

Reconnaissance et indexation de formes

Quentin Cormier

Yassine Hamoudi

4 mai 2015

Table des matières

1	Introduction	1
2	Méthode	1
2.1	Descripteur associé à l'image	2
2.2	Normalisation des images	3
3	Résultats	3
3.1	Sensibilité aux perturbations	4
3.2	Performances	5
4	Discussion	7
5	Conclusion	7

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 classier 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. La résistance de notre algorithme à différentes perturbations (rotation, bruit...) sera également vérifiée théoriquement et expérimentalement.

2 Méthode

L'objectif est de classier un certain nombre d'images noir et blanc en une catégorie parmi 70 possibles. On dispose pour cela d'une dizaine d'images "d'entraînement" par catégories (le *training set*, noté \mathcal{T}).

A partir de ces images d'entraînement on doit pouvoir classier une image inconnue dans l'une de ces 70 catégories.

Au vu de la taille relativement restreinte du *training set*, on décide d'encoder l'image en un vecteur de réels (le *descripteur* associé à l'image), et d'utiliser une distance entre deux descripteurs pour classier une image inconnue.

Ainsi, pour une image inconnue M , on calcule le descripteur associé d_M , puis on cherche l'image la plus proche dans le *training set* :

$$M' = \operatorname{argmin}_{M' \in \mathcal{T}} d(d_M, d_{M'}),$$

d étant la distance entre descripteurs choisie.

On prédit alors la classe de M comme étant celle de M'

2.1 Descripteur associé à l'image

La partie principale du projet consiste donc à calculer un descripteur associé à chaque image. Ce vecteur de réels doit dépendre de la forme de l'image. Idéalement, deux images ayant des formes similaires doivent donner deux descripteurs similaires.

Dans la mesure du possible, il doit être également invariant par translation, par rotation et par dilatation : traduire une image, la tourner, l'agrandir ou la rétrécir ne change pas la catégorie de l'image, et donc ne devrait pas changer le descripteur. On aimerait également une robustesse au bruit : un léger bruit ne doit modifier que très légèrement le descripteur.

On choisit ici d'utiliser un descripteur inspiré par la physique qui respecte ces différentes contraintes.

Onde associée à un domaine

En physique, il est bien connu qu'une onde stationnaire se propageant dans une cavité ne peut se propager seulement en certaines fréquences discrètes qui dépendent de la forme de la cavité. On se propose de construire un descripteur basé sur ces fréquences. On comprends bien qu'il sera facile de garantir l'invariance par rotation et translation : tourner ou bouger la cavité ne change pas les fréquences qui peuvent se propager dans ladite cavité !

Une onde dans une cavité vérifie l'équation suivante :

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

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

Si l'on recherche les solutions stationnaires, on pose $E = f(x, y)e^{i\omega t}$, et on obtient :

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

Ainsi, on peut voir f comme un vecteur propre de l'opérateur Laplacien et $\frac{\omega^2}{v^2}$ la valeur propre associée.

Calcul du descripteur

Calcul des valeurs propres du Laplacien d'une image

Pour calculer effectivement 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 alors la matrice de l'opérateur Laplacien discrétisé Δ^* suivant, 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

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

Puis on multiplie chaque ligne $i + Nj$, $i \in \{1, \dots, n\}, j \in \{1, \dots, n\}$ 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 :

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

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

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

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

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

On est donc ramené à calculer les valeurs propres d'une matrice creuse. On utilise la fonction `numpy.linalg.eigvals` pour trouver effectivement les valeurs propres recherchées.

Calcul du descripteur associé aux valeurs propres

A partir du spectre trié par ordre croissant $\{\lambda_0, \lambda_1, \dots\}$ calculé précédemment, on construit de manière similaire à [?] et [?] deux familles de descripteurs :

- $\{\frac{\lambda_0}{\lambda_1}, \frac{\lambda_0}{\lambda_2}, \dots\}$
- $\{\frac{\lambda_0}{\lambda_1}, \frac{\lambda_1}{\lambda_2}, \frac{\lambda_2}{\lambda_3}, \dots\}$

On discutera du choix du descripteur final à partir de tests numériques détaillés dans la seconde partie.

2.2 Normalisation des images

Etant donné une image en entrée, on commence par rogner les bords jusqu'à ce que chaque bord de l'image contienne au moins un pixel blanc. Ceci permet de réduire la taille de l'image en enlevant les bords inutiles. Puis dans un second temps, on redimensionne l'image de façon à ce que le maximum de la largeur et de la hauteur de l'image soit de 50 pixels. Cette opération permet de se ramener à des images de taille au plus $N = 50 * 50 = 2500$. En effet, on sera ensuite amené à calculer les valeurs propres d'une matrice de taille $N * N$, opération relativement coûteuse en temps de calcul. Enfin on rajoute un cadre noir autour de l'image : cette opération permet de fermer la cavité définie par le motif de l'image.

Exemple :

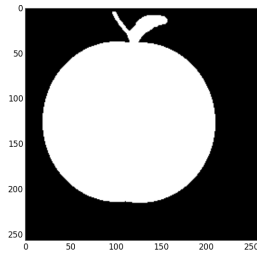
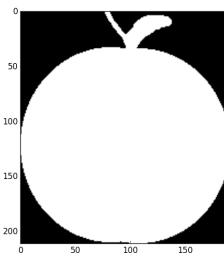


Image initiale



Bords rognés

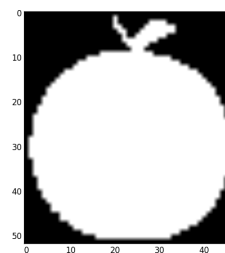


Image après redimensionnement
et ajout du cadre noir

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 dégradations 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 20s 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.

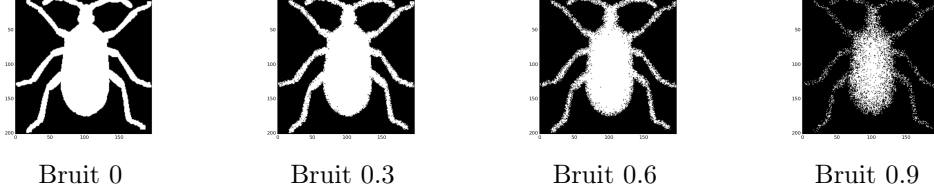


	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8
0 degré	118.57	177.16	252.35	285.53	361.22	404.89	420.69	489.36
30 degrés	120.28	182.75	257.54	297.97	370.39	413.53	426.99	472.08
120 degrés	120.29	182.88	266.09	308.13	372.66	414.83	428.60	472.21
220 degrés	120.42	182.95	250.94	299.81	366.01	406.55	424.32	478.39

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. Par conséquent, on peut s'attendre à ce qu'une déformation relativement faible du contour impacte peu ces valeurs (contrairement aux valeurs associées à des longueurs d'onde faibles). **Vrai ?** 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.



	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) : $d(x, y) = (\sum_{i=1}^n (x_i - y_i)^{10})^{1/10}$.
- Distance euclidienne carrée (sqeuclidean) : $d(x, y) = \sum_{i=1}^n (x_i - y_i)^2$.
- Distance cosinus : $d(x, y) = 1 - \frac{\sum_{i=1}^n (x_i - y_i)^2}{\|x\|_2 \|y\|_2}$.

Nous avons utilisé des descripteurs de type 1 ($\lambda_1/\lambda_2, \lambda_1/\lambda_3, \lambda_1/\lambda_4, \dots$), de type 2 ($\lambda_1/\lambda_2, \lambda_2/\lambda_3, \lambda_3/\lambda_4, \dots$), 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 cosinus pour ce dernier test.

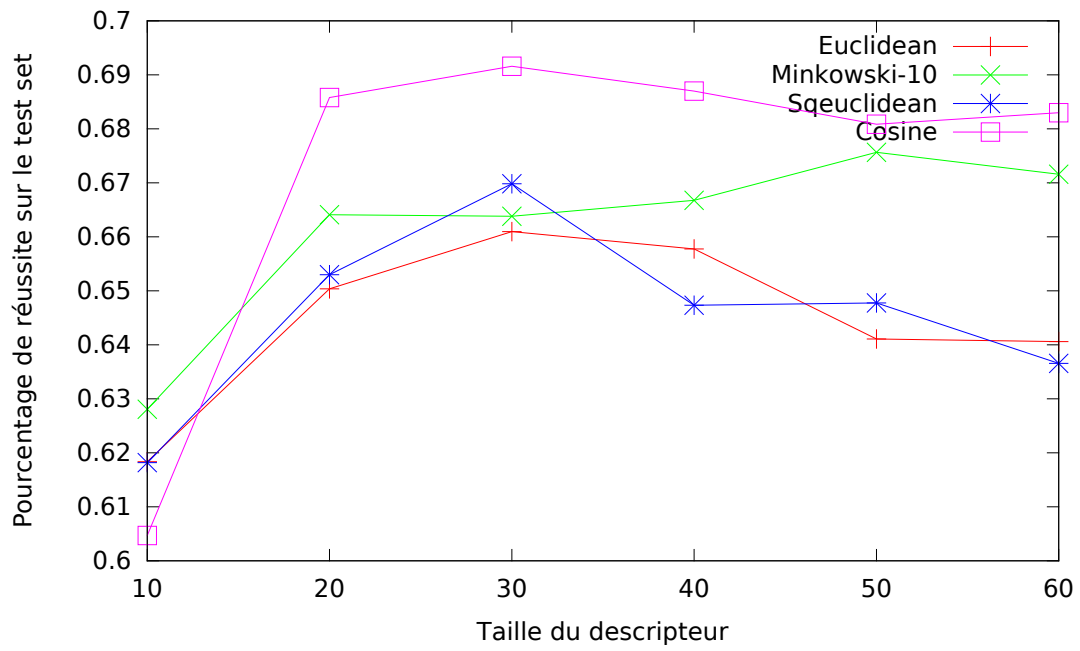


FIGURE 1 – Performances du descripteur 1 selon la distance utilisée

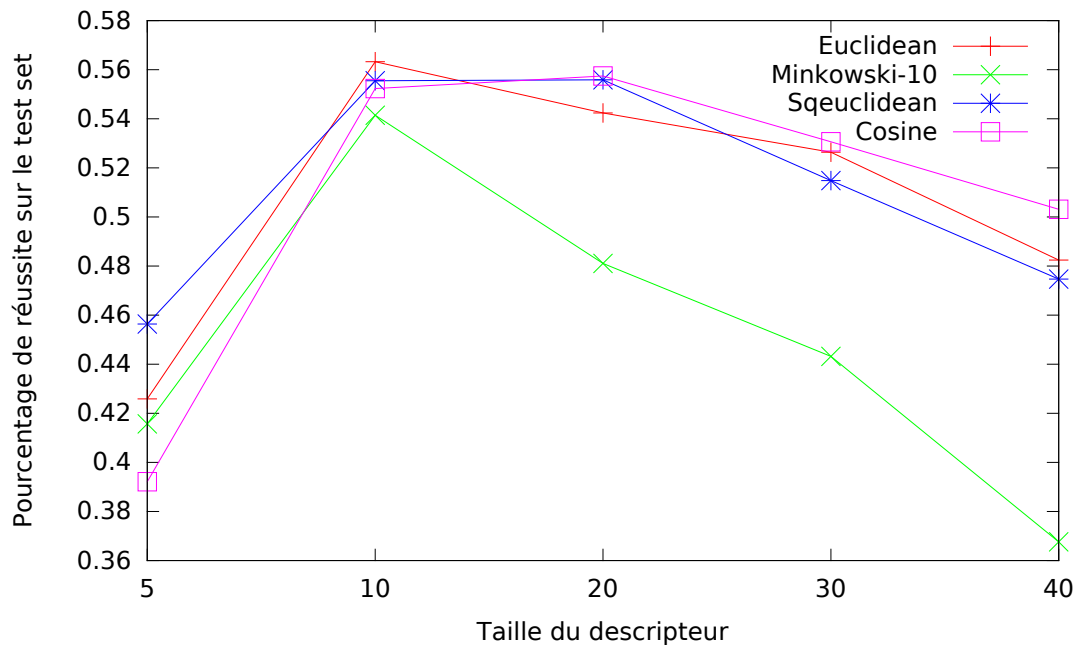


FIGURE 2 – Performances du descripteur 2 selon la distance utilisée

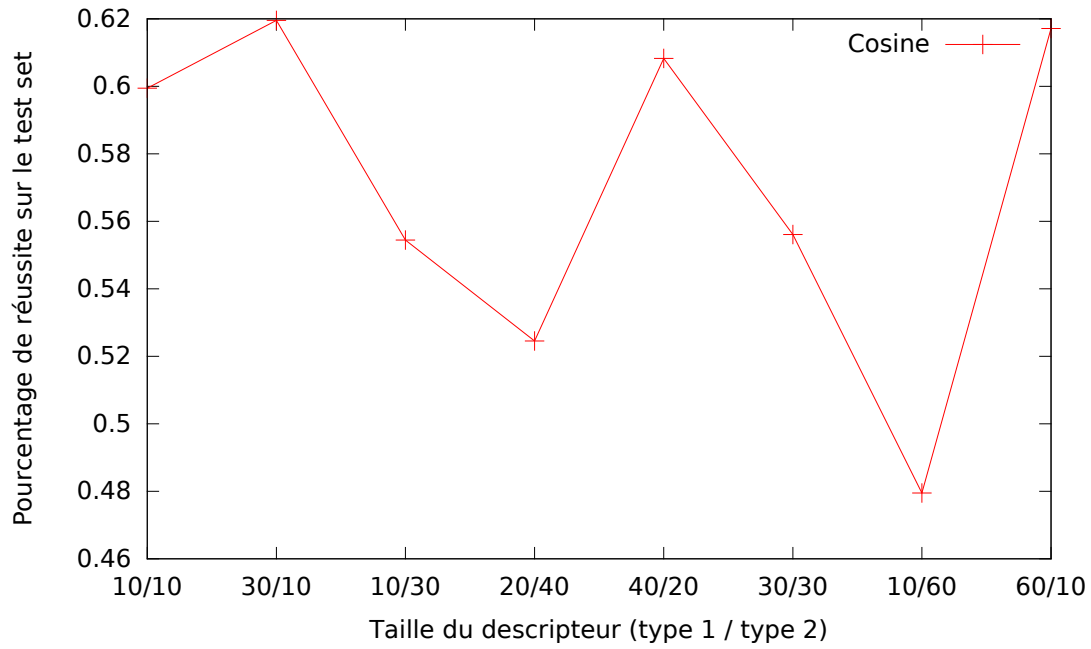


FIGURE 3 – Performances des descripteurs 1 et 2 mélangés, selon la distance utilisée

4 Discussion

Le descripteur de type 1 ($\lambda_1/\lambda_2, \lambda_1/\lambda_3, \lambda_1/\lambda_4, \dots$) 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 cosme pour un descripteur de type 1 de taille 30 (environ 70% de réussite). Nous utilisons ces caractéristiques dans notre algorithme final.

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'algorithme 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 atouts 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, ...).

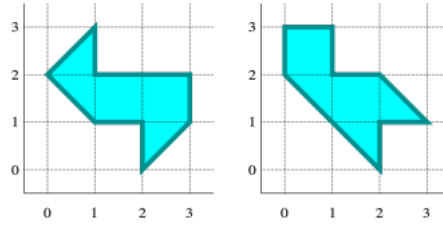


FIGURE 4 – Deux images renvoyant des sons identiques (source : Wikipédia)

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 $\{\lambda_1, \lambda_2, \dots\}$ de l'image, on en déduit les fréquences pouvant se propager dans la cavité définie par l'image, de la forme $\{\alpha\sqrt{\lambda_1}, \alpha\sqrt{\lambda_2}, \dots\}$, 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 3 valeurs propres. Enfin, on associe la même puissance à chacune des 3 fréquences calculées.

On peut s'amuser à 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
```