Yves Benabou Consulting
EPITA
Novembre - Décembre 2021

La Vidéo Numérique en Pratique:
Décompression et Affichage de Flux Vidéo

Objectifs:

- Décoder des flux vidéo
- Interpréter les formats de sortie
- Traiter les images natives en vue de leur affichage
- Désentrelacer les images entrelacées
- Afficher en cadence les images

Prérequis:

- Une machine sur laquelle vous pouvez:
  - compiler du C
  - utiliser tout autre langage et environnement favoris (Python...)
  - être administrateur pour pouvoir installer ce qui est nécessaire.
- Installez **`ffmpeg`**, **`vlc`** et/ou **`vlc-nox`** (**`cvlc`**)
- Installez un visualiseur d'images capable de présenter des **`pgm`** et des **`ppm`** (**`geeqie`** par exemple)

Indications:

- Les fichiers vidéo fournis sont dans le dossier **`videos`**
- Les outils de décompression fournis sont dans le dossier **`tools`**
- Les usages recommandés sont dans les fichiers **`README.xxx`**
  avec **`xxx`** = le nom de l'outil.

A. Jouer un flux MPEG-2 élémentaire de test:

1. Dans le dossier **`videos/elementary`**, avec **`vlc`**, visualisez les séquences MPEG-2 suivantes :

   ```
   vlc -V <renderer> bw_numbers.m2v / pendulum.m2v / Jaggies1.m2v
   # avec <renderer> = x11 ou xvideo, selon ce qui fonctionne sur votre machine
   ```

   Ce sont les flux de test conformes à la spécification MPEG-2 que vous avez vus en cours. Ils ont pour but d'aider à implémenter un décodage et un affichage adéquats.

   Attention: `bw_numbers.m2v` est très pédagogique mais clignote fortement quand on le désentrelace en bob!

   Choisissez l'une d'entre elles pour vos expérimentations ci-dessous.

2. Avec **`mpeg2dec`**, convertissez en pile d'images votre séquence MPEG-2 choisie (cf. aide de **`mpeg2dec`**).
   Il est recommandé de sortir les images au format **`pgm`** pour pouvoir les traiter ultérieurement.

3. Observez les **pgm** générées. Comment sont-elles structurées?
   Quel est le format de l'image: résolution, profondeur, sampling mode?

4. Modifiez **mpeg2dec** pour logger simplement les flags
   progressive_frame, top_field_first, repeat_first_field
   de chaque image décodée.

5. Avec votre propre code et dans le langage de votre choix, implémentez un convertisseur d'images vers un format plus humainement lisible (**ppm RGB** est assez universel).
   Cela vous servira à:
   - Comprendre et implémenter une conversion YUV → RGB
   - Faire une application que vous allez progressivement enrichir.

6. Implémentez une option pour que votre programme rende soit en **ppm**, soit directement à l'écran (API libre).

7. Dans le cas d'un rendu à l'écran, rajoutez une option permettant de forcer la cadence du flux élémentaire via la ligne de commande (en images par seconde).

8. Modifiez **mpeg2dec** pour que depuis le MPEG-2 Sequence Header, vous remontiez également la valeur **frame_period,** sachant que :
   frame_period = 27M / cadence_ips

9. Faites en sorte que votre programme gère cette **frame_period** si elle est disponible, sinon se rabatte sur une valeur par défaut de 25 ips.
   La cadence que vous forcez éventuellement en ligne de commande (question A.7) est prioritaire.

10. Implémentez un désentrelaceur bob.
    Pour vous aider, comparez vos résultats avec **vlc** en mode bob :
    vlc -V <renderer> --vout-filter=deinterlace
    --deinterlace-mode bob <file.m2v>

Rappels :

- A partir d'une frame entrelacée (paire de fields), le désentrelaceur doit fournir deux frames désentrelacées.

- Durant le désentrelacement, faites attention au réalignement de la chroma et de la luma en fonction du sampling mode et de la polarité top/bottom du field en cours.

- Le désentrelaceur doit se conformer à la spécification MPEG-2, c'est à dire ne désentrelacer que les images marquées comme entrelacées et dans le bon ordre (flags de la question A.4)
  ,
- Conseil: gardez la possibilité de forcer ces flags via la ligne de commande de votre application, cela vous sera utile par la suite.

B. Jouer un flux vidéo de chaîne d'infos américaine assez notoire:

1. Dans le dossier **videos/ts,** avec **ffplay**, jouez le fichier **cnn.ts**
   Quel est le PID du flux vidéo?

2. **mpeg2dec** sait démultiplexer un PID de TS (option **-t <pid>**)
   pour tenter de le décoder comme un flux vidéo élémentaire.
   Connaissant le PID vidéo de **cnn.ts**, convertissez-le en pile de **pgm**
   via **mpeg2dec.**

3. Avec votre application, désentrelacez les **pgm** et jouez-les en cadence.
   Qu'observez-vous?

4. Visualisez les flags **progressive_frame, top_field_first**
   provenant de **mpeg2dec**.
   A votre avis, que se passe-t-il ?

   Pour vous aider, comparez votre résultat visuel avec **vlc**
   en mode bob :
   vlc -V <renderer> --vout-filter=deinterlace
   --deinterlace-mode bob cnn.ts

5. Dans **mpeg2dec**, trouvez et loggez le flag **progressive_sequence**.
   Si il est **== 1** , ce flag dit que dans une séquence MPEG-2, toutes les images sont progressives.
   Que constatez-vous?
   Quelle (triste) erreur de compréhension a effectué l'encodeur?

6. Quelle option heuristique pouvez-vous implémenter pour tenter de jouer convenablement ce fichier?

C. Jouer un flux vidéo de chaînes de divertissement asiatiques:

On s'intéresse au flux **`videos/ts/ctv.ts`**, qui contient plusieurs programmes.

1. Avec **`ffplay`**, trouvez les numéros de ces programmes.
   Relevez le troisième PID vidéo.

2. Démultiplexez-le avec **`mpeg2dec`** et jouez-le via votre application.

3. Sans tenir compte du jeu d'acteurs, est-ce que le gâteau est vraiment appétissant? Autrement dit, quel problème observe t'on uniquement pendant les effets spéciaux?

   Pour vous aider, comparez avec **`vlc`** comme précédemment.

   Selon-vous, que s'est-il passé au montage dans cette séquence précise?

4. Avec **`ffplay`**, trouvez le premier PID vidéo.

5. Idem question C.2.

6. En vous efforçant davantage à ignorer le jeu d'acteurs, quelle particularité rencontrez-vous avec la signalisation des images de ce flux?

D. Vers un meilleur désentrelaceur:

1. Faites un désentrelaceur adaptatif spatial simple, c'est à dire qui:

   i. Découpe chaque field en petites zones,

   ii. Estime un seuil « immobile / en mouvement » pour chaque zone entre deux fields de même polarité (rendez ce seuil configurable),

   iii. Désentrelace en bob les zones en mouvement du field courant,

   iv. Sinon, mélange en weave les zones immobiles du field courant avec le field précédent.

2. Notamment avec **`ctv.ts`**, où observez-vous le plus d'améliorations dans l'image désentrelacée? Pourquoi ?

3. Pour vous aider, comparez votre résultat visuel avec **`vlc`** en mode de désentrelaement « X » (littéralement) :
   `vlc -V <renderer. --program=??? --vout-filter=deinterlace --deinterlace-mode X ctv.ts`