INF6804 - Visions par ordinateur

Hiver 2024

Travail Pratique 01

Groupe 01

Alexandre Gélinas - 2083465

Elizabeth Michaud - 2073093

Soumis à : Khalil Sabri

Guillaume-Alexandre Bilodeau

Date :

9 février 2024

Table des matières

1. Présentation des deux approches à comparer

Le sujet de ce travail pratique consiste à comparer deux méthodes de description d'image sur une banque d'image fournie. La première méthode est basée sur les matrices de co-occurrence (MC) et la deuxième se base sur les modèles binaires locaux (LBP). La méthode MC se base sur des statistiques de positionnement des différents degrés de couleur selon un vecteur prédéfini (dx et dy ou distance et angle) pour la comparaison. On essaie ainsi avec cette méthode de déterminer si notre image contient des motifs de positionnement de couleur de pixel qui se répètent et de pouvoir le retrouver sur une image du même thème. En ce qui concerne la méthode LBP, la description d'une image se base sur l'intensité lumineuse de pixel formant un cercle autour du pixel en cours d'évaluation. On forme ensuite un vecteur pour chaque valeur du cercle indiquant si notre valeur est plus grande ou plus petite que notre pixel central.

Ainsi, notre but est d'effectuer une recherche d'image par le contenu, ou CBIR, afin de retrouver les images similaires. De plus, nous devrons déterminer si le fait d'effectuer des boîtes englobantes sur nos images a un impact sur nos résultats. Pour comparer nos images, nous devrons utiliser plusieurs techniques de comparaison afin d'évaluer la précision des 2 méthodes ainsi que de leur temps de calcul.

2. Hypothèses de performance pour des cas spécifiques

Selon notre compréhension théorique des deux approches étudiées, nous pensons que la méthode des matrices de co-occurrence sera meilleure que la méthode LBP lorsque le contenu des régions comparées est relativement texturé. Comme la méthode se base sur des statistiques de positionnement de couleur ou d'intensité lumineuse de pixel par rapport à un autre, la texture d'une image qui se répète énormément sur un objet, comme un motif carrelé, serait beaucoup plus détectée par cette méthode. Comme nous faisons une sommation de la récurrence du positionnement relatif de deux couleurs, les motifs de la texture de l'image devraient avoir un nombre plus élevé de récurrences dans le descripteur. Donc, si une autre image contient le même motif de texture, les deux images seront identifiées comme étant semblables.

D'un autre côté, si l'objet de l'image est moins texturé et plus uniforme, la méthode des modèles binaires locaux devrait être meilleure. Cette dernière se base sur l'uniformité des pixels par région, ce qui permettra de facilement caractériser des régions similaires de l'image. Par exemple, une image d'un objet à la lumière sera définie comme semblable à une image du même objet plus sombre, ce qui n'est pas le cas de la méthode MC. De plus, elle prend en compte la position du motif. Ainsi, une image dont les pixels semblables sont près, c'est-à-dire une image uniforme, va être mieux caractérisée par ce descripteur. C'est ainsi pourquoi nous pensons que LBP serait meilleure dans ce contexte.

Enfin, pour l'efficacité des résultats, nous pensons que la méthode par matrices de co-occurrence serait plus rapide, car elle utilise moins de données afin de former son descripteur.

