Reconnaissance et indexation de formes

Quentin Cormier Yassine Hamoudi

4 mai 2015

Table des matières

1 Introduction

Il existe de nombreuses manières de caractériser une image en vue de reconnaître l'objet qu'elle représente. Cependant, la plupart des descripteurs "élémentaires" (périmètre de l'image, nombre de composantes connexes, surface ...) sont peu efficaces individuellement et nécessitent d'être combinés astucieusement pour donner de bons résultats.

Nous avons recherché un moyen robuste et plus concis de classifier en 70 catégories le millier d'images du projet. Notre méthode consiste à étudier la propagation d'une onde à l'intérieur d'une image. En effet, cette propagation dépend directement de la forme de l'objet. De la même manière que deux tambours de forme circulaire renvoient des sons proches, deux images de pommes devraient propager les ondes de façon similaires. Par ailleurs, cette méthode semble insensible à de nombreuses perturbations (rotation, translation, dilatation...).

Notre travail s'inspire de [?] et [?]. Nous étudierons dans un premier temps quels descripteurs peuvent être construits à partir du problème de la propagation d'une onde dans une image. Nous chercherons également la métrique la plus appropriée pour comparer deux vecteurs descripteurs associés à deux images. Nous vérifierons enfin théoriquement et expérimentalement la résistance de notre algorithme à différentes perturbations (rotation, bruit...).

2 Méthode

On dispose d'un ensemble $\mathcal{E}$ d'images en noir et blanc regroupées en 70 catégories. Les images de chaque catégorie représentent un même objet (par exemple : une pomme, une voiture, un os...). L'objectif est de classer de nouvelles images à l'intérieur d'une des catégories précédentes.

Pour ce faire, nous allons encoder chaque image par un vecteur de réels (le *descripteur*), et utiliser une distance entre descripteurs pour classifier les images inconnues : étant donné une image M, nous calculons son descripteur d_M puis nous cherchons l'image M' de $\mathcal{E}$ qui minimise $d(d_M, d_{M'})$ (pour une distance d fixée). On prédit alors la classe de M comme étant celle de M'.

2.1 Normalisation des images

Avant de construire les descripteurs, nous appliquons aux images une étape de normalisation.

Tout d'abord, nous recadrons l'image jusqu'à ce que chaque bord contienne au moins un pixel blanc.

Ensuite, nous redimensionnons l'image de sorte que le maximum de la largeur et de la hauteur soit de 50 pixels. Cette opérations permet de se ramener à des images de taille au plus $N = 50 * 50 = 2500$. En effet, il sera nécessaire par la suite de calculer les valeurs propres d'une matrice de taille $N * N$. Cette opération étant relativement coûteuse en temps de calcul, nous avons cherché le redimensionnement optimal qui garantisse un temps de calcul raisonnable (environ 30s pour $N = 2500$).

Enfin nous ajoutons un cadre noir autour de l'image. Ceci permet de fermer la cavité définie par le motif de l'image.

Exemple :

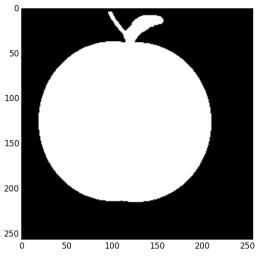
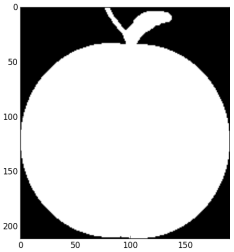


Image initiale



Bords rognés

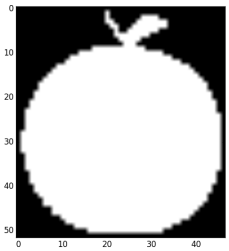


Image après redimensionnement et ajout du cadre noir

2.2 Onde associée à un domaine

Notre descripteur prend appui sur le phénomène physique suivant : lorsqu'une onde stationnaire se propage dans une cavité, la propagation n'a lieu que selon certaines fréquences discrètes qui dépendent de la forme de la cavité.

