Trabalho Prático 1 - CAV

Relatório

Grupo de trabalho:

Filipe Manco - 45500
Frederico Honório - 42914

Índice

Estrutura de dados

Toda a realização deste guião foi baseada numa classe, denominada `YUV`, que representa uma frame de vídeo. Esta é uma estrutura de dados básica que implementa apenas as funcionalidades mais primitivas. Todas as restantes funcionalidades requeridas no guião são implementadas por classes adicionais que se baseiam nesta classe.
A classe YUV providencia as seguintes funcionalidades:

- Leitura e escrita de ficheiros no formato YUV fornecido nas aulas
- Display de frames no ecrã
- Conversão de cores para preto e branco
- Inversão de cores
- Aumento/redução da luminância
- Subamostragem temporal

Implementação

A classe YUV é constituída basicamente por um buffer interno, que armazena a informação da frame actual, e um conjunto de atributos que permitem efectuar operações sobre a informação do buffer. Para além da informação necessária para a manipulação do buffer existem também variáveis relativas à manipulação do ficheiro e estruturas de dados do openCV para o display da imagem.

Manipulação do buffer

O buffer possui sempre a informação da frame mais recente. Sempre que uma operação é efectuada sobre o objecto, a operação é efectuada sobre os dados presentes no buffer. Este buffer é basicamente um espaço de memória que armazena informação de cada pixel da imagem. De referir que a imagem é armazenada no formato YUV em modo planar, isto é, os três planos da imagem (Y, U e V) são armazenados de forma separada, ao invés do modo packed em que as componentes de cada pixel são dispostas de forma consecutiva. O buffer pode ser alterado através de métodos de manipulação de imagem, ou através de métodos de leitura do ficheiro.
A classe YUV é capaz de ler e escrever ficheiros em qualquer dos formatos YUV444, YUV422 ou YUV420, no entanto, qualquer operação é realizada sobre os dados no formato YUV444. Para acomodar esta funcionalidade, o buffer lógico referido desdobra-se em dois buffers reais, um buffer para os dados no formato original e outro para os dados em YUV444. A utilização destes buffers pretende-se eficiente a nível espacial, como tal, foram efectuadas as seguintes optimizações:

- A componente Y está apenas presente num buffer
- O segundo buffer só é alocado quando o formato original não é YUV444

Desta forma a estrutura só ocupa o espaço mínimo essencial para armazenar todos os dados.

Leitura e escrita de ficheiros

Para permitir a manipulação de ficheiros a classe YUV pode ser instanciada de três formas distintas:

- Associada a um ficheiro para leitura
- Associada a um ficheiro para escrita
- Não associada a um ficheiro

Estes três modos de instanciação permitem utilizar a mesma classe para fins distintos. De notar que depois de instanciada, o modo da classe não pode ser alterado, isto é, um objecto instanciado para leitura não pode ser usado para escrita e vice-versa.

Display de frames

O display de frames é realizado recorrendo ao openCV. O openCV aceita dados em formato RGB no modo packed. A conversão para RGB é sempre efectuada a partir do buffer em formato YUV444, de acordo com as seguintes equações, aplicadas a cada pixel:

```
b = 1.164 * (y - 16) + 2.018 * (u - 128);
g = 1.164 * (y - 16) - 0.813 * (u - 128) - 0.391 * (v - 128);
r = 1.164 * (y - 16) + 1.596 * (v - 128);
```

YuvEffects

A aplicação YuvEffects não foi implementada por falta de tempo. No entanto, todos os métodos necessários para a realização desta aplicação estão implementados na classe YUV. Desta forma, a implementação desta aplicação resume-se a parsing de argumentos da linha de comandos e chamadas a métodos da classe YUV.

Conversão para Preto e Branco

A conversão de uma frame para preto e branco corresponde a manter apenas a componente Y e atribuir às componentes U e V a constante 127 em todos os pixeis.

Inversão de cores

A inversão de cores é efectuada através das seguintes equações:

```
y = 255 - y;
u = 255 - u;
v = 255 - v;
```

Alteração da luminância

A alteração da luminância da frame é alcançada por uma mudança da componente Y. A equação utilizada é a seguinte:

```
y = (1.0 + factor) * y;
```

Nesta equação factor é uma percentagem que indica em quanto deve ser alterada a luminância, podendo ser um valor positivo (aumento de luminância) ou negativo (decréscimo de luminância).