3. Hypothèses de performance concernant les boîtes englobantes (bounding boxes)

En ce qui concerne les performances pour les boîtes englobantes, nous croyons que cela sera plus efficace uniquement lorsque l'objet en question a une forme qui occupe au maximum le volume d'un rectangle. Si par exemple, nous avons un dauphin courbé, nous ne pourrons pas vraiment mieux définir l'objet en question avec un rectangle englobant, car l'arrière-plan ne pourra pas être complètement supprimé par la boîte. Dans un cas comme celui-ci, il serait plus efficace d'identifier directement les contours du dauphin pour éliminer le fond. Ainsi, il restera beaucoup plus de pixels qui n'appartiennent pas à l'objet que si celui-ci avait été en forme de rectangle par exemple. Un objet rectangulaire comme un téléviseur pourrait facilement réduire au maximum l'arrière-plan avec une boîte englobante. De ce fait, comme nous avons beaucoup d'images qui prennent de l'espace que nous ne pouvons pas enlever par les boîtes, la performance ne devrait pas être énormément meilleure. Cependant, nous nous attendons quand même à voir une légère amélioration des 2 méthodes de descripteurs étant donné que ces dernières pourront plus facilement identifier un objet qui à une dimension d'image similaire à celle de la requête (sachant que la forme de la requête sera semblablement la même que ceux dans la base de données).

4. Description des expériences, des données et critères d'évaluation

Pour débuter nos expériences, nous avons tout d'abord appliqué les boîtes englobantes sur nos images afin d'avoir directement l'ensemble des données prêtes à usage. Nous avons essayé de respecter au maximum l'objet en entier sans rogner une partie de l'objet pour bien représenter la complexité de ce type d'objet dans la vie de tous les jours. Par la suite, nous avons implémenté les différentes fonctions de descriptions d'images qui seront abordées au point suivant. Nous avons appliqué nos méthodes avec plusieurs techniques pour pouvoir comparer aussi les différentes implémentations de nos descripteurs. De plus, nous avons passé plusieurs paramètres à ces dernières afin de bien tester le descripteur sous toutes ses formes. En ce qui concerne les méthodes de comparaison, nous avons utilisé l'intersection, la norme 1, la norme 2 et la distance de Bhattacharyya. L'ensemble de ces méthodes ont été implémentées avec les divers outils de la librairie NumPy afin d'accélérer le processus de traitement sur les descripteurs. Au niveau des métriques d'évaluation, nous prenions les 3 meilleures images par requêtes selon nos méthodes de comparaison et nous inscrivions si l'image de la base de données était bel et bien celle voulue par la requête. On obtient ainsi un score sur 3 pour chaque requête et chaque méthode de comparaison.

En ce qui concerne les images de la base de données, il y avait plusieurs caractéristiques que nous avons remarquées qui ont pu affecter nos résultats. Premièrement, les objets comme l'avion et la voiture sont très longs et ont une petite hauteur, ce qui peut parfois faire croire à nos méthodes que ces deux objets sont similaires. Deuxièmement, la couleur de l'image n'est pas toujours de la même qualité, ce qui peut affecter grandement la comparaison. Troisièmement, l'image sur le visage est très similaire entre elles, ce qui permet de facilement la retrouver lors de la comparaison. Seul l'arrière-plan de chacune change, ce qui permet de retrouver les mêmes couleurs de pixel environ aux mêmes endroits. Quatrièmement, certains objets sur les images ne sont pas

similaires, mais les images comportent des fonds similaires. Par exemple, le fond vert foncé de l'image ball_query et le vert foncé de l'image cat_5. Dernièrement, les images sur les cornichons ont une très grande taille de pixel, ce qui demande parfois 100 fois plus de temps de calcul uniquement pour ces images comparativement aux autres images. La qualité de l'image vient donc jouer un très grand rôle dans la vitesse d'exécution de nos algorithmes.