En effet, une onde se propageant dans une cavité vérifie l'équation suivante :

$$\Delta E + \frac{1}{v^2}\frac{\partial^2 E}{\partial t^2} = 0$$

avec par exemple $E(x, y, t) = 0$ sur les bords de la cavité (condition aux limites de Dirichlet).

Si l'on recherche les solutions stationnaires de l'équation précédente, on pose $E = f(x, y)e^{iwt}$, et on obtient :

$$\begin{cases} \Delta f - \frac{w^2}{v^2} f = 0 \\ f \text{ s'annulant sur les bords de la cavité.} \end{cases} \quad (1)$$

f peut donc être vue comme un vecteur propre de l'opérateur Laplacien et $\frac{w^2}{v^2}$ la valeur propre associée.

Ainsi, les valeurs propres du Laplacien sont corrélées à la forme de la cavité. Nous avons donc essayé d'utiliser ces valeurs pour construire notre descripteur. Par ailleurs, la propagation d'une onde est inchangée lorsque la cavité est translatée, agrandie, ou subit une rotation. Par conséquent, notre descripteur devrait remplir la plupart des contraintes de stabilité souhaitées.

2.3 Calcul des valeurs propres du Laplacien d'une image

Pour calculer les valeurs propres du Laplacien dans le domaine défini par l'image, on "encode" l'image de taille $N = n * n$ dans un vecteur de taille n^2. On construit ensuite la matrice de l'opérateur Laplacien discrétisé Δ^*, de taille N*N :

$$\Delta^* = \begin{pmatrix} -A_n & I_n & \cdots & 0 \\ I_n & -A_n & \ddots & \vdots \\ \vdots & \ddots & \ddots & I_n \\ 0 & \cdots & I_n & -A_n \end{pmatrix}$$

avec :

$$A_n = \begin{pmatrix} -4 & 1 & \cdots & 0 \\ 1 & -4 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \cdots & 1 & -4 \end{pmatrix}$$

Ensuite, on multiplie chaque ligne $i + Nj$ ($i, j \in \{1, ..., n\}$) de Δ^* par 0 ou 1, suivant la valeur du pixel $M(i, j)$. On obtient alors une nouvelle matrice Δ^*_M, toujours de taille $N * N$, qui vérifie pour tout vecteur E de taille N :

$$(\Delta^*_M E)(x, y) = \begin{cases} (\Delta^* E)(x, y) & \text{lorsque } (x, y) \text{ est dans le domaine définit par l'image} \\ 0 & \text{sinon} \end{cases}$$

Ainsi, si E vecteur de taille N est un vecteur propre de Δ^*_M associé à la valeur propre λ, on a :

$$(\Delta^*_M E)(x, y) = \begin{cases} \lambda E(x, y) & \text{lorsque } (x, y) \text{ est dans le domaine définit par l'image} \\ 0 & \text{sinon} \end{cases}$$

Donc E est bien une solution de l'équation discrétisée (1) : E correspond à une onde stationnaire associée à la valeur propre λ pour le domaine définit par l'image.

On peut généraliser facilement cela pour des images rectangulaires de taille $N = h * w$.

Le calcul des valeurs propres de la matrice (creuse) précédente s'effectue avec la fonction `numpy.linalg.eigvals`.

2.4 Calcul du descripteur associé aux valeurs propres

Nous construisons deux types de descripteurs à partir du spectre trié par ordre croissant $\{\lambda_1, \lambda_2, ...\}$ calculé précédemment. Ces descripteurs (présentés dans [?] et [?]) ont été choisis pour leur invariance à différentes perturbations (rotation, translation, ...).

— Descripteur de type 1 : $\{\frac{\lambda_1}{\lambda_2}, \frac{\lambda_1}{\lambda_3}, \frac{\lambda_1}{\lambda_4}, ...\}$

— Descripteur de type 2 $\{\frac{\lambda_1}{\lambda_2}, \frac{\lambda_2}{\lambda_3}, \frac{\lambda_3}{\lambda_4}, ...\}$

Le choix du descripteur et sa taille (ainsi que la métrique à utiliser pour comparer les descripteurs) seront discutés dans la partie suivante, à partir de résultats expérimentaux.

