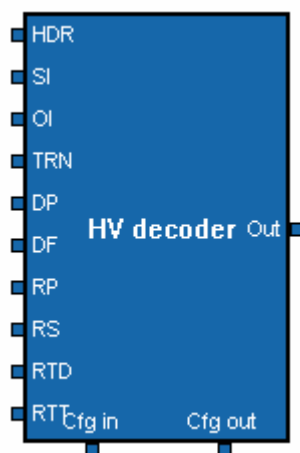


# HV decoder

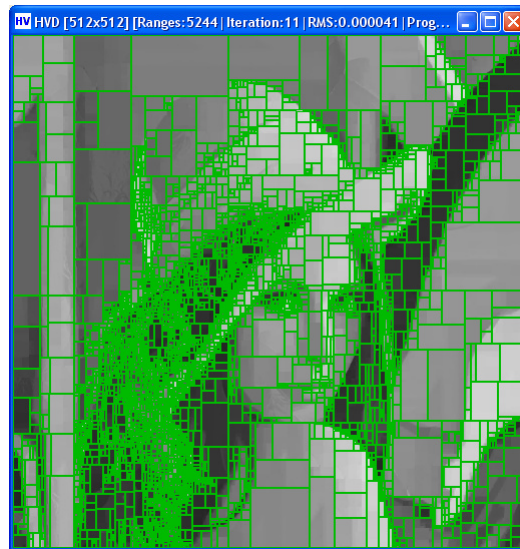
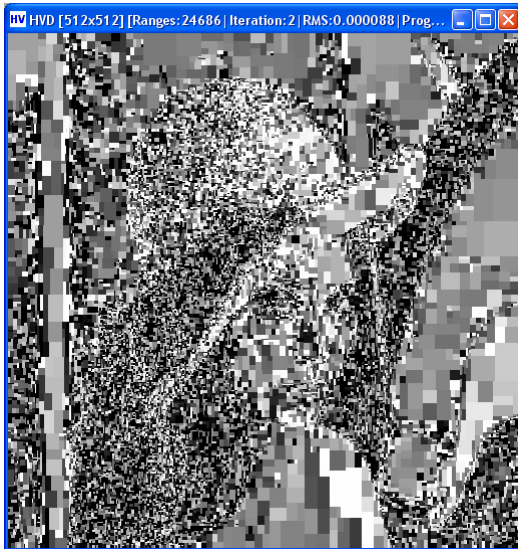
## Obsah

1. Popis.....	1
2. Závislosti .....	2
3. Implementačné informácie .....	2
4. Piny .....	2
5. Konfigurácia .....	3
5.1 Konfiguračná štruktúra.....	3
5.2 Konfiguračný dialóg.....	4
6. Definícia triedy .....	5
7. Príklad použitia .....	7

## 1. Popis



Trieda *HV decoder* je implementáciou iteračného dekodéra pre IFS systém popísaný triedou *HV encoder*. Vstupom je IFS systém zakódovaný vo výstupných štruktúrach triedy *HV encoder* a výstupom je šedo tónový obraz blízky atraktoru vstupného IFS. Dekódovací proces prebieha v iteráciách, kedy sa aplikujú jednotlivé transformácie na predošlú iteráciu. Ako počiatočný vstup do dekodéra je použitý čierny obraz (obsahujúci len nuly). Iteračným prístupom nie je možné dosiahnuť atraktor, teda výstup je len priblíženie sa k tomuto atraktoru. Vzďialenosť výstupu a atraktora môžeme kontrolovať nastavením parametrov dekodéra. Zapnutím voľby vizualizačného okna môžeme sledovať priebeh dekompresie v okne.



V okne sa zobrazuje postupnosť iteračných krokov. V záhlaví okna sa zobrazujú štatistické informácie ako rozlíšenie výstupu, počet blokov (*Ranges*), počet vykonaných iterácií (*Iteration*), RMS odchýlka od atraktora (*RMS*) a percentuálna kompletnosť operácie (*Progress*).