5. Description des deux implémentations utilisées

Notre implémentation de la méthode des matrices de co-occurrence utilise la fonction "graycomatrix" de la librairie Scikit-image. Cette fonction prend en paramètre une image en noir et blanc, un tableau de distance, un tableau d'angles en radian et d'autres paramètres optionnels. Elle retourne un descripteur de taille 256 x 256, soit un descripteur qui compare chaque intensité de gris avec chaque intensité de gris. De plus, chaque case de la matrice 256 x 256 est une matrice de taille *m x n* où *m* est le nombre de distances dans le tableau de distances passé en paramètre à la fonction et *n* est le nombre d'angles dans le tableau d'angles passé en paramètre. Les distances que nous avons choisies sont 1, 5 et 10 et les angles choisis sont 0, pi / 2, pi et 3 * pi / 2. Nous avons utilisé l'option "normed = True" dans les paramètres de la fonction pour que les différentes tailles d'images n'influencent pas la comparaison des descripteurs. Comme la méthode des matrices de co-occurrence compare normalement des images en couleur et non des images en noir et blanc comme le fait la fonction "graycomatrix", nous avons appelé la fonction 3 fois, c'est-à-dire une fois pour chaque canal de couleur (rouge, vert, bleu). Cela a donc nécessité un prétraitement des images. Nous lisons une image à l'aide de la librairie OpenCV en fournissant à la fonction "imread" le chemin vers l'image à lire et en précisant que nous voulons l'image en couleur. La fonction nous retourne un tableau sur lequel il faut effectuer un traitement pour obtenir 3 matrices distinctes chacune comportant uniquement les intensités d'un canal de couleur. Nous appelons ensuite 3 fois la fonction "graycomatrix" pour obtenir les descripteurs liés à chaque canal de couleur. Donc, si nous avons un tableau de 3 distances et de 4 angles, pour chaque image, nous obtenons 3 canaux x 3 distances x 4 angles = 36 descripteurs. Par la suite, nous effectuons la comparaison de l'image requête (query) avec chacune des images de la base de données. Pour chaque paire d'images à comparer et pour chaque méthode de comparaison de descripteurs, nous comparons chacune des 36 paires de descripteurs. Par exemple, nous comparons le descripteur rouge à distance d'un pixel à un angle de pi/2 de l'image airplane_query avec le descripteur rouge à distance d'un pixel à un angle de pi/2 de l'image ball_4. Le résultat pour une paire d'images pour une seule méthode de comparaison est donc une sommation des 36 résultats de comparaison des 36 paires de descripteurs. Une fois que les comparaisons de l'image requête ont été faites pour toutes les images de la base de données, nous pouvons trouver les 3 images les plus semblables à l'image requête pour chaque méthode de comparaison.

Pour ce qui est de la méthode des modèles binaires locaux, nous avons utilisé la fonction "local binary pattern" qui venait directement de la librairie de Scikit-image. Celle-ci nous permet d'y passer les différents paramètres comme le nombre de points d'échantillonnage ainsi que le rayon. En ce qui concerne les paramètres, nous avons décidé d'effectuer nos tests avec des rayons entre 1 et 5 et du nombre de points d'échantillonnage entre 4 et 24

afin de bien avoir une représentation de la méthode sous plusieurs paramètres. Ces nombres sont tirés du site MathWorks et sont les paramètres habituels utilisés par cette méthode.

En ce qui concerne les boîtes englobantes, nous avons manuellement appliqué ces dernières sur chacune des images afin de bien caractériser celles-ci et de bien représenter l'objet dans son ensemble.

6. Présentation des résultats de tests

6.1. Résultats pour la précision

Nous avons mis en forme nos résultats en tableau pour chaque méthode pour les comparer plus facilement.

Les tableaux I et II montrent les résultats obtenus pour la méthode de classification par matrices de co-occurrence. La note attribuée par requête par comparaison est sur 3. La colonne de total par requête est sur 12 et la colonne à la droite de celle-ci représente le score en pourcentage. Pour la ligne de total par méthode de comparaison, elle est sur 24 et la ligne en dessous de celle-ci représente le score en pourcentage.

