1873653 – DALPHOND, Jean-Olivier
1928777 – SARWAT, Mariam

INF8770 - RAPPORT DE TRAVAIL PRATIQUE #3

Travail présenté à M. Hugues Perreault

École Polytechnique de Montréal
11 décembre 2020

Question 1

On voit que les images en JPEG sont beaucoup compressées, et des effets de blocs apparaissent très clairement. À l'inverse, les images PNG sont très belles et les transitions de couleurs se font très naturellement. Cela aura un impact sur l'efficacité de reconnaissance des algorithmes. On sait que le format MP4 est déjà compressé, même si certains macroblocs sont codés sans perte, mais la qualité reste très bonne.

Pour l'idée d'Alice, nous considérons que les caractéristiques de performances seront comme suit :

Temps de traitement	L'extraction des images est nécessaire, et le temps de traitement sera donc plus long avec cette approche comme il faut traiter chaque trame.
Précision de la reconnaissance	Pour obtenir un bon pourcentage, on devra utiliser les images PNG, qui sont nettement plus fidèles à l'original. Les effets de bloc présents dans les JPEG pourraient fausser les données. Toutefois, avec les PNG, on devrait avoir d'excellents résultats.
Écart moyen	Étant donné qu'on peut trouver la trame avec précision, on devrait se retrouver avec un écart nul ou quasi nul.
Mémoire de stockage utilisée	Cette méthode devrait utiliser moins de mémoire que l'algorithme de Bob. On a besoin de garder le vidéo en mémoire, mais traiter une image à la fois.

Pour l'algorithme proposé par Bob, voici les caractéristiques de performance que nous avons identifiées :

Temps de traitement	Cette méthode devrait être plus rapide que celle d'Alice, car on prend pour acquis que les images sont déjà extraites. Toutefois, elle devrait être plus lente que celle de Carol, car on doit quand même traiter chaque trame.
Précision de la reconnaissance	On a le choix entre PNG et JPEG, puisqu'on extrait et stocke les images de chaque trame et on a accès aux deux versions lors de la comparaison. Dans les deux cas, on devrait obtenir un bon taux de précision. L'image compressée par JPEG est toujours la même, si l'image source est identique.
Écart moyen	L'écart devrait être nul ou quasi nul, comme pour l'algorithme d'Alice, car on doit traiter toutes les trames.
Mémoire de stockage utilisée	La compression JPEG est intéressante ici, car on économise beaucoup de mémoire, mais dans les deux cas, on utilise quand même beaucoup de mémoire. C'est l'algorithme le plus gourmand à ce niveau, parmi les trois proposés.

Finalement, voici nos impressions par rapport à l'algorithme proposé par Carol :

Temps de traitement	On a théoriquement 29,97 fois (certains vidéos ont d'autres taux, mais une valeur assez répandue) moins d'images à traiter ici, donc cette proposition devrait être la plus rapide, de loin.
Précision de la reconnaissance	Cet algorithme est le moins précis, puisqu'on considère la similitude (histogramme et « la plus faible » distance euclidienne) et non l'exactitude (distance euclidienne nulle).
Écart moyen	L'écart sera d'une seconde au maximum, ce qui peut être acceptable dans certains contextes, mais peut manquer de précision selon les besoins. Cela peut générer des faux négatifs si on a une scène très courte et que l'échantillonnage passe par-dessus.
Mémoire de stockage utilisée	Cet algorithme est le moins gourmand en termes de mémoire, car on a beaucoup moins d'images à traiter.

Dans le cas de Bob, on peut utiliser JPEG ou PNG comme mentionnée. Puisqu'on a accès aux images de comparaison dans les deux formats, et que l'on peut effectuer la conversion lors de l'analyse des trames, il serait plus efficace d'utiliser le format JPEG en termes de mémoire utilisée. La conversion vers JPEG demande toutefois plus de puissance de calcul, alors si c'est cette facette qui prévaut dans les considérations, il pourrait s'avérer préférable de choisir PNG.

Question 2

Exécution du code :

```
python3 generate_video_hists.py
…
python3 find_image.py
…
```

Nous avons inclus avec le rapport le code et les résultats obtenus avec les valeurs de BIN_COUNT à [8, 8, 8]. Avec l'implémentation actuelle, on retourne un résultat même si la valeur de la distance euclidienne n'est pas nécessairement nulle. Cela est problématique si l'on ne considère que cet algorithme, mais nous savons qu'habituellement, l'utilisation des histogrammes constitue une première étape pour défricher la plus grande partie des résultats. Nous pouvons déterminer un seuil à partir duquel les résultats sont considérés comme faux. Par la suite, il existe d'autres techniques pour trouver avec plus de précision une correspondance (filtrage des gradations et fondus enchaînés, comparaison des arêtes, etc.).

