Codificação de Áudio e Vídeo

Relatório
Trabalho Prático nº 2

Filipe Manco | 45500
filipe.manco@ua.pt

Frederico Honório | 42914
fredericohonorio@ua.pt

Universidade de Aveiro
Departamento de Electrónica Telecomunicações e Informática

Índice

Arquitectura do software

Do nosso ponto de vista a estrutura implementada no primeiro trabalho não foi bem conseguida. Como tal, para este segundo trabalho, a estrutura do software foi completamente redesenhada. Uma grande parte das funcionalidades apresentadas na primeira parte do trabalho (estruturas de dados, leitura e escrita de ficheiros, etc) foram portadas para esta versão, no entanto, por se mostrar desnecessário, algum código não foi portado (por exemplo a aplicação de efeitos).

A arquitectura proposta nesta iteração permite executar tarefas numa pipeline, usando para a comunicação entre fases a estrutura de dados YuvFrame. De seguida é mostrado um diagrama que pretende ilustrar possíveis utilizações do software.

A classe YuvFrame é, como o nome indica, a representação de uma frame YUV. Esta estrutura permite realizar de forma automática todas as operações básicas sobre uma frame deste tipo, nomeadamente conversões entre formatos (YUV444, YUV422 e YUV420), extração e substituição de blocos e píxeis ou comparação de frames.

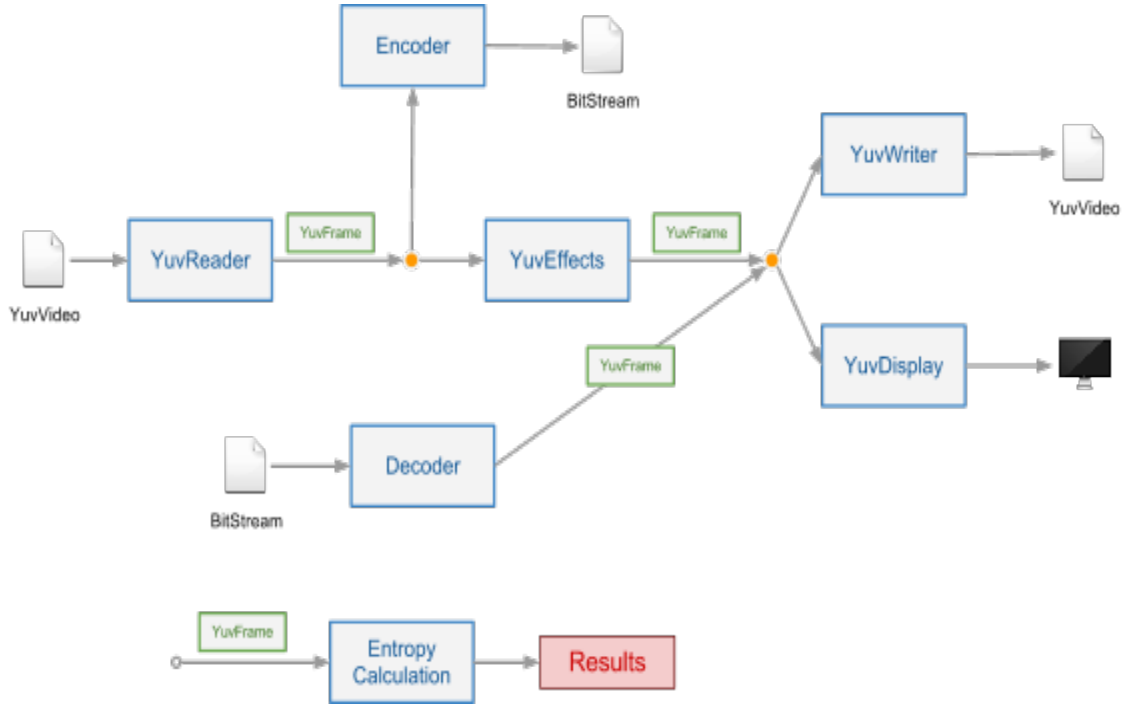

Figura 1: Arquitectura da nova solução

Relativamente às classes responsáveis pela manipulação da frame, os nomes são auto-explicativos, indicando a sua funcionalidade. A classe Block é a representação de um bloco de píxeis de uma frame YUV, sendo utilizada como estrutura de dados auxiliar para a extracção e substituição de blocos de uma YuvFrame.

Para a codificação e descodificação de vídeos foram criadas várias classes auxiliares. Nomeadamente a classe BitStream para a leitura de escrita de ficheiros bit-a-bit, a classe Golomb, que permite a codificação usando códigos de Golomb e a classe Predictor que fornece preditores lineares e não lineares como os usados nas normas JPEG e JPEG-LS.