Tableau I : Résultats de précisions pour la méthode des matrices de co-occurrence avec images non modifiées

Requête / Comparaison	Intersection	Norme 1	Norme 2	Bhattacharyya	Total	Total (%)
Airplane	3	3	3	3	**12**	**100%**
Ball	1	1	0	0	**2**	**17%**
Car	3	3	2	3	**11**	**92%**
Cat	3	3	2	3	**11**	**92%**
Dolphin	1	1	0	1	**3**	**25%**
Face	2	2	2	3	**9**	**75%**
Lotus	0	0	0	0	**0**	**0%**
Pickle	3	3	3	3	**12**	**100%**
Total	**16**	**16**	**12**	**16**	**60**	**63%**
Total (%)	**67%**	**67%**	**50%**	**67%**	**63%**	

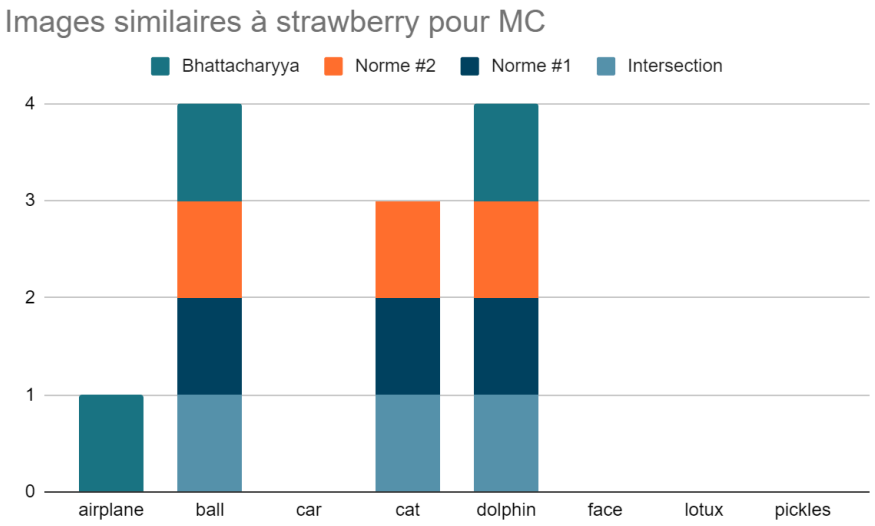
Selon le Tableau I, nous pouvons voir que la méthode des matrices de co-occurrence avec images non modifiées a réussi à bien classer les images semblables dans le top 3 dans 63% des cas.

Tableau II : Résultats de précisions pour la méthode des matrices de co-occurrence avec images "cropped"

Requête / Comparaison	Intersection	Norme 1	Norme 2	Bhattacharyya	Total	Total (%)
Airplane	2	2	2	2	**8**	**67%**
Ball	0	0	0	0	**0**	**0%**
Car	3	3	2	1	**9**	**75%**
Cat	3	3	2	3	**11**	**92%**
Dolphin	1	1	1	1	**4**	**33%**
Face	3	3	3	3	**12**	**100%**
Lotus	0	0	0	0	**0**	**0%**
Pickle	3	3	1	3	**10**	**83%**
Total	**15**	**15**	**11**	**13**	**54**	**56%**
Total (%)	**63%**	**63%**	**46%**	**54%**	**56%**	

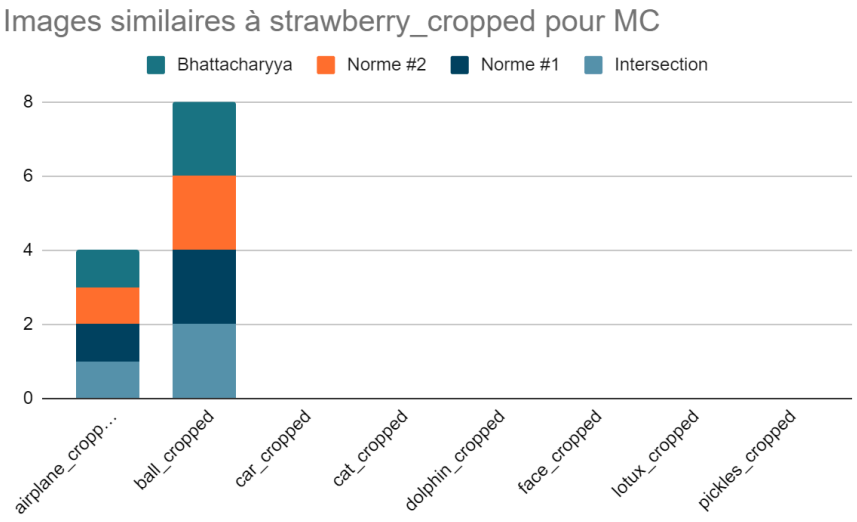
Selon le Tableau II, nous pouvons voir que la méthode des matrices de co-occurrence avec images "cropped" a réussi à bien classer les images semblables dans le top 3 dans 56% des cas.

