# TP 2 : codage d'une convolution

- Dans ce TP, nous implémenterons une opération simple utilisée en traitement d'images (entre autre) : la **convolution** ou 'filtrage linéaire'

- C'est une opération simple qui consiste à calculer, en chaque pixel, une nouvelle valeur obtenue comme une combinaison linéaire pondérée des pixels pris dans un certain voisinage (3x3, 5x5, etc.)

- Exemple pour un filtre 3x3

$$\hat{I}(x,y) = \sum_{i \in \{-1,0,1\}} \sum_{j \in \{-1,0,1\}} w_{ij} I(x+i, y+j)$$

# Filtre Gaussien

Le filtre Gaussien est un cas spécifique d'un filtre linéaire dont les poids sont donnés par le masque suivant :

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

* 1/16

Effet de lissage :

# Travail demandé

- Implémentez une version du filtre Gaussien qui utilise la mémoire globale pour stocker l'image (un pixel par thread). Les valeurs du filtre (pondérations) seront données en paramètre du noyau.

- Implémentez une deuxième version utilisant la mémoire partagée (chaque block copie dans la shared memory l'information dont il a besoin)

- Comparez les performances relatives de vos deux approches sur des images relativement grosses.

- Le filtre Gaussien est séparable, ce qui signifie qu'on peut obtenir le même résultat en convoluant 2 fois en 1D (convolution Gaussienne 1D en X puis en Y).

    - Expliquez pourquoi la convolution séparable est plus rapide théoriquement (par un calcul de complexité)

    - Implémentez une version séparable du filtrage Gaussien. Comparez les performances de vos différentes approches