Codificação de Entropia

A codificação de Golomb é mais indicada quando os valores a codificar são pequenos, isto torna-a indicada para codificar as diferenças em valores de pixel, que será o caso em que a utilizaremos mais.

A codificação de Golomb simples só funciona para números inteiros positivos. A solução para codificar números negativos é transformar os números positivos em pares positivos, e os negativos em ímpares positivos da seguinte maneira:

$$v' = \begin{cases} v \times 2, & v >= 0 \\ |v| \times 2 - 1, & v < 0 \end{cases}$$

Onde v é o valor original, e v' o valor que será efectivamente codificado.
Implementamos a codificação (e descodificação) de Golomb na classe Golomb.

Predição Espacial Intra-Frame

A codificação intra-frame utiliza a informação anterior na mesma frame. Em geral são usados os pixeis numa vizinhança do pixel que queremos codificar. O tipo de predição define que pixeis utilizamos para fazer a predição. A classe Predictor implementa as funções de predição com os sete tipos do JPEG e a predição não-linear do JPEG-LS.

A codificação de uma frame é então o calculo da predição para cada pixel, e a escrita da diferença entre o valor previsto e o valor efectivo. Isto é claramente mais eficiente em termos de espaço, em vez de guardar o valor absoluto de cada pixel, guardamos apenas a diferença entre pixeis, codificando-a com a classe Golomb. A descodificação é apenas o oposto. Os valores de diferença são lidos e somados aos valores do preditor. Todo este processo é feito pela classe IntraCoder.

A quantização é também implementada nesta classe. A implementação é a simples divisão inteira dos valores de diferença por um factor de quantização. Isto reduz efectivamente a qualidade da imagem pois introduz erro num processo que não tinha perdas, mas também reduza o valor da diferença, resultado numa quantidade de informação menor. Durante o processo de codificação, e quando a quantização é utilizada, é necessário reescrever os pixeis da frame actual com o valor quantizado, porque a utilização de predição implica que a informação acedida pelo preditor têm de ser igual tanto na codificação como na descodificação.

Codificação Intra-Frame usando preditores

A codificação intra-frame de um vídeo é a simples codificação intra-frame de cada uma das frames do vídeo, assegurando que o ficheiro resultante tem também um cabeçalho com a informação da dimensão, tipo de chroma subsampling, e a taxa de frames por segundo. A classe IntraFrame também faz a leitura e escrita deste cabeçalho.

Codificação com compensação de movimento por blocos

Este é um tipo de codificação inter-frame, o que significa que utiliza informação de uma ou mais frames anteriores para codificar a frame actual. Neste caso específico o algoritmo utiliza apenas a frame imediatamente anterior à actual.

A codificação por blocos é um tipo de codificação que analisa cada frame em blocos de píxeis. Neste caso concreto o tamanho dos blocos é configurável. Um pormenor interessante é o caso em que o vídeo utiliza “chroma subsampling”. Neste caso, e na nossa implementação em concreto, são considerados para as componentes U e V blocos de dados contíguos ignorando o facto de os píxeis correspondentes não serem contíguos.

A codificação utilizando compensação de movimento por blocos é realizada da seguinte forma: para cada bloco da frame actual é procurado na frame anterior um bloco semelhante nas imediações do bloco actual; para o melhor bloco encontrado (isto é, o bloco com menor erro médio) são codificadas apenas as diferenças; o deslocamento para o bloco anterior é também codificado e incluído no bitstream.

Codificação Híbrida

A codificação híbrida é feita pela classe HybCoder e consiste em codificar uma keyframe com codificação intra-frame e um número configurável de frames subsequentes com codificação inter-frame. Isto é útil em situações em que há poucas alterações entre frames ou estas são deslocações de segmentos da imagem, mas há também mudanças em que a codificação inter-frame produz diferenças maiores. O facto de o período entre keyframes ser fixo, no entanto, implica que as mudanças de cena, por exemplo, teriam de ocorrer exactamente quando a keyframe é escolhida, para reduzir o número de keyframes ao mínimo necessário.

Para tentar reduzir esse problema adicionamos a opção de tentar estimar a melhor altura para introduzir uma keyframe. Esta opção passa por comparar a frame actual com a frame anterior, verificando se o valor médio da diferença entre os pixeis passa um determinado threshold. Se passar, é feita uma codificação intra-frame, senão é inter-frame. Este método não é o melhor

contudo. Imaginando uma frame que é constituída por blocos da frame anterior em posições aleatórias, a sua diferença excederá o threshold, mas a codificação inter-frame mostraria-se vantajosa. A melhor opção seria fazer ambas as codificações e comparar os tamanhos produzidos, ou a qualidade em caso de codificação lossy. A implementação deste método seria mais complexa e a execução mais demorada.