Le diagramme à bande qui suit montre les résultats pour la requête sur l'image strawberry qui n'a pas d'image équivalente dans la base de données.



Nous pouvons remarquer que notre implémentation de MC a principalement trouvé des similitudes entre l'image strawberry et les images de ball, de cat et de dolphin.

Le diagramme à bande qui suit montre les résultats pour la requête sur l'image strawberry_cropped.



Nous pouvons remarquer que notre implémentation de MC a principalement trouvé des similitudes entre l'image strawberry et les images de ball, et de airplane.

Pour ce qui est des modèles binaires locaux, nous avons fait un tableau III qui comprend l'ensemble des résultats obtenus. Comme nous avons fait l'expérience avec les divers paramètres du descripteur, cela nous permet de bien représenter ce dernier globalement. Ainsi, nous avons effectué nos tests avec les paramètres 1, 3 et 5 pour le rayon (R) et 4, 8, 12, 16, 20 et 24 pour le nombre d'échantillons (P). Chaque 3 meilleurs résultats de chaque requête pour chaque comparateur sont ensuite ajoutés au total afin de comptabiliser les résultats.

Tableau III : Résultats de précisions pour la méthode des modèles binaires locaux

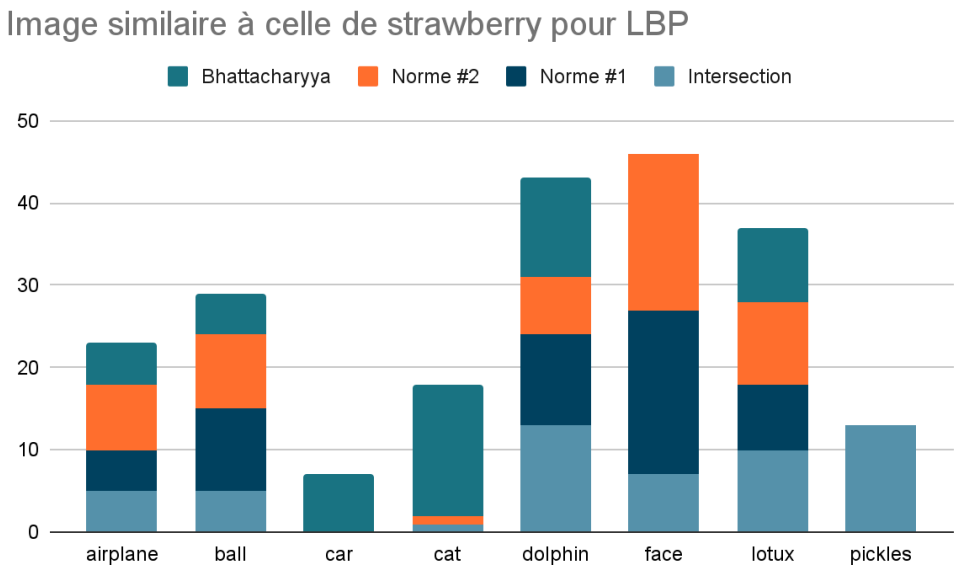
Requête \ Comparaison	Intersection	Norme #1	Norme #2	Bhattacharyya	**Total**	**Total (%)**
airplane	22	49	42	18	**131**	**61%**
airplane_cropped	17	40	32	37	**126**	**58%**
ball	7	7	10	6	**30**	**14%**
ball_cropped	4	2	7	10	**23**	**11%**
car	17	50	40	35	**142**	**66%**
car_cropped	19	46	52	23	**140**	**65%**
cat	11	28	28	24	**91**	**42%**
cat_cropped	14	30	26	21	**91**	**42%**
dolphin	0	1	4	2	**7**	**3%**
dolphin_cropped	2	12	9	2	**25**	**12%**
face	22	44	38	15	**119**	**55%**
face_cropped	31	51	48	18	**148**	**69%**
lotus	15	18	18	11	**62**	**29%**
lotus_cropped	11	16	9	8	**44**	**20%**
pickles	54	54	54	18	**180**	**83%**
pickles_cropped	53	54	52	18	**177**	**82%**