## 2. Závislosti

Trieda *HV decoder* používa tieto externé definičné súbory a knižnice:

**Headers:** *filtergraph.h*, *datatypes.h*

**Libs:** *filtergraph.lib*

## 3. Implementačné informácie

Informácie o triede:

**Názov:** *HV decoder*

**Verzia:** *1.0*

**Magic:** *140*

Informácie o definíciách:

**Header:** *hvdecoder.h*

**Konfiguračný header:** *hvdecodercfg.h*

**Lib:** *hvdecoder.lib*

Informácie o knižnici obsahujúcej triedu:

**Názov knižnice:** *HV decoder Library*

**Verzia knižnice:** *1.0*

**Dll súbor knižnice:** *hvdecoder.dll*

## 4. Piny

Piny sú popísané spôsobom: „*[index pinu na filtri] názov pinu: popis pinu*“.

**[0] Cfg in:** Konfiguračný vstup. Vstupom je serializovaná konfigurácia.

- [1] **Cfg out:** Konfiguračný výstup. Výstupom je serializovaná konfigurácia.
- [2] **HDR:** Vstup *Header*. Vstupom je buffer obsahujúci hlavičkovú štruktúru, v ktorej sú dáta potrebné na korektné dekodovanie.
- [3] **SI:** Vstup *Scale Indices*. Vstupom je pole celých čísel v štruktúre *TIntArray*, ktoré obsahuje kvantizačné indexy škálovacích faktorov domén.
- [4] **OI:** Vstup *Offset Indices*. Vstupom je pole celých čísel v štruktúre *TIntArray*, ktoré obsahuje kvantizačné indexy posúvacích faktorov domén.
- [5] **TRN:** Vstup *Transformations*. Vstupom je pole celých čísel v štruktúre *TIntArray*, ktoré obsahuje zakódované modifikačné časti transformácií.
- [6] **DP:** Vstup *Domain Positions*. Vstupom je pole celých čísel v štruktúre *TIntArray*, ktoré obsahuje pozície domén.
- [7] **DF:** Vstup *Domain Factors*. Vstupom je pole celých čísel v štruktúre *TIntArray*, ktoré obsahuje škálovacie faktor blokov.
- [8] **RP:** Vstup *Range Positions*. Vstupom je pole celých čísel v štruktúre *TIntArray*, ktoré obsahuje pozície blokov.
- [9] **RS:** Vstup *Range Sizes*. Vstupom je pole celých čísel v štruktúre *TIntArray*, ktoré obsahuje veľkosti blokov.
- [10] **RTD:** Vstup *Range Tree Distanties*. Vstupom je pole celých čísel v štruktúre *TIntArray*, ktoré obsahuje deliace vzdialenosti v strome blokov.
- [11] **RTT:** Vstup *Range Tree Types*. Vstupom je pole celých čísel v štruktúre *TIntArray*, ktoré obsahuje typy delení v strome blokov.
- [12] **Out:** Výstup. Výstupom je šedo tónový obraz v štruktúre *TPlane*, ktorý predstavuje výstup iteráčného dekodéra.

Na korektné načítanie IFS systému nie sú potrebné všetky vstupy. Potrebné sú *HDR*, *SI*, *OI*, *TRN*, *DP*, *DF* a jedna z dvojíc (*RP*, *RS*), (*RTD*, *RTT*). Ak sú prítomné vstupy *RP*, *RS*, tak informácie o blokov sú dekodované z tejto dvojice. Ak nie, tak sa pristupuje k vstupom *RTD*, *RTT*. Ak ani táto dvojica nie je prítomná, dekodovanie sa ukončí s chybou.

## 5. Konfigurácia

### 5.1 Konfiguračná štruktúra

Konfiguračná štruktúra je definovaná nasledovne:

```
typedef struct {
    int      DecompressionMode;    // mód dekompresie
    int      XVal;                 // x-ový škálovací faktor
    int      YVal;                 // y-ový škálovací faktor
    int      Iterations;           // maximálny počet iterácií
    float     MaxError;            // maximálna povolená RMS chyba
    float     Quality;             // kvalita
    BOOL      PostProcessing;       // dodatočné vyhladenie
    BOOL      FeedbackWindow;      // vizualizačné okno
    BOOL      FeedbackWindow_Ranges; // kreslenie blokov vo vizualizačnom okne
} THVDecoderConfig;

// dekompresné módy
DM_MAXITERATIONS    0           // maximálny povolený počet iterácií
DM_MAXERROR         1           // maximálna povolená RMS odchylka
```

