Software description and configuration

Christine Guillemot, Laurent Guillo, Marco Cagnazzo, Giuseppe Valenzise, Béatrice Pesquet-Popescu

Projet PERSEE
SCHÉMAS PERCEPTUELS ET CODAGE VIDÉO 2D ET 3D
ANR-09-BLAN-0170

Livrable **D5.3** *26/09/2013*

Software description and configuration

Christine	GUILLEMOT	IRISA
Laurent	GUILLO	IRISA
Marco	CAGNAZZO	LTCI
Giuseppe	VALENZISE	LTCI
Béatrice	PESQUET-POPESCU	LTCI

Contents

Introduction

This document gathers information about configuration and instructions for use of two software carried out during the Persee project: DCR and WTM.

1 DCR

The *Don't Care Region* (DCR) approach is based on the idea that in multiple-view-plus-depth video, depth maps are not directly viewed, but are only used to provide geometric information for view synthesis at decoder. Thus, as long as the resulting geometric error does not lead to unacceptable quality for the synthesized view, each depth pixel only needs to be reconstructed at the decoder coarsely within a tolerable range. We first formalize the notion of tolerable range per depth pixel asDonâĂŹt Care Region (DCR), by studying the synthesized viewdistortion sensitivity to the pixel value – a sensitive depth pixel will have a narrow DCR, and *vice versa*.

1.1 Overview

We now define per-pixel DCRs for depth map $\mathbf{D}_n$, assuming target synthesized view is n. In the following, we will rather refer to the *disparity* field $\mathbf{d}_n$, which can be obtained from the depth once the camera parameters are known. A pixel $v_n(i,j)$ in texture map $\mathbf{v}_n$, with associated disparity value $d_n(i,j)$, can be mapped to a corresponding pixel in view $n+1$ through a view synthesis function $s(i,j;d_n(i,j))$. In the simplest case where the views are captured by purely horizontally shifted cameras, $s(i,j;d_n(i,j))$ corresponds to a pixel in texture map $\mathbf{v}_{n+1}$ of view $n+1$ displaced in the x-direction by an amount proportional to $d_n(i,j)$. The view synthesis error, $\varepsilon(i,j;d)$, can thus be defined as the absolute error between reconstructed and original pixel value, given disparity d for pixel (i,j); i.e., $\varepsilon(i,j;d) = |s(i,j;d) - v_n(i,j)|$. If $\mathbf{d}_n$ is compressed, the reconstructed disparity value $\tilde{d}_n(i,j)$ employed for view synthesis may differ from $d_n(i,j)$ by an amount $e(i,j) = \tilde{d}_n(i,j) - d_n(i,j)$, resulting in a (generally larger) view synthesis error $\varepsilon(i,j;d_n(i,j)+e(i,j)) > \varepsilon(i,j;d_n(i,j))$. We define the *Don't Care Region* $\mathrm{DCR}(i,j) = [\mathrm{DCR}_{low}(i,j), \mathrm{DCR}_{up}(i,j)]$ as the *largest* contiguous interval of disparity values containing the ground-truth disparity $d_n(i,j)$, such that the view synthesis error for any point of the interval is smaller than $\varepsilon(i,j;d_n(i,j)) + \tau$, for a given threshold $\tau > 0$. Note that DCR intervals are defined *per pixel*, thus giving precise information about how much error can be tolerated in the disparity maps.

The DCR information can be then used in order to perform a more effective motion estimation, to encode the prediction residual, and to enhance the use of the SKIP mode. We have implemented in Matlab a function that can generate the DCR's of a MVD sequence, and save them into a binary file that can in turn be used by a modified H.264/MPEG-4 AVC encoder (JM reference software v. 18.0)

1.2 Matlab functions

```
[DCR_low DCR_up] = generate_DCR(isLeft, thres, depth_scale)
```

Inputs

`isLeft`: binary flag indicating whether the DCT is to be computed for the left or for the rigth view. E.g., if you have views 3 and 5 in "Kendo", and you have to synthesize view 4, you shall use isLeft = 1 to generate the DCR

needed to encode disparity of view 3, and isLeft = 0 to generate the DCR relative to disparity of view 5. Note that DCR is defined as the worst-case errror when you have texture and depth from one same view (e.g. texture and depth from view 3) and you want to generate the other view (view 5)