Subsampling temporal

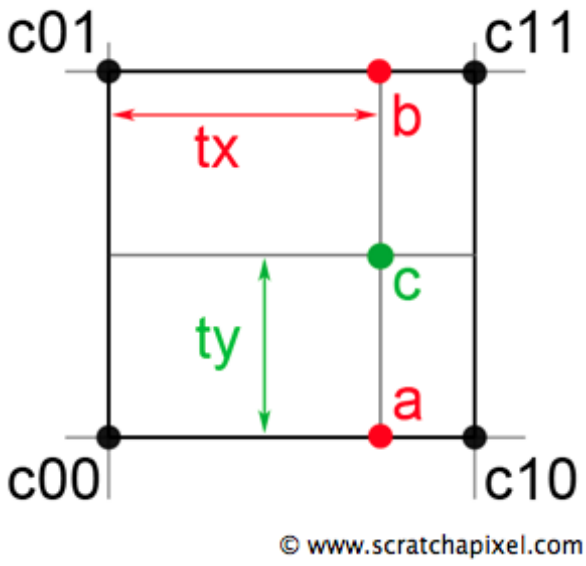
Para implementar o subsampling temporal, alteramos o método de ler as frames, e caso esta função seja utilizada, discartamos `factor`-1 frames dependendo do factor de subsampling utilizado. A framerate é também ajustada, sendo dividida por `factor`.

YuvBlock

Na classe YUV, disponibilizamos métodos para copiar e preencher blocos de uma frame, mas a aplicação para testar a copia de blocos não está completa.

YuvResize

A classe YuvResize permite redimensionar a frame actual de um objecto YUV. Tem duas operações, expansão e redução, e cada uma destas tem um modo simples e um melhorado. Fornece também um método de criar uma copia vazia do objecto redimensionado.
A expansão simples é feita criando na imagem de destino um quadrado de pixeis com as dimensões factor x factor, onde factor é o valor indicado. Depois é percorrida a grelha de pixeis e replicando o valor do pixel actual para o quadrado correspondente.
No caso melhorado utilizamos interpolação bilinear, e neste caso são tidos em conta os valores adjacentes ao pixel que queremos expandir para calcular os valores intermédios. A seguinte imagem ilustra o processo.



Os pontos c01, c11, c00, c10 são os valores da imagem original e o ponto `c` é o valor que queremos estimar. Fazendo uma interpolação linear de c01 e c11 obtemos `b`, c00 e c10 dá-nos `a`, e fazendo a interpolação linear de `a` e `b` obtemos `c`. Na nossa implementação

apenas aplicamos este método à componente Y, visto que a performance em tempo real seria sacrificada, a implementação para as restantes componentes é, no entanto, virtualmente igual.
Para a redução, há também uma relação entre um quadrado de pixeis na imagem original e um único pixel na imagem reduzida, no modo simples escolhemos o primeiro pixel deste quadrado.
No modo melhorado, criamos a média dos pixeis no quadrado original, atribuímos esse valor ao pixel.

Entropy and Peak Signal to Noise Ratio

These functions where implemented by accessing the pixel values for each frame and computing the values for each component, in the end, these values are averaged.

Cálculo de bitrates

Bitrates de transmissão

O cálculo dos bitrates necessários para a transmissão do vídeo em tempo real é efectuado através das equações mostradas abaixo.
Seja R a bitrate necessária em kbps, nRows o número de linhas do vídeo, nCols o número de colunas e fps a frame rate do vídeo.

YUV444:
```
R = 3 * nRows * nCols * fps * 8 / 1000
```

YUV422:
```
R = [(nRows * nCols) + (2 * (nRows / 2) * nCols)] * fps * 8 / 1000
```

YUV420:
```
R = [(nRows * nCols) + (2 * (nRows / 2) * (nCols / 2))] * fps * 8 / 1000
```

Os cálculos são exemplificados abaixo para algumas sequências. Para as restantes sequências os cálculos são semelhantes.

carphone-176x144-30.yuv:
```
R = [(176 * 144) + (2 * 88 * 72)] * 30 * 8 / 1000 = 9123,84 kpbs
```

galleon-176x144-30-422.yuv:
```
R = [(176 * 144) + (2 * 88 * 72)] * 30 * 8 / 1000 = 12165,12 kpbs
```

old_town_cross-1280x720-50-444.yuv:
```
R = 3 * 1280 * 720 * 50 * 8 / 1000 = 1105920 kpbs
```

Taxas de compressão

O cálculo da taxa de compressão necessário para transmitir o vídeo em tempo real num canal de 128kbps é efectuada recorrendo à seguinte equação (sendo R a taxa de transmissão necessária para o vídeo original):

```
C = R / 128
```

Cálculos exemplificativos são mostrados abaixo.

carphone-176x144-30.yuv:

```
C = 9123,84 / 128 = 71,28
```

galleon-176x144-30-422.yuv:

```
C = 12165,12 / 128 = 95,04
```

old_town_cross-1280x720-50-444.yuv:

```
C = 1105920 / 128 = 8640
```

Referências

http://www.scratchapixel.com/lessons/3d-advanced-lessons/interpolation/bilinear-interpolation/