A codificação híbrida de um video continua a necessitar da escrita e leitura do cabeçalho do ficheiro. As frames são depois escritas no ficheiro, e caso seja usada a escolha de keyframe automática é escrito um bit que indica o tipo de codificação da frame antes da mesma.

Codificação com TDC

A transformada discreta de cosenos permite expressar uma função (neste caso uma sucessão de valores), como uma soma de cosenos. Os coeficientes gerados têm informação suficiente para reconstruir, possivelmente com perdas, os valores introduzidos. Estes coeficientes podem também ser quantizados. A nossa implementação, a classe DCTCoder, realiza esta transformação em blocos de 8x8 pixeis. A escolha da dimensão do bloco deve-se ao facto da norma JPEG definir as matrizes de quantização para luminância e crominância nestas dimensões, matrizes que também usamos. A quantização é feita multiplicando estas matrizes por um factor. Cada frame é dividida em blocos, e os coeficientes resultantes da transformada de cada bloco é guardada.

Resultados

Para cada um dos codificadores desenvolvidos foram também desenvolvidos scripts para a execução de testes. Esses scripts foram disponibilizados em conjunto com o código da solução estando disponíveis na pasta resultados.

Alguns dos resultados dos testes, assim como algumas conclusões que foram obtidas da análise dos resultados e durante a execução do trabalho, são apresentados de seguida.

Codificação intra-frame usando preditores

A codificação intra-frame usando preditores mostra-se útil quando existem padrões previsíveis nas frames. Para os preditores implementados, esta codificação mostra-se útil quando existem muitas zonas de cor semelhante nas frames. Esta é uma codificação bastante boa em termos temporais.

Dos resultados obtidos verificámos que, para o pior caso, correspondente a um vídeo de 1280x720 píxeis, o tempo de codificação por frame foi de aproximadamente 500ms.

Relativamente às taxas de compressão para compressão lossless, foram testados um preditor linear e um preditor não linear. Para o primeiro caso conseguiu-se em média uma taxa de compressão de 20% e para o segundo uma taxa de 24%.

Um caso interessante é o do vídeo “mobile_calendar-176x144-30-422.yuv”. Para este vídeo as taxas de compressão lossless foram péssimas. No caso do preditor não linear o stream codificado é 2% maior que o vídeo original, e no caso do preditor não linear a compressão foi de 0%. Este facto deve-se à natureza do vídeo. Pelo que podemos verificar na figura 2, mostrada abaixo, o vídeo tem uma zona bastante grande com grandes alterações de cor de pixel para pixel, como tal a previsão falha muitas vezes e o erro codificado é muito grande, aumentando muito o tamanho do vídeo codificado.

Figura 2: Frame do vídeo “mobile_calendar-176x144-30-422.yuv”

Os resultados para a codificação são mostrados na tabela 1 abaixo. Os testes foram efectuados para 3 conjuntos de quantizações diferentes.

Quantização	Taxa de compressão média	PSNR
Y: 2, U: 4, V: 4	38%	Y: 51,37 dB U: 43,77 dB, V: 43,72
Y: 4, U: 8, V: 8	45%	Y: 43,48 dB, U: 37,13 dB, V: 38,05
Y: 8, U: 16, V: 16	48%	Y: 36,74 dB, U: 30,98 dB, V: 32,96

Tabela 1: Resultados de codificação lossy para codificadores intra-frame usando preditores

Codificação inter-frame com compensação de movimento

Este tipo de codificação é bastante útil para situações em que a há movimento da cena ao longo do tempo sem grandes alterações desta.

Situações limite em que apenas há um movimento da câmara sobre uma cena parada são o melhor caso para este tipo de codificação, pois para cada bloco (exceptuando os blocos dos limites da frame) há sempre um bloco na frame anterior igual, o que permite que a grande maioria da frame actual seja codificada com erros de zero. Situações de mudança de cena são o pior caso pois é difícil encontrar correspondências na frame anterior para blocos da frame actual, resultando na codificação de erros muito grandes.

Codificador híbrido

O codificador híbrido é, relativamente ao codificador anterior, tipicamente bastante mais eficiente no que diz respeito a taxas de compressão. Isto à custa de um tempo de codificação bastante mais elevado.

Para este codificador foram testados os efeitos da alteração de cada um dos seguintes parâmetros: tamanho do bloco, área de pesquisa e período da key frame. Os resultados são apresentados na tabela 2. De notar que estes resultados foram obtidos para codificações lossless, fazendo variar apenas cada um dos parâmetros indicados.