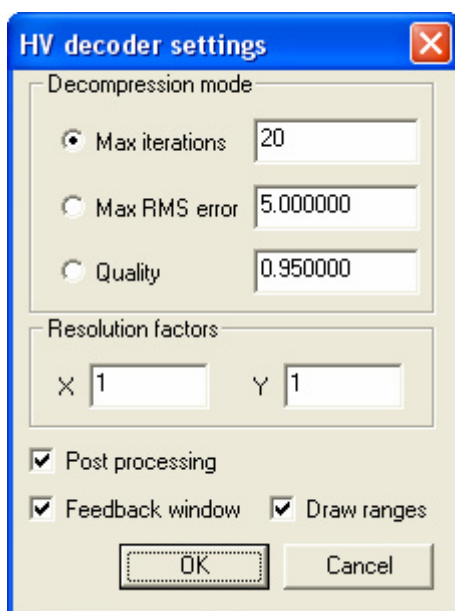
Mód dekompresie určuje za akých podmienok sa má dekódovací proces skončiť. Mód *DM\_MAXITERATIONS* predstavuje mód s maximálnym povoleným počtom iterácií. Potom hodnota *Iterations* predstavuje maximálny povolený počet iterácií. Po dosiahnutí tohto počtu sa proces zastaví. Mód *DM\_MAXERROR* použije hodnotu *MaxError* ako maximálnu povolenú odchýlku od atraktora (odchýlku od atraktora môžeme z hora odhadnúť z RMS odchýlky dvoch po sebe idúcich iterácií). Proces končí pri dosiahnutí danej blízkosti k atraktoru. Mód *DM\_QUALITY* použije hodnotu *Quality* ako maximálnu povolenú RMS odchýlku od atraktora v závislosti od dynamického rozsahu kódovanej predlohy načítaného z hlavičkovej štruktúry. Hodnota *Quality=1.0* znamená najvyššiu kvalitu (žiadna odchýlka od atraktora), hodnota *Quality=0.0* znamená najnižšiu kvalitu.

Výstupné rozlíšenie možno škálovať celočíselnými faktormi v smere x faktorom *XVal* a v smere y faktorom *YVal*.

Voľba *PostProcessing* povolí dodatočné vyhladenie iteračného výstupu. Vyhladenie sa aplikuje na okolie hrán blokov, kvôli zmierneniu nežiaducich blokovitých artefaktov, ktoré vznikajú pri kódovaní v nižších kvalitách.

Voľba *FeedbackWindow* zapne vizualizačné okno, v ktorom môžeme sledovať priebeh dekódovacieho procesu a vidieť jednotlivé iterácie. Zapnutá vizualizácia môže výrazne spomaliť dekódovací proces. Voľba *FeedbackWindow\_Ranges* povolí zobrazovanie stromu blokov vstupného IFS vo vizualizačnom okne.

## 5.2 Konfiguračný dialóg



Konfiguračný dialóg poskytuje voľby popísané v konfiguračnej štruktúre. V bloku *Decompression mode* si zvolíme mód dekompresie. Voľba *Max iterations* zapne mód *DM\_MAXITERATIONS*, a nastaví hodnotu *Iterations* na danú hodnotu. Voľba *Max RMS error* zvolí mód *DM\_MAXERROR* a nastaví hodnotu *MaxError* na danú hodnotu. Voľba *Quality* zvolí mód *DM\_QUALITY* a nastaví hodnotu *Quality* na danú hodnotu.

V bloku *Resolution factors* môžeme meniť škálovacie faktory výstupného rozlíšenia. Hodnoty *X* a *Y* nastaví hodnoty *XVal* a *YVal* na dané hodnoty.

Voľba *Post processing* zapne voľbu *PostProcessing*, voľba *Feedback window* zapne voľbu *FeedbackWindow* a voľba *DrawRanges* zapne voľbu *FeedbackWindow\_Ranges*.

## 6. Definícia triedy

Trieda *HV decoder* je definovaná nasledovne:

```
class THVDecoderFilter : public TFilter {
public:
    HINSTANCE hInstance;           // identifikácia inštancie knižnice

    TPin* CfgInPin;                 // vstupný konfiguračný pin
    TPin* CfgOutPin;                // výstupný konfiguračný pin
    TPin* HDRPin;                   // vstupné piny...
    TPin* SIPin;
    TPin* OIPin;
    TPin* TRNPin;
    TPin* DPPin;
    TPin* DFPin;
    TPin* RPPin;
    TPin* RSPin;
    TPin* RTDPin;
    TPin* RTTPin;
    TPin* OutPin;                  // výstupný pin

    THeader *Header;               // vstupné dáta prečítané zo vstupných pinov
    TIntArray *ScaleIndexes;
    TIntArray *OffsetIndexes;
    TIntArray *Transformations;
    TIntArray *DomainPositions;
    TIntArray *DomainFactors;
    TIntArray *RangePositions;
    TIntArray *RangeSizes;
    TIntArray *RangeTreeDistanties;
    TIntArray *RangeTreeTypes;

    int Width;                     // šírka výstupného obrazu
    int Height;                    // výška výstupného obrazu
    CPlane *Plane1;                // prvá pomocná rovina
    CPlane *Plane2;                // druhá pomocná rovina
    CPlane *ActivePlane;           // ukazovateľ na aktívnu rovinu
    TPlane OutPlane;               // výstupná rovina

    CQuantizer *sQ;                // kvantizér pre škálovacie faktory domén
    CQuantizer *oQ;                // kvantizér pre posúvacie faktory domén

    CPointerArray *RangePool1;     // prvý zoznam blokov
    CPointerArray *RangePool2;     // druhý zoznam blokov
```

```

CPointerArray *ActiveRangePool;    // aktívny zoznam blokov

CFeedbackWindow *FW;              // inštancia vizualizačného okna

BOOL StopFlag;                    // indikátor zastavenia
DET_FLOAT progress;               // kompletnosť operácie
double RMSError;                  // RMS odchylka od atraktora

int iteration;                    // číslo iterácie

int DecompressionMode;            // nastavenia...
int XVal;
int YVal;
int Iterations;
float MaxError;
float Quality;
BOOL PostProcessing;
BOOL FeedbackWindow;
BOOL FeedbackWindow_Ranges;

DET_FLOAT smax;                  // konštanty...
DET_FLOAT smin;
DET_FLOAT omax;
DET_FLOAT omin;
DET_FLOAT inmax;
DET_FLOAT inmin;
DET_FLOAT emax;

int getInputData();              // pomocné funkcie...
int buildRangePool();
int calcConstants();
int createQuantizers();
int createPlanes();
int freeTempResources();

int buildRange(CRange *range, CPlane *srcPlane, CPlane *destPlane);
int render();
int makePartition(CRange *range, int dist, int type);
int makeRange(CRange *range);
int hFlip(CPlane *plane);
int vFlip(CPlane *plane);
DET_FLOAT getMaxScale();
DET_FLOAT rms(CPlane *plane1, CPlane *plane2);
int postprocess();
int postprocessRange(CRange *range, CPlane *OldPlane, CPlane *NewPlane);
int makeOutputData();

int setConfigFromPin(TPin *pin);  // nastavovanie konfigurácie na pinoch...
int putConfigOnPin(TPin *pin);
int freeConfigOnPin(TPin *pin);

```

```

    int clearOutputPins();           // čistenie výstupných pinov...
    int freeOutputData();
public:
    THVDecoderFilter();             // koštruktor
    THVDecoderFilter(HINSTANCE hInst); // koštruktor s parametrami
    ~THVDecoderFilter();            // deštruktor

    int setConfigData(TBuffer config, int type); // predefinované funkcie triedy TFilter...
    TBuffer getConfigData(int type);
    int initialize();
    int run();
    int reset();
    int stop();
    int finalize();
    int showConfigDialog();
};

```

## 7. Príklad použitia

