanford Department of Electrical Engineering

Included Software

Source files and a Visual Studios project for the Wyner-Ziv video codec are included in this package. The project has been successfully tested on Windows XP and Windows Vista systems. Any Visual Studios version 2005 or later is compatible. All code is written in ANSI C++.

The only external dependency of this software package is the Matlab library. Any Matlab version 7.0 or later is compatible.

Installation Instructions

The software can be downloaded and installed on any Windows system.

Usage of Chen-Image

This video codec uses the service of Chen-Image, a supplementary image/video processing library. Files for Chen-Image are in the folder *Chen Image*, while the files for the actual video codec are in the folder *LDPC Video DCT – VS 2005*. Necessary dependencies on Chen-Image have already been correctly set in the Visual Studios project.

Linking to the Matlab External Interface

The codec calls Matlab functionality from within the C++ code. The path to the Matlab external interface library needs to be defined inside the Visual Studios project prior to building the project. First, we will locate the external interface library that comes with any standard Matlab installation.

1) Find *matlabroot*, the folder where Matlab is installed. For example:
   C: / Program Files / MATLAB
2) The folder containing the necessary header files will be:
   matlabroot / extern / include
3) The folder containing the necessary library files will be:
   matlabroot / extern / lib / win32 / microsoft

[1] Contact: dmchen@stanford.edu

Having found the location of the Matlab external interface library, we can now update the Visual Studios project.

1) Go into *LDPC Video DCT – VS 2005*. Open the Visual Studios solution file (.sln). There are two projects in this solution: *Chen-Image* and *LDPC Video DCT*.
2) Select project *LDPC Video DCT* in the *Solution Explorer*. Select from the menu bar *Project > Properties*.
3) In the dialog that pops up, select the *Configuration Properties* menu.
4) In the *Configuration* menu at the top, make sure the setting reads *Active (Debug)*.
5) Select the sub-menu *C/C++ > General*. In the line *Additional Include Directories*, we need to input the location of the Matlab header files found above.
6) Select the sub-menu *Linker > General*. In the line *Additional Library Directories*, we need to input the location of the Matlab library files found above.
7) Close *OK* to save the options and close the dialog.
8) Repeat steps 2-7 for the other project *Chen-Image* in the same solution.
9) Select from the menu bar *Build > Rebuild Solution*.

<u>Locating the LDPC Ladder File</u>

A custom-designed LDPC ladder file needs to be used. This ladder file can be downloaded from:

http://msw3.stanford.edu/~dchen/DVC/50688_regDeg3_1on132.zip

Once downloaded and unzipped, the location of the ladder file must be specified in the Visual Studios project. Set *CodingOption::pLogFileName* (see below) as the full path to the ladder file.

<u>Linking to the Intel IPP Library (Optional)</u>

For users who have access to the Intel IPP library, noticeable speed gains can be obtained by utilizing that library in this project. This section can be safely ignored by all other users.

Similar to the processing of linking to the Matlab external interface (see above), we need to specify the path to the Intel IPP files in the Visual Studios project. First, we will find where these files are installed on the computer.

1) Find *intelroot*, the folder where Intel IPP is installed. For example:
   C: / Program Files / Intel IPP
2) The folder containing the necessary header files will be:
   intelroot / include
3) The folder containing the necessary library files will be:
   intelroot / stublib

Having found the location of the Intel IPP library, we can now update the Visual Studios project.

1) Go into *LDPC Video DCT – VS 2005*. Open the Visual Studios solution file (.sln). There are two projects in this solution: *Chen-Image* and *LDPC Video DCT*. Both projects need to be updated.
2) Select project *LDPC Video DCT* in the *Solution Explorer*. Select from the menu bar *Project > Properties*.
3) In the dialog that pops up, select the *Configuration Properties* menu.
4) In the *Configuration* menu at the top, make sure the setting reads *Active (IntelIPP)* rather than *Acitive (Debug)*.
5) Select the sub-menu *C/C++ > General*. In the line *Additional Include Directories*, we need to input the location of the Intel IPP header files found above.
6) Select the sub-menu *Linker > General*. In the line *Additional Library Directories*, we need to input the location of the Intel IPP library files found above.
7) Close *OK* to save the options and close the dialog.
8) Repeat steps 2-7 for the other project *Chen-Image* in the same solution.
9) Select from the menu bar *Build > Rebuild Solution*.