N.B. : Nous avons fait quelques essais pour paralléliser *find_image.py*, et nous avons laissé *find_image_threaded.py* dans la remise, mais nous avons abandonné l'implémentation après quelques tentatives. Le résultat qu'on obtenait était que le traitement semblait effectivement parallélisé, mais la sortie des résultats dans le tableur Excel ne fonctionnait pas.

PNG :

Cela étant dit, on constate que l'utilisation des PNG donne d'assez bons résultats avec cet algorithme. On retrouve même des images avec une distance euclidienne nulle, ce qui implique que l'image étudiée est exactement celle prise dans le vidéo, au minutage précis. La plus grande distance euclidienne normalisée trouvée est de 1,169657, qui n'est d'ailleurs pas une image tirée des vidéos (une image en « out »). Toutes les images reconnues qui ne sont pas des faux positifs proviennent du bon vidéo, et leur distance euclidienne normalisée maximale est de 0,397407. Par-dessus cette valeur, toutes les images reconnues sont des faux positifs. C'est donc une performance parfaite (100%) si l'on ne considère pas les faux positifs. En considérant les faux positifs toutefois, on tombe à 77,5%.

Les données recueillies pour procéder à l'analyse totalisent 3520973 octets, ce qui est très peu. Le temps d'indexage est environ 29,2 secondes (semble être *multithreaded*) et le temps de recherche dans les vidéos totalise et 17,34641 secondes (un seul *thread* selon notre compréhension), ce qui est également très peu. En moyenne, c'est 0,08673205 seconde par image PNG. D'autres équipes ont des temps de traitement pour les algorithmes d'Alice ou de Bob qui dépassent les 3 heures. Seulement 7 images identifiées correctement présentent un delta sur le minutage de plus d'une seconde. Le delta moyen de minutage est de 0,518252 seconde, ce qui est très raisonnable somme toute.

JPEG :

Pour les résultats en JPEG, les résultats nous ont surpris. On a aussi réussi à identifier correctement tous les vidéos qui se trouvaient réellement dans les vidéos. Comme dans le cas des PNG, par la nature de l'algorithme, les images qui ne proviennent pas réellement d'un vidéo sont reconnues, donc on a une grande quantité de faux positifs : 45 au total. Considérant la qualité très douteuse des JPEG, c'est un résultat intéressant. Les pourcentages sont les mêmes que pour PNG. Toutefois, les choses se gâtent considérablement lorsqu'on prend en considération le delta de minutage. 54 analyses présentent un delta supérieur à une seconde, ce qui trahit une très grande imprécision. Si l'on considère seulement les vidéos reconnues correctement, on obtient un delta moyen de 1,582913 secondes. Cela n'est pas vraiment satisfaisant, car on excède en moyenne le seuil d'une seconde qu'on a configuré pour la récupération des trames de référence. Le temps total de recherche des images JPEG est de 15,62236 secondes, soit légèrement moins que pour les PNG. En moyenne, on a besoin de 0,0781118 seconde par image.

En somme, on constate que le temps de traitement est très rapide comme on l'avait prévu. Un temps qui se calcule en minutes est excellent dans le contexte. La précision est un facteur qui nous a surpris, car nous nous attendions à avoir un taux inacceptable même pour les PNG, mais nous avons obtenu un résultat quand même excellent! Toutefois, pour les JPEG, c'est nettement insuffisant et imprécis. On le constate d'ailleurs par l'écart moyen : avec les PNG on se trouve en bas d'une seconde en moyenne, mais avec les JPEG c'est bien supérieur. On peut présumer

qu'avec d'autres algorithmes appliqués subséquemment, on pourrait identifier avec une assez grande précision les vidéos. Finalement, en termes de mémoire de stockage, c'est un algorithme très efficace, car pour l'entièreté des vidéos, nous avons utilisé 3,4 Mo. Rappelons que cette quantité de données est en lien avec la prise d'images de référence chaque seconde. Tout de même, c'est très peu!

Question 3

Exécution du code :

```
python3 generate.py
…
python3 find.py
…
```

Nous avons visé l'utilisation des PNG pour la conception de notre algorithme, mais nous avons quand même fait l'essai en JPEG par curiosité. En somme, nous avons utilisé l'algorithme de Carol de la question 2, et l'avons amélioré de façon à corriger ses défauts. Bien sûr, nous avons inclus les fichiers dans la remise.