Total	**299**	**502**	**469**	**266**	**1536**	**44%**
Total (%)	**35%**	**58%**	**54%**	**31%**	**44%**	

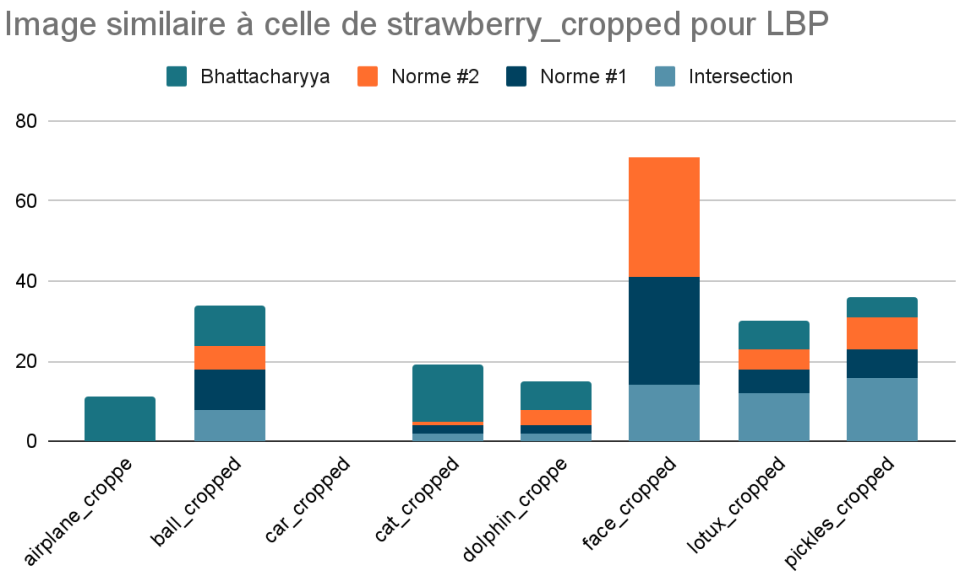
Selon le tableau III, nous pouvons voir que le résultat des 3 meilleures images pour l'ensemble du descripteur est d'environ 44 %.

Tableau IV : Comparaison des résultats avec les boîtes englobantes pour LBP

Total Normal	Total Cropped
44%	45%



En ce qui concerne l'image de fraise pour LBP, celle-ci fut associée avec les images de visage, de dauphin et de lotus le plus souvent. De plus, l'association fut moins significative avec les images de voiture, de chat et de cornichons.



6.2. Résultats pour l'efficacité (temps de calcul)

Pour l'efficacité de nos descripteurs, la méthode des matrices de co-occurrence a pris environ 185 secondes, soit 3 minutes et 5 secondes pour les images sans la boîte englobante et environ 108 secondes, soit 1 minute et 48 secondes pour les images avec la boîte englobante.

Pour ce qui est de la méthode des modèles binaires locaux, celle-ci lui a pris environ 383 secondes, soit 6 minutes et 23 secondes pour les images sans la boîte englobante et environ 92 secondes, soit 1 minute et 32 secondes pour les images avec la boîte englobante.

7. Discussion des résultats et retour sur les hypothèses

À l'aide de la méthode MC, nous avons eu de la facilité à trouver des images semblables pour les queries d'avion, de voiture, de chat et de cornichons. Pour la requête de chat, nous pensons que la méthode MC a facilement pu identifier des images similaires grâce aux motifs présents sur le chat de la query qui sont très semblables aux motifs dans les images de chat de la base de données. Pour la requête de cornichons, nous remarquons que les 3 images qui sont constamment classées comme les plus similaires à l'image query sont pickles_1, pickles_2 et pickles_4. Cela est probablement à cause du fond de l'image. En effet, le pot de cornichons dans l'image pickles_query est posé sur le plancher et il en est de même pour le pot de cornichons des images pickles_1, pickles_2 et pickles_4, alors que ce n'est pas le cas du pot de cornichons des images pickles_3 et pickles_5. D'un autre côté, la méthode MC n'a pas réussi à bien identifier les images similaires à l'image lotus_query. Nous pensons que cela est dû au fait que le lotus de la requête est très blanc avec un fond noir, ce qui n'est le cas d'aucune image de lotus de la base de données.

Également, nous pouvons observer que le fait d'avoir des boîtes englobantes n'aide pas à mieux trouver les images similaires. En effet, nous avons une perte de précision de 7% pour les images "cropped" par rapport aux images régulières. Une des requêtes dont la performance a le plus diminué est la requête d'avion. Nous pensons que nous avons obtenu de meilleurs résultats avec les images normales, car l'image airplane_query comporte beaucoup de gazon ainsi qu'un contour blanc, ce qui est également le cas des images d'avions dans la base de données. Lorsque nous utilisons les images avec une boîte englobante, nous obtenons pour chacune des méthodes de comparaison que l'image cat_cropped_2 est similaire à airplane_cropped_query. Cela est probablement causé par le fait que l'image cat_cropped_2 a un fond avec beaucoup de vert et que ce vert est semblable au vert du gazon de l'image airplane_cropped_query.

Pour ce qui est de la requête strawberry, les images qui ont été identifiées comme semblables par presque toutes les méthodes de comparaison sont les images ball_2, cat_4 et dolphin_4. Nous pouvons constater que toutes ces images ont un fond noir tout comme l'image strawberry. Lorsqu'on réduit le fond avec les images avec boîtes englobantes, nous obtenons des résultats différents. Effectivement, l'image airplane_cropped_4 revient plusieurs fois dans les résultats. Nous pensons que cela est dû au fait que la fraise et l'avion dans l'image airplane_cropped_4 sont presque exactement de la même couleur.

En analysant ces résultats pour LBP, les images de cornichons ont été celles qui ont eu le plus de facilité à être retrouvées par ce descripteur. D'un autre côté, les images de dauphin ont été les plus difficiles à retrouver. Pour ce qui est des boîtes englobantes, on peut voir une légère amélioration des résultats, ce qui correspond bel et bien à notre hypothèse. De plus, on peut même remarquer que certaines images ont été moins performantes avec l'aide des boîtes englobantes comme l'avion et le lotus. Bref, on ne peut pas affirmer définitivement qu'il y aura toujours une amélioration des résultats avec cette modification. Pour la similarité des images avec la fraise, nous pensons que cela est dû à de grande surface de couleur similaire pour les images qui ont été les plus associées. Par exemple, l'image de cornichon n'avait pas beaucoup de couleur qui occupait une grande surface sur l'image comparativement au visage. Ceci semble ainsi confirmer notre hypothèse sur cette méthode avec LBP. Pour l'image de fraise avec les boîtes englobantes, on a remarqué environ le même principe que mentionné plus haut. Les images qui avaient beaucoup d'arrière-plans comme les cornichons ou le lotus ont considérablement changé la manière dont ils étaient interprétés par le descripteur. N'ayant plus autant d'arrière-plans, les images comme ceux du cornichon ont pu afficher des couleurs plus uniformes sur de plus grande surface, ce qui a augmenté leur score d'association. Inversement, ceux dont l'arrière-plan était uniforme comme le lotus ont vu réduire leurs scores d'association, car les détails de l'objet n'étaient pas complètement uniformes. Enfin, comme les images de visage n'ont presque pas été modifiées par les boîtes encombrantes, les résultats ont quasi été les mêmes.