Parâmetro	**Taxa de compressão média**	**Tempo de codificação (ms/ frame)**
Tamanho do Bloco		
5	32,60%	1 283
10	33,86%	989
15	33,29%	931
Área de pesquisa		
5	33,03%	360
10	33,86%	1 041
15	34,11%	1 909
Período da keyframe		
Dinâmico	34,76%	1 000
10	32,66%	986
20	33,27%	979
50	33,62%	997
100	33,75%	1 043

Tabela 2: Resultados de codificação lossless para o codificador híbrido

Relativamente ao tamanho do bloco verificámos que o melhor parâmetro é o intermédio. Apesar de a amostra ser relativamente pequena é fácil perceber que para blocos mais pequenos haverá mais blocos por frame e consequentemente mais overhead para a

codificação das deslocações, para blocos maiores será mais complicado encontrar blocos semelhantes e a codificação não será tão boa. Desta forma, o valor médio dependerá de um tradeoff entre os dois aspectos anteriores. Relativamente aos tempos de codificação estes foram relativamente constantes.

No que diz respeito à área de pesquisa facilmente se percebe que o aumento deste parâmetro permite melhores codificações à custa de um maior tempo de codificação. Isto devido ao facto de, aumentado a área de pesquisa, haver maior probabilidade de encontrar um bloco semelhante ao actual, no entanto, uma pesquisa maior demora obviamente mais tempo. Também se pode facilmente pensar que este crescimento não irá ocorrer para áreas de pesquisa demasiado grandes, estabilizando a certo ponto. Isto porque, em condições normais, de movimento na cena, a existência de blocos semelhantes dá-se nas imediações do bloco. Excluindo situações especiais por exemplo havendo padrões na cena. Por fim à também que considerar que a codificação de grandes deslocamentos é dispendiosa. Este aspecto pode ser melhorado escolhendo o parâmetro M para o codificador de Golomb a partir da área de pesquisa, optimizando assim o número de bits despendidos neste aspecto.

Por fim no que diz respeito ao período de introdução de uma key frame, verificamos que o método mais vantajoso é o método dinâmico, mantendo um tempo de execução muito semelhante aos restantes, mesmo existindo o overhead introduzido pelos cálculos necessários à decisão de introdução da keyframe. Esta enorme vantagem deve-se ao facto de este método introduzir keyframes apenas quando necessário. Introduzir poucas keyframes irá causar uma grande acumulação de erro aumentado o tamanho da codificação inter-frame. Introduzir muitas keyframes irá causar um aumento do tamanho pois as keyframes ocupam mais espaço (por serem codificadas em modo intra-frame) do que as restantes.

Neste caso a decisão para a introdução de uma nova frame baseia-se no erro médio entre a frame actual e a anterior. Existem outras opções que iriam melhor certos aspectos do vídeo. Uma delas é a codificação de todas as frames em modo intra e em modo inter e a posterior escolha da codificação que produziu um menor resultado. Esta opção iría melhorar a taxa de compressão. Para o caso de codificação lossy, podería-se considerar uma análise do erro introduzido a cada frame, e introduzir uma keyframe sempre que esse erro ultrapassar um determinado limiar. Esta opção iria melhorar o SNR do vídeo codificado.

Analisando os resultados para cada um dos vídeos testados, verificamos que o vídeo “football-176x144-30-422.yuv” é consistentemente pior utilizando este método. Este é um vídeo com muito movimento muito rápido. Já o vídeo “galleon-176x144-30-422.yuv” é consistentemente melhor, sendo este um vídeo com muito pouco movimento. Isto atesta os pontos fortes desta codificação já apresentados acima.

De notar que não foram efectuados testes para valores maiores de tamanho de bloco e área de pesquisa porque o tempo de codificação se revelou demasiado elevado para ser acabado em tempo útil.

Codificação com TDC

Para este algoritmo de codificação intra-frame foram efectuados testes para diferentes valores do parâmetro de quantização. Os resultados médios no que diz respeito à taxa de compressão tempo de codificação e PSNR são apresentados na tabela 3 de seguida.

Quantização	Taxa de compressão média	Tempo de codificação médio (ms/frame)	PSNR médio (dB)
0	3%	1 302	57,23
100	48%	1 243	34,82
250	50%	1 240	32,40

Tabela 3: Resultados para o codificador intra-frame usando TDC

Verificamos relativamente ao método de codificação intra-frame usando preditores, que este algoritmo é bastante mais dispendioso temporalmente, no entanto, oferece também melhores resultados.

Relativamente ao parâmetro de quantização, como seria de esperar, uma maior quantização fornece uma maior taxa de compressão à custa de uma menor qualidade do vídeo. Os tempos de codificação são relativamente constantes, pois uma quantização diferente não implica diferenças no algoritmo e implica sempre os mesmo cálculos.

Verificou-se que para a quantização zero, em que as perdas apenas são devidas a arredondamentos, nalguns casos houve até algum aumento no tamanho do vídeo.

Outros resultados

Para evitar estender muito o relatório foram apenas indicados os resultados principais. Anexada ao relatório existe uma folha de cálculo com a grande maioria dos resultados obtidos em bruto, dos quais foi extraída a informação aqui apresentada.