`thres`: it is the τ in the report (and in our PCS paper): the highest tolerated threshold to define the DCR. We advice to use $\tau = 5$ or similar values

`depth_scale`: it is the scaling factor from depth to disparity. It must be known a priori. It depends on the sequence, that should in any case be rectified. For Kendo, the value to use is 0.204

Outputs

`DCR_low` and `DCR_up`, respectively the lower and upper bound of DCR interval per pixel. In order to be used by the encoder, they must be converted into a binary file by using write_DCR_bin.m

Note that this function generates a DCR for a single image. So you need to call it image-by-image for an entire sequence. The inputs of the scripts are the texture and the depth of the right and left view, that should be passed as png files with names (left_depth.png, left_texture.png, etc.)

`write_DCR_bin.m`

This script writes a matrix ($M \times N \times F$, with $M \times N$ is the spatial resolution and F the number of frames) into a file that can be used by the encoder. The DCR must be stored into variables named `DCR_low` and `DCR_up`; they will be written into two files named `DCR_low.bin` and `DCR_up.bin` Please note that `generate_DCR` only generates one 2D matrix. It is up to the user to generated as many matrices as you need and to stack them into a 3D matrix.

1.3 Modified encoder `lencode.exe`

This is the modified version of H.264 (JM v.18.0) using the files `DCR_low.bin` and `DCR_up.bin` produced by the `write_DCR_bin.m` function in order to perform the DCR-based encoding as described in the delivrable D4.3 and in the PCS paper. It works in baseline mode, with some restrictions, as one can see from the `encoder_baseline.cfg` file. In particular, the RDO should be "high complexity" and the motion estimation should be "full search".

2 WTM

WTM is an intra prediction method based on a linear combination of template matching predictors. The method was previously described in [1]. After a quick reminder, the following sections presents how some details peculiar to this method were implemented and how to configure WTM. An example of use is then given.

2.1 Quick overview

WTM aims at providing an intra prediction for blocks of 4x4, 8x8, 16x16 and 32x32 sizes. This prediction is based on a linear combination of template matching predictors belonging to the causal neighbourhood.

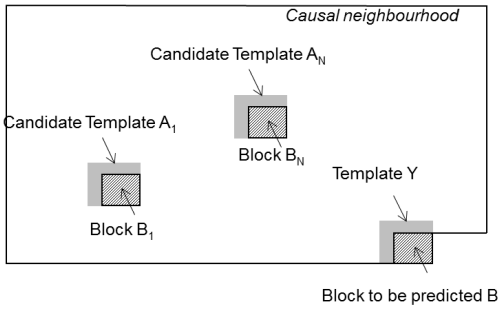

Figure 1: Search regions from causal neighbourhood.

Then, N blocks B_i surrounded by the best matching areas are used to compute predictors P_i, which are then averaged to get the prediction P of the block B:

$$P = \frac{1}{N} \sum_{i=1}^{N} P_i \tag{1}$$

WTM relies on this general approach but there are three main enhancements:

- it uses 4 different template shapes whatever the block size: the traditional L-shape which is 1 pixel large and three other shapes with the left, the top part of both can be 4 pixel large. However, only one template shape is used to determine all template predictors.
- the correlation factors is based on the dot product between the template and the template predictors.
- template predictors are not searched within all the causal neighbourhood but within only two or three search windows. The number of search windows is

related to the rank of the block to be predicted within the prediction unit (PU) and their size depends on the size of the block to be predicted.

For more details about these three characteristics see [1].The following sections gives information about how they have been used and implemented.

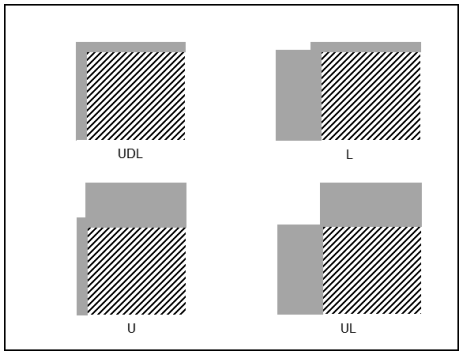

Figure 2: Shape of templates.

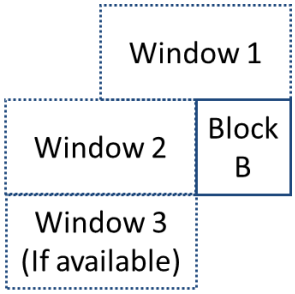

Figure 3: Search windows positions relatively to block B.