Coding Options

All parameters that control video coding are contained in the *CodingOption* data structure. At the top of *main.cpp*, these options are set.

bool *bSaveDecodedFrame* : saves decoded frames to file if set true
bool *bSaveMotionField* : saves decoded motion fields to file if set true
int *nCodingStructure* : decides the GOP coding structure
    CODING_IPPPPPPP : one key frame for every seven Wyner-Ziv frames
    CODING_IPIPIPIP : one key frame for every Wyner-Ziv frame
    CODING_SINGLE_I : one key frame at the very beginning, then all Wyner-Ziv frames
int *nInterpolateMethod* : decides the motion block-to-pixel interpolation method
    INTERP_NN : nearest-neighbor
    INTERP_LINEAR : bilinear
int *nReconstructMethod* : decides the coefficient reconstruction method
    RECON_CENTROID : use centroid of decoded quantization bin
    RECON_MOTION : use side-information-assisted nearest-neighbor binning
int *nQuantizationArray* : decides the quantization table for the blockwise DCT
    QUANT_JPEG : use JPEG-recommended perceptually weighted table
    QUANT_FLAT : use flat table
int *nQuantizationLevel* : decides quantization granularity [0, 3]
    0 : highest quality
    3: lowest quality
int *nStartFrame* : first frame of video sequence to test
int *nEndFrame* : last frame of video sequence to test
bool *bTestMotionOracle* : runs the motion oracle decoder
bool *bTestMotionLearning* : runs the motion learning decoder
bool *bTestZeroMotion* : runs the no-motion-compensation decoder
bool *bTestIntra* : runs the intra-JPEG decoder

char* *pExperimentFolderName* : full path to the folder where results are stored
char* *pVideoFileName* : full path to the input YUV video file
  only QCIF 4:2:0 format is currently supported
char* *pLadderFileName* : full path to the LDPC ladder file
char* *pLogFileName* : full path to the general-purpose log file
  name is set by default to an ASCII timestamp at the moment of creation (see below)
char* *pLogFileRDName* : full path to the rate-distortion points log file
  name is set by default to an ASCII timestamp at the moment of creation (see below)

Log Files

Two log files are generated for every simulation. One file is the general-purpose log file, which records every message sent to standard output. This file will be useful for reviewing detailed information about the events during the simulation. The other file is the rate-distortion log file, which stores only rate-PSNR values. This file can be directly imported into Matlab or Excel to plot rate-distortion curves.

The name automatically assigned to the general-purpose log file, to avoid naming conflicts, is a modified timestamp reflecting the moment of creation:
  DayOfWeek-Month-Day-Hour-Minute-Second-Year.log

The rate-distortion log file is named similarly, except with *RD* append:
  DayOfWeek-Month-Day-Hour-Minute-Second-Year-RD.log

The columns of the rate-distortion log file will be:
  Frame Number
  Oracle Rate Y, Rate U, Rate V, PSNR Y, PSNR U, PSNR V,
  Learning Rate Y, Rate U, Rate V, PSNR Y, PSNR U, PSNR V
  Zero Motion Rate Y, Rate U, Rate V, PSNR Y, PSNR U, PSNR V
  Intra Rate Y, Rate U, Rate V, PSNR Y, PSNR U, PSNR V

If any mode is switched off (e.g. learning), the rate and PSNR values for that mode are no recorded in the rate-distortion log file.

Output Images

If the flags *bSaveDecodedImage* or *bSaveMotionField* are set true, then output images of important decoded results will be saved. These images are stored in a subfolder of the current experiment folder, identified by the quantization parameter *nQuantizationLevel*:
  pExperimentFolderName / quant-nQuantzationLevel-images

[mode]-motionX.jpg : x-component of blockwise motion field
  subtract 128 to get the true value

[mode]-motionY.jpg : y-component of blockwise motion field
subtract 128 to get the true value
[mode]-motionP.jpg : probability of motion vector reported in *motionX.jpg* and *motionY.jpg*
divide by 255 to get the true value
[mode]-Y.jpg : decoded luminance component
[mode]-U.jpg : decoded chrominance U component
[mode]-V.jpg : decoded chrominance V component