Dans la nouvelle version du fichier *hist.py*, nous avons ajouté une méthode qui calcule la distance Tchebychev plutôt que la distance euclidienne. Ce choix a été fait suite à la consultation d'un article qui fait la comparaison de différents algorithmes et qui montre la distance normalisée calculée. Le cas de figure qui a retenu notre attention est celui de la figure 3, où l'on voit que l'image *doge_school.png* présente une distance euclidienne supérieure à l'image originale bruitée *doge_noise.png* et ce n'est pas logique, tandis qu'avec la distance Tchebychev, les résultats sont plus cohérents. En effet, nous avons considéré que l'image bruitée *doge_noise.png* s'apparente plus à la qualité de nos images jpeg. (Rosebrock, 2014)

Dans le fichier *generate.py*, nous avons apporté des modifications pour vérifier toutes les trames, plutôt qu'une à chaque seconde comme dans la question 2. La raison est toute simple : préciser la détection du minutage.

Finalement, dans le fichier *find.py*, nous avons ajouté un seuil pour le traitement des JPEG. En effet, puisqu'il s'agit d'un format de données avec perte, il est impossible d'avoir une distance de 0 par rapport à la trame originale (non compressée) du vidéo, peu importe la précision du minutage. Dans le cas des images PNG, comme il n'y a pas de perte et qu'on compare avec chaque trame, toute distance supérieure à 0 représente une image qui ne fait pas partie d'aucun vidéo. Pour les JPEG, nous avons procédé par essai-erreur afin de trouver un seuil convenable, et notre choix s'est arrêté sur la valeur de 0,2, mais il est important de noter ici que les résultats étaient loin d'être parfaits. À vrai dire, c'était impossible, avec notre algorithme, d'avoir un résultat véritablement satisfaisant. Voici un tableau qui synthétise les différents paramètres étudiés, que nous verrons plus en détails :

Tableau 1 – Résultats obtenus avec notre algorithme. ©2020 Mariam Sarwat, Jean-Olivier Dalphond.

Paramètre	**Observation (JPEG, PNG)**	
	JPEG	PNG
Temps d'exécution moyen (s) N.B. : Exclut la génération des histogrammes	3,0946705	1,9962855
Validité (%)	94	100
Écart de minutage (s)	2,157718	0
Stockage utilisé (Mo)	91,0	

La génération des histogrammes se fait plus rapidement que nous avons imaginé, soit en 68,7 secondes. La recherche dans les vidéos demande nettement plus de temps que pour l'algorithme de Carol à la question 2, mais dans un facteur plus raisonnable qu'anticipé. Pour trouver une image une fois les histogrammes générés, le délai est beaucoup plus long. En ce qui concerne les JPEG, le temps total est de 618,9341 secondes, ou 3,0946705 secondes par image en moyenne. Pour les PNG, c'est 399,2571 secondes au total, ou 1,9962855 secondes en moyenne, soit une meilleure performance par rapport à JPEG. Cette différence est due au fait qu'on passe par la totalité des histogrammes pour retrouver celui avec la plus petite distance dans le cas de JPEG, tandis que pour PNG on arrête la recherche dès qu'on retrouve une distance égale à 0.

La taille des fichiers d'index est de 95379795 octets, ou 91,0 Mo. C'est 27 fois plus volumineux que pour la question 2, mais c'est quand même très raisonnable, surtout considérant qu'on a un résultat parfait avec les PNG.

C'est justement la validité des trouvailles qui fait notre fierté! On obtient 100% avec PNG, ce qui était l'objectif de base. Avec les JPEG, c'est 94%, ce qui peut sembler raisonnable. L'analyse des deltas de minutage donne un portrait plus représentatif. Pour les PNG, on a un delta moyen de 0 et une distance moyenne de 0, car toutes les images ont été reconnues sans faute. Pour les JPEG, le delta maximal trouvé est de 12 secondes, ce qui n'est vraiment pas précis, et bien qu'on trouve 5 valeurs nulles de delta, le delta moyen est de 2,157718 secondes. En d'autres mots, ce n'est pas vraiment fiable. On trouve d'ailleurs 6 vidéos identifiés faussement (faux positifs), et 12 vidéos présents non identifiés (faux négatifs).

**Nos résultats pour la question 2 et 3 se retrouvent dans des fichiers Excel inclut dans la remise. On a deux onglets, un pour les résultats des images jpeg et un pour les png.

Références

Rosebrock, A. (2014, juillet 14). *How-To: 3 Ways to Compare Histograms using OpenCV and Python*. Récupéré sur pyimagesearch: https://www.pyimagesearch.com/2014/07/14/3-ways-compare-histograms-using-opencv-python/