3 Résultats

Nous exposons les résultats obtenus grâce à la méthode détaillée précédemment. Nous allons étudier dans un premier temps la sensibilité de notre algorithme aux perturbations (rotation, redimensionnement, bruit, ...), puis nous détaillerons les résultats de classification sur le dataset d'images.

3.1 Sensibilité aux perturbations

Les valeurs propres du Laplacien de Dirichlet vérifient un certain nombre de propriétés mathématiques qui garantissent que notre descripteur est insensible au redimensionnement, à la rotation et à la translation. Nous vérifions expérimentalement ces propriétés ci-dessous.

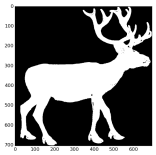
Redimensionnement Etant donné un domaine Ω et un facteur $a > 0$, on a $\lambda_k(a\Omega) = \frac{\lambda_k(\Omega)}{a^2}$ (voir [?]). Or, notre descripteur utilise des rapports de valeurs propres, il est donc inchangé par redimensionnement : $\frac{\lambda_k(a\Omega)}{\lambda_m(a\Omega)} = \frac{\lambda_k(\Omega)}{\lambda_m(\Omega)}$. Nous avons calculé différents rapports pour l'image `camel-1.pgm`, les résultats figurent table 1. Les variations d'une image à l'autre peuvent s'expliquer par les dégrations des contours suite au redimensionnement. Les rapports restent tout de même très proches. En pratique, nous utilisons une longueur de 50 pixels afin d'obtenir des temps de calcul raisonnables (environ 30s pour calculer les valeurs propres associées à une image de taille 50x50).

	λ_1/λ_2	λ_1/λ_3	λ_1/λ_4	λ_2/λ_3	λ_3/λ_4	λ_4/λ_5
75 pixels	0.63	0.42	0.36	0.67	0.84	0.89
50 pixels	0.61	0.42	0.32	0.68	0.83	0.89
25 pixels	0.53	0.39	0.31	0.73	0.80	0.81

TABLE 1 – Rapports de valeurs propres en fonctions de la longueur de l'image `camel-1.pgm` redimensionnée (l'image initiale est de taille 346x346)

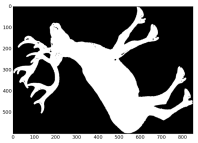
Translation Nous recadrons systématiquement l'image afin de conserver le plus petit rectangle contenant la figure. Ceci nous permet d'être insensible aux translations.

Rotation Il a été démontré mathématiquement que les valeurs propres sont inchangées lorsque le domaine subit une rotation (voir [?]). Ce résultat se vérifie facilement à partir de quatre rotations appliquées sur `deer-20.pgm`. Les valeurs propres associées à chaque figure sont regroupées dans la table 2.

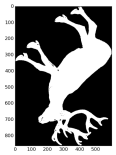


0 degré

30 degrés



120 degrés



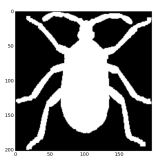
220 degrés

	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8
0 degré	118.57	177.16	252.35	285.53	361.22	404.89	420.69	489.36
30 degrés	120.28	182.75	257.54	297.97	370.39	413.53	426.99	472.08
120 degrés	120.29	182.88	266.09	308.13	372.66	414.83	428.60	472.21
220 degrés	120.42	182.95	250.94	299.81	366.01	406.55	424.32	478.39

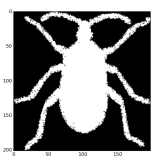
TABLE 2 – Valeurs propres associées à `deer-20.pgm` en fonctions de l'angle de rotation

Bruit Le bruit peut déformer le domaine de l'image et modifier par conséquent les valeurs propres. Cependant, les premières valeurs propres $(\lambda_1, \lambda_2, \lambda_3, \dots)$ correspondent à la fondamentale et aux premières harmoniques, et donc aux composantes de la solution au Laplacien de Dirichlet de longueur d'onde élevée. Ces longueurs d'ondes étant très supérieures aux longueurs caractéristiques relatives à l'ajout de bruit (quelques pixels au plus), on peut s'attendre à ce qu'une déformation relativement faible du contour impacte peu ces longueurs d'ondes (contrairement aux valeurs associées à des longueurs d'onde faibles). Par ailleurs, le redimensionnement systématique de l'image que l'on applique permet de gommer partiellement le bruit.

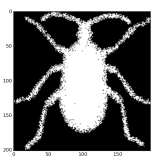
Afin de tester la robustesse au bruit, nous avons implémenté un modèle de bruit de Kanungo (pour un facteur de bruit $0 \leq a \leq 1$, tout point x du domaine à distance d du bord est colorié en noir avec probabilité a^d). La table 3 démontre le faible impact du bruit sur les valeurs propres.



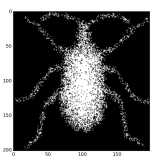
Bruit 0



Bruit 0.3



Bruit 0.6



Bruit 0.9

	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8
Bruit 0	83.93	153.41	258.76	263.10	353.64	378.19	495.73	500.11
Bruit 0.3	84.20	154.12	259.27	264.12	354.84	379.19	498.19	504.15
Bruit 0.6	84.42	155.22	262.05	264.96	355.90	388.19	503.35	514.37
Bruit 0.9	87.02	157.19	264.77	276.22	363.54	388.21	506.28	522.61

TABLE 3 – Valeurs propres associées à `beetle-13.pgm` en fonctions du facteur de bruit

3.2 Performances

Nous avons testé notre algorithme avec 4 métriques différentes pour mesurer la distance entre descripteurs :

— Distance euclidienne : $d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$.
— Distance de Minkowski (facteur 10) : $d(x,y) = \left(\sum_{i=1}^{n}(x_i - y_i)^{10}\right)^{1/10}$.
— Distance euclidienne carrée (sqeuclidean) : $d(x,y) = \sum_{i=1}^{n}(x_i - y_i)^2$.
— Distance cosine : $d(x,y) = 1 - \frac{\sum_{i=1}^{n}(x_i - y_i)^2}{\|x\|_2 \|y\|_2}$.

Nous avons utilisé des descripteurs de type 1 ($\lambda_1/\lambda_2, \lambda_1/\lambda_3, \lambda_1/\lambda_4, \ldots$), de type 2 ($\lambda_1/\lambda_2, \lambda_2/\lambda_3, \lambda_3/\lambda_4, \ldots$), et un mélange des 2 types.

Etant donné un descripteur (de taille fixée) et une distance, nous avons séparé les 1050 images en deux catégories : un *train set* (composé d'environ 80% des images) et un *test set* (constitué des images restantes). Pour chaque image du *test set* nous avons recherché l'image du *train set* qui minimise la distance utilisée. L'algorithme de classification réussit si les deux images appartiennent à la même catégorie.

Nous avons répété 20 fois l'expérience précédente (parallélisée sur 4 coeurs) afin d'obtenir des écarts types inférieurs à 0.03. Les pourcentages de réussites sont regroupés figure 1 pour le descripteur 1 et figure 2 pour le descripteur 2. La figure 3 illustre les performances lorsque le descripteur est composé en partie du type 1 et du type 2 (par exemple l'abscisse 30/10 correspond à un descripteur de taille 40 dont les 30 premières composantes sont de type 1 et les 10 dernières de type 2). Nous avons utilisé uniquement la métrique cosine pour ce dernier test.

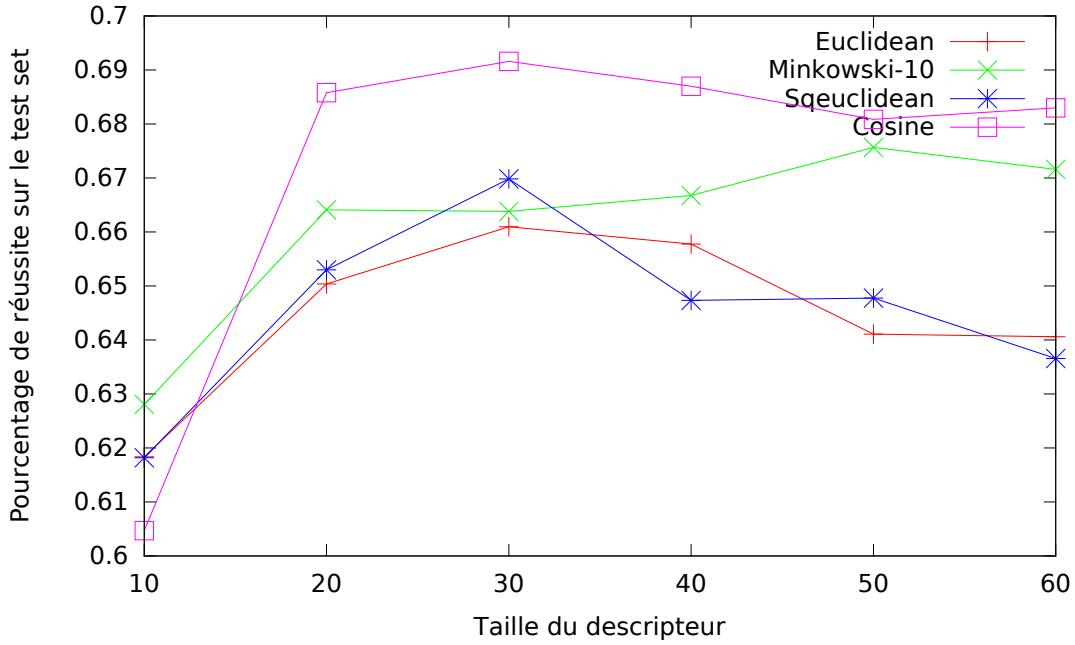


FIGURE 1 – Performances du descripteur 1 selon la distance utilisée

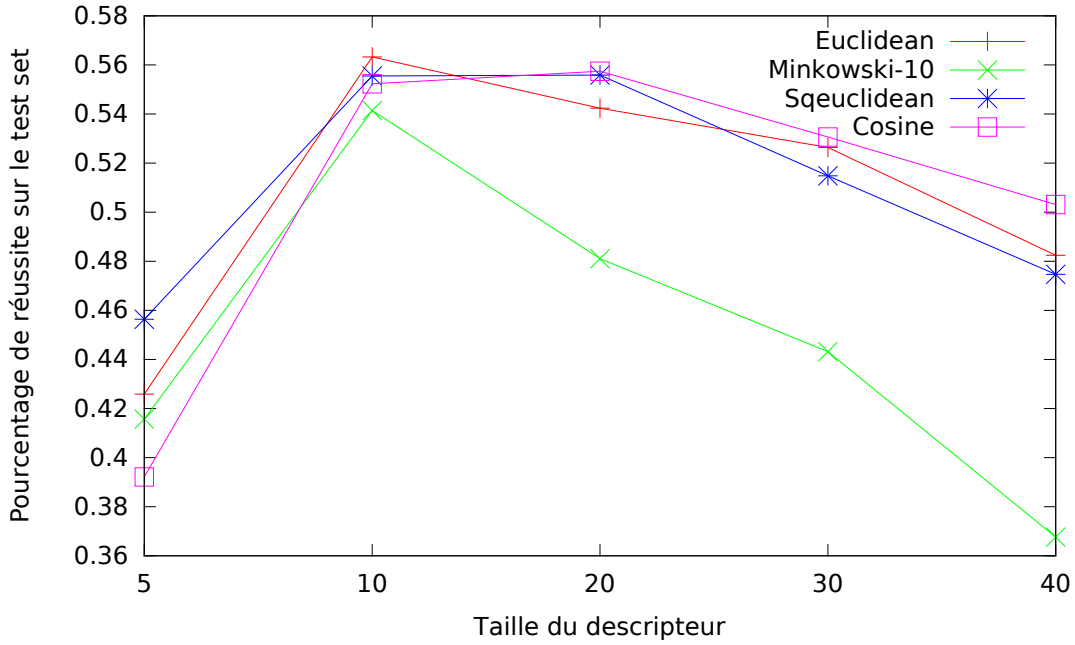


FIGURE 2 – Performances du descripteur 2 selon la distance utilisée

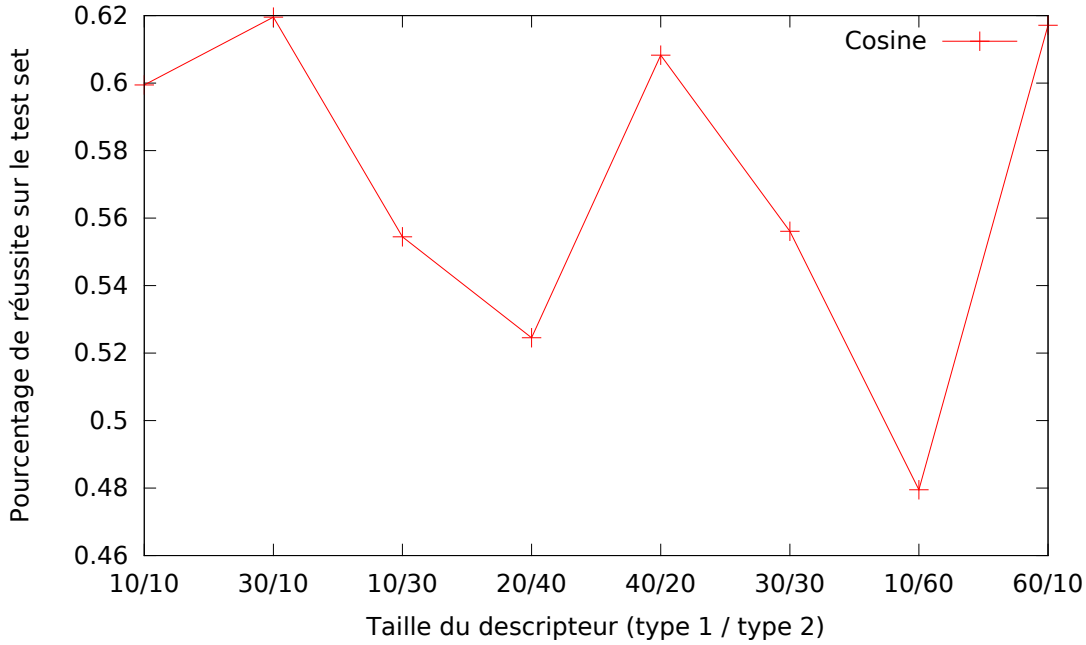


FIGURE 3 – Performances des descripteurs 1 et 2 mélangés, selon la distance utilisée

4 Discussion

Le descripteur de type 1 ($\lambda_1/\lambda_2, \lambda_1/\lambda_3, \lambda_1/\lambda_4, \ldots$) se révèle être le plus efficace. En effet, comme on peut le constater figure 1, quelle que soit la métrique utilisée il permet de dépasser les 60% de réussite, alors que le descripteur de type 2 (figure 2) ne franchit pas 56%. Mélanger les deux descripteurs ne permet pas non plus d'augmenter les performances comme l'illustre la figure 3.

Le meilleur score est obtenu avec la métrique cosine pour un descripteur de type 1 de taille 30 (environ 70% de réussite). Nous avons donc utilisé ces caractéristiques dans notre outil final. La similarité entre deux images M_1, M_2 (`similarity.sh`) est définie comme étant $1 - \frac{cosine(M_1,M_2)}{2}$ (le facteur 2 permet de normaliser à 1 : deux images proches ont une similarité proche de 1).

Afin de classifier une image M parmis les 70 classes (`classify.sh`) nous calculons pour chacune des classes l'inverse de la distance minimale de M à une image de la classe. Les 70 valeurs obtenues sont ensuite normalisées pour obtenir un vecteur unitaire V de taille 70 tel que $V[i]$ est proche de 1 lorsque l'image d'entrée est similaire à l'objet de la classe i.

Par ailleurs, la faible sensibilité aux perturbations (rotation, dilatation, bruit...) confirme notre hypothèse de départ : lorsque le domaine de l'image est faiblement changé, la propagation des ondes est peu impactée.

Afin d'améliorer la classification, plusieurs paramètres de l'algorithmes pourraient être étudiés plus en détail. Ainsi, on constate que le choix de la métrique est important (pour un descripteur fixé, les performances peuvent changer de 20% d'une métrique à l'autre). Une approche par machine learning pourrait permettre par exemple de pondérer chaque composante du descripteur (peut-être faut-il donner d'avantage d'importance à λ_1/λ_2 qu'à λ_1/λ_3).

5 Conclusion

Reconnaître la forme d'un objet à partir du son qu'il émet est un problème mathématique ancien (voir [?]). Bien qu'il existe des objets différents émettant des sons identiques (la figure 4 en est un exemple), dans la majorité des cas le bruit émis est une bonne caractéristique de l'objet.

Comme on a pu le constater dans ce projet, utiliser la propagation d'une onde (via les valeurs propres du Laplacien de Dirichlet) comme seul descripteur permet d'obtenir un outil de classification atteignant 70% de réussite. Un des atoûts principaux de cette méthode est sa résistance aux perturbations (rotation, bruit...). Par ailleurs, de nombreuses améliorations semblent possibles pour améliorer les performances (choix d'une meilleur métrique, enrichissement du descripteur avec d'autres caractéristiques de l'image, ...).

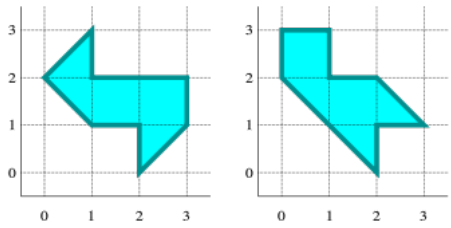


FIGURE 4 – Deux images renvoyant des sons identiques (source : Wikipédia)

Bonus

Nous avons essayé de reconstruire un son à partir des valeurs propres du Laplacien de Dirichlet. Pour cela, à partir du spectre des valeurs propres $\{\lambda_1, \lambda_2, ...\}$ de l'image, on en déduit les fréquences pouvant se propager dans la cavité définie par l'image, de la forme $\{\alpha\sqrt{\lambda_1}, \alpha\sqrt{\lambda_2}, ...\}$, α étant une constante choisie de façon à ce que le spectre obtenu soit compris entre 100Hz et 2kHz.

En pratique, $\alpha = 40$ et on ne retient que les trois premières valeurs propres. On associe la même puissance à chacune des 3 fréquences calculées. On peut changer ces différents paramètres pour entendre des sons différents.

Afin d'entendre le son associé à l'image `beetle-11.pgm` par exemple, entrer :

```
./sound.sh database/beetle-11.pgm
```

Nous avons également développé un petit jeu qui consiste à retrouver parmi les sons de plusieurs objets celui appartenant à la même catégorie qu'un motif de départ. Il est accessible par :

```
./sound_game.sh
```

Références

[Kac66] Mark Kac. Can one hear the shape of a drum ? *American Mathematical Monthly*, pages 1–23, 1966.

[KHR07] Mohamed A. Khabou, Lotfi Hermi, and Mohamed Ben Hadj Rhouma. Shape recognition using eigenvalues of the dirichlet laplacian. *Pattern Recognition*, 40(1) :141–153, 2007.

[ZKBM04] M. Zuliani, C. S. Kenney, S. Bhagavathy, and B. S. Manjunath. Drums and curve descriptors. In *British Machine Vision Conference*, Sep 2004.