En somme, la méthode MC est plus efficace pour l'image de l'avion, de la voiture, du chat, du visage et des cornichons et la méthode LBP est plus efficace pour l'image du ballon et du lotus. Bien évidemment, l'image du dauphin ne peut pas vraiment être évaluée, car les performances des 2 méthodes ne sont pas significatives afin d'y mettre un résultat.

En ce qui concerne l'efficacité, on peut voir que la méthode des matrices de co-occurrence prend beaucoup moins de temps. Cependant, nous avons appliqué beaucoup plus de paramètres et avons beaucoup plus de résultats pour les modèles binaires locaux.

Pour revenir à nos hypothèses, celles-ci ont assez bien été validées. Cependant, comme les images de la vie de tous les jours ne sont pas exprimées par une plus grande texturation ou uniformisation, il n'est pas possible de déterminer une méthode de descripteur qui est significativement meilleure que l'autre. Chacun a ses forces et ses faiblesses. Les images comme le chat qui sont beaucoup plus texturées furent mieux associées par la méthode des matrices de cooccurrences que des modèles binaires locaux. Cependant, les images comme ceux du lotus qui ont beaucoup moins de texture et qui sont plus uniformes sont beaucoup plus associées avec cette dernière méthode. Pour ce qui est de l'amélioration de nos tests, nous aurions pu augmenter le nombre d'images texturées et non texturées afin de voir une différence beaucoup plus significative à travers les différents types d'images. Bien que cette différence soit un peu visible dans nos résultats, elle n'est pas suffisamment significative afin d'affirmer nos hypothèses avec conviction. Il pourrait toujours y avoir un type d'image plus générique qui n'est pas supporté par un de nos descripteurs. De plus, nous aurions pu augmenter le nombre de tests sur les boîtes englobantes en utilisant une différente précision de boîte. Ainsi, nous aurions peut-être pu y voir un modèle concernant ces dernières qui ne semblent pas être présentes dans nos résultats.

En ce qui concerne nos résultats sur les images de dauphins, celles-ci semblent avoir été parmi les plus difficiles à détecter par nos 2 descripteurs. Nous pensons que cela est dû à la qualité des images de ces derniers. De plus, il y avait une grande différence de luminosité entre chaque image comparée, ce qui pourrait potentiellement affecter nos descripteurs. Enfin, même si les boîtes englobantes avaient pu réduire cela, la forme des dauphins nous empêche de retirer la majeure partie de l'arrière-plan, ce qui ne permet pas d'améliorer nos performances. Nous aurions pu utiliser le descripteur par contour qui aurait probablement été plus significatif étant donné que les dauphins de nos images avaient une forme très similaire, peu importe l'image. Ce dernier nous aurait permis de bien encadrer le dauphin de notre image pour comparer son contour avec celui des autres. Cependant, ceci n'aurait pas nécessairement été plus efficace pour les autres types d'images comme les images de lotus ou de cornichons.

Références

[1] OpenCV: https://docs.opencv.org/4.x/index.html

[2] Scikit-image: https://scikit-image.org/docs/stable/auto_examples/index.html

[3] NumPy: https://docs.scipy.org/doc/numpy/user/quickstart.html

[4] MathWorks: Extract local binary pattern (LBP) features - MATLAB extractLBPFeatures (mathworks.com)