

# Le morphage d'images



Steph Hoffman

GIF-4105/7105 Photographie Algorithmique, Hiver 2015  
Jean-François Lalonde

# Morphage

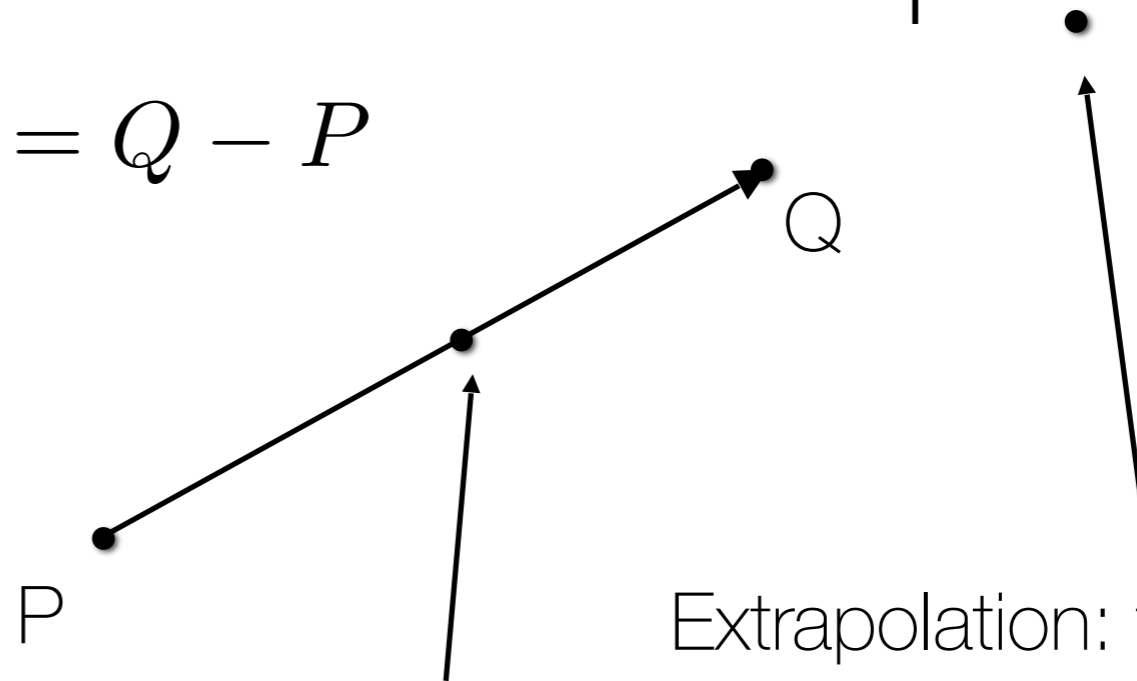


- Morphage: calculer une moyenne de deux images
  - Pas une moyenne des deux images
  - Mais une image de l'objet moyen!
- Comment générer une animation?
  - Calculer une moyenne pondérée

# Calculer la moyenne de deux points

$$v = Q - P$$

Quelle est la moyenne  
de P et Q?



Interpolation linéaire

$$(1 - t)P + tQ$$
$$0 \leq t \leq 1$$

$$P + 0.5v$$
$$= P + 0.5(Q - P)$$
$$= 0.5P + 0.5Q$$

Extrapolation:  $t < 0$  ou  $t > 1$

$$P + 1.5v$$
$$= P + 1.5(Q - P)$$
$$= -0.5P + 1.5Q$$

- P et Q peuvent être n'importe quoi!
  - points en 2D, 3D
  - Couleurs RGB (3D)
  - Images entières! (NxM D)

# Idée 1 : dissoudre (“cross-dissolve”)



- Interpole l'image entière
  - $image = (1-t)*image1 + t*image2$
- Pourquoi ça “marche”?

Démonstration



# Idée 2: alignement, puis dissoudre



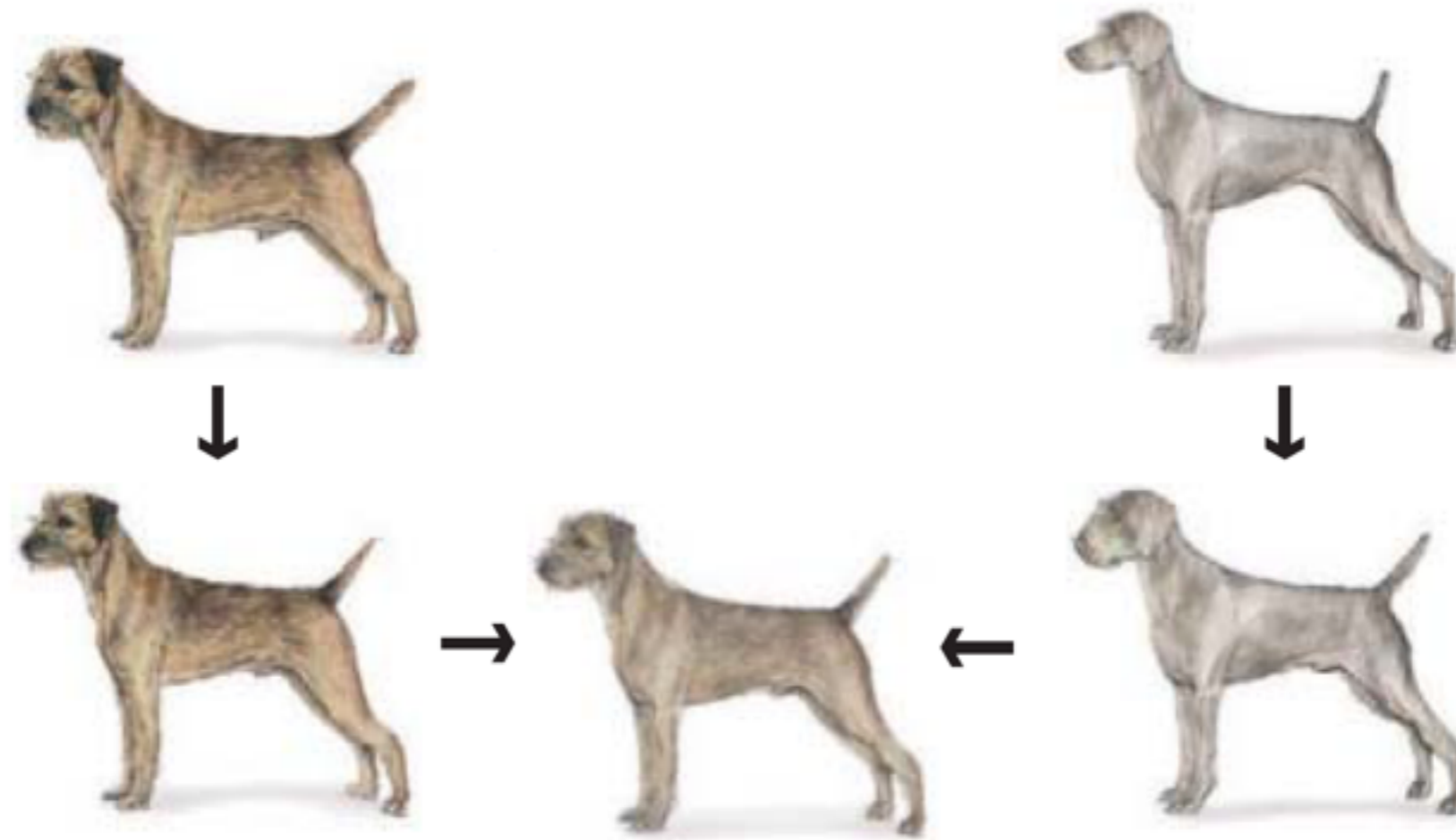
- On aligne d'abord, puis on dissout
  - Transformation affine (ou projective) globale
- Problème?

# Calculer le chien moyen



- Que faire?
  - Dissoudre ne fonctionne pas
  - Alignement global (e.g. affine) ne fonctionne pas!
  - Idées?
- Correspondances!
  - Queue vers queue, tête vers tête, pattes vers pattes, etc.
  - Il nous faut des transformées *locales*!

# Idée 3: alignement local, puis dissoudre



- Pour chaque trame  $t$
- Calculer la forme moyenne (i.e. le chien moyen!)
  - alignement local
- Calculer la couleur moyenne
  - dissoudre les correspondances



# Transformations locales (non-paramétriques)



- Transformations locales nécessitent beaucoup de paramètres
- Techniquement, on pourrait avoir une transformation  $(u,v)$  pour chaque pixel  $(x,y)$ !
- Lorsqu'on connaît  $(u,v)$  (champ vectoriel), on peut facilement calculer la couleur par interpolation

# Rappel

- Transformations globales



translation



rotation

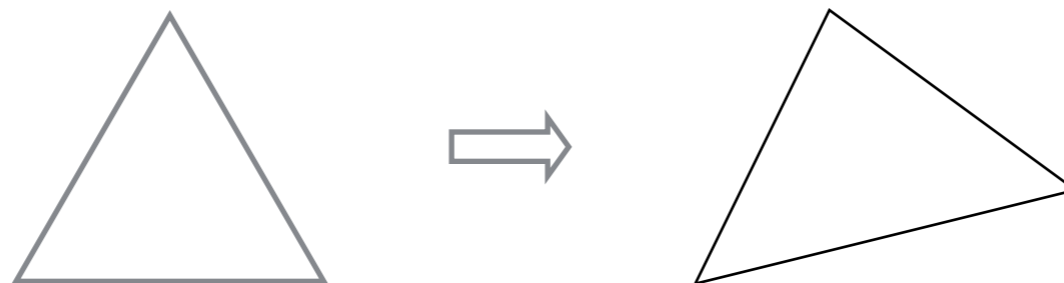


affine



perspective

- Appliquer une transformation
- Estimer une transformation