2.2 Some implementation details

Distinctive features listed in the previous sections lead to the following choices of implementation.

2.2.1 WTM activation

WTM is not always activated for all PU sizes. Is activation depends on the class of videos belonging to the corpus provided by JCT-VC and the PU sizes.

The cases for which WTM is activated are listed in the Table 1

Table 1: Activation of WTM according to video classes and PU sizes

	HE					LC				
	4x4	8x8	16x16	32x32	64x64	4x4	8x8	16x16	32x32	64x64
Class A	-	X	X	X	-	-	X	X	-	-
Class B	X	X	X	X	-	X	X	X	-	-
Class C	X	X	X	-	-	X	X	-	-	-
Class D	X	X	X	-	-	X	X	-	-	-
Class E	X	X	X	X	-	X	X	X	-	-
Class F	X	X	X	X	-	X	X	X	-	-

Table 2: Relation between intra mode and shape of template

INTRA mode	Shape
10	UDL
11	U
12	L
13	UL

2.2.2 Template shape signalling

The 4 template shapes are available. Consequently, two pieces of information must be signalled to the decoder: when a block is predicted with WTM and which shape of template was used. To do so, four direction modes have been overloaded: from the mode 10 up to the mode 13. They are associated to a shape (cf. Fig.1) as listed in Table 2. An extra bit is added for all of these four modes and set to true if WTM is used as described in Fig. 4.

```
prediction_unit( x0, y0, log2PUWidth, log2PUHeight, PartIdx ,
                 InferredMergeFlag ) {
  if( skip_flag[ x0 ][ y0 ] ) {
    if(NumMPMCand  >  1 )
      merge_idx[ x0 ][ y0 ]
  } else if( PredMode  ==  MODE_INTRA ) {
    prev_intra_luma_pred_flag[ x0 ][ y0 ]
    if( prev_intra_luma_pred_flag[ x0 ][ y0 ] )
      if( NumMPMCand > 1 )
        mpm_idx[ x0 ][ y0 ]
    else
      rem_intra_luma_pred_mode[ x0 ][ y0 ]
    if(PredMode is associated to WTM)
      WTM_mode_flag[x0][y0]
  } else { /* MODE_INTER */
    ...
  }
```

Figure 4: Signaling taking into account WTM.

Table 3: Search areas characteristics

Block B size	Search windows number	Search windows width	Search windows height
4x4	3	12	4
8x8	2	20	8
16x16	2	8	16
32x32	2	4	32

2.2.3 Number and dimensions of search windows

The number of search windows and also their size depond on the size of the block to be predicted.

The characteristics of the search areas are summarized in Table 3.

2.3 Configuration

The WTM algorithm is written on top of the test model of HEVC, release 4.0. So, the configuration files dedicated to WTM are based on the HTM-4.0 all intra encoding configuration file.

A section is added to specify parameters relation to WTM. In particular, this sections indicates whether:

- WTM is activated or not
- 4x4, 8x8, 16x16, 32x32, WTM prediction are activated

An optional parameter, STMObserver, can be set to generate statistics or prediction maps.

The following excerpt of a configuration file lists the parameters related to STM.

```
#============ WTM ================
STM : 1 # 0 : unsed, 1: activated

STM4x4          : 1 # Prediction activated for 4x4 block size
STM8x8          : 1 # Prediction activated for 8x8 block size
STM16x16        : 1 # Prediction activated for 16x32 block size
STM32x32        : 1 # Prediction activated for 32x32 block size

STMObserver : 3 # 0:unused, 1: % of selection, 2: stats files, 3: output frames
```

The other sections of the configuration file are kept unchanged.

2.4 Example of use

To build the sofware, refer to the "how-to" provided in the deliverable 3.4 and uploaded to the website of the Persee project (http://persee.irccyn.ec-nantes.fr/prive/).

Once built, the encoding is launched with the following command:

```
TAppEncoder -c tests.cfg
```

where "tests.cfg" is the configuration file.
To decode the encoded video, just enter the following command:

```
TAppDecoder -b str.bin -o dec.yuv
```

where "str.bin" is the encoded video (the name was specified in the encoding configuration file) and "dec.yuv" the name of the decoded video.

References

[1] Persee, "2d coding tools final report," ANR-09-BLAN-0170, Delivrable D 3.4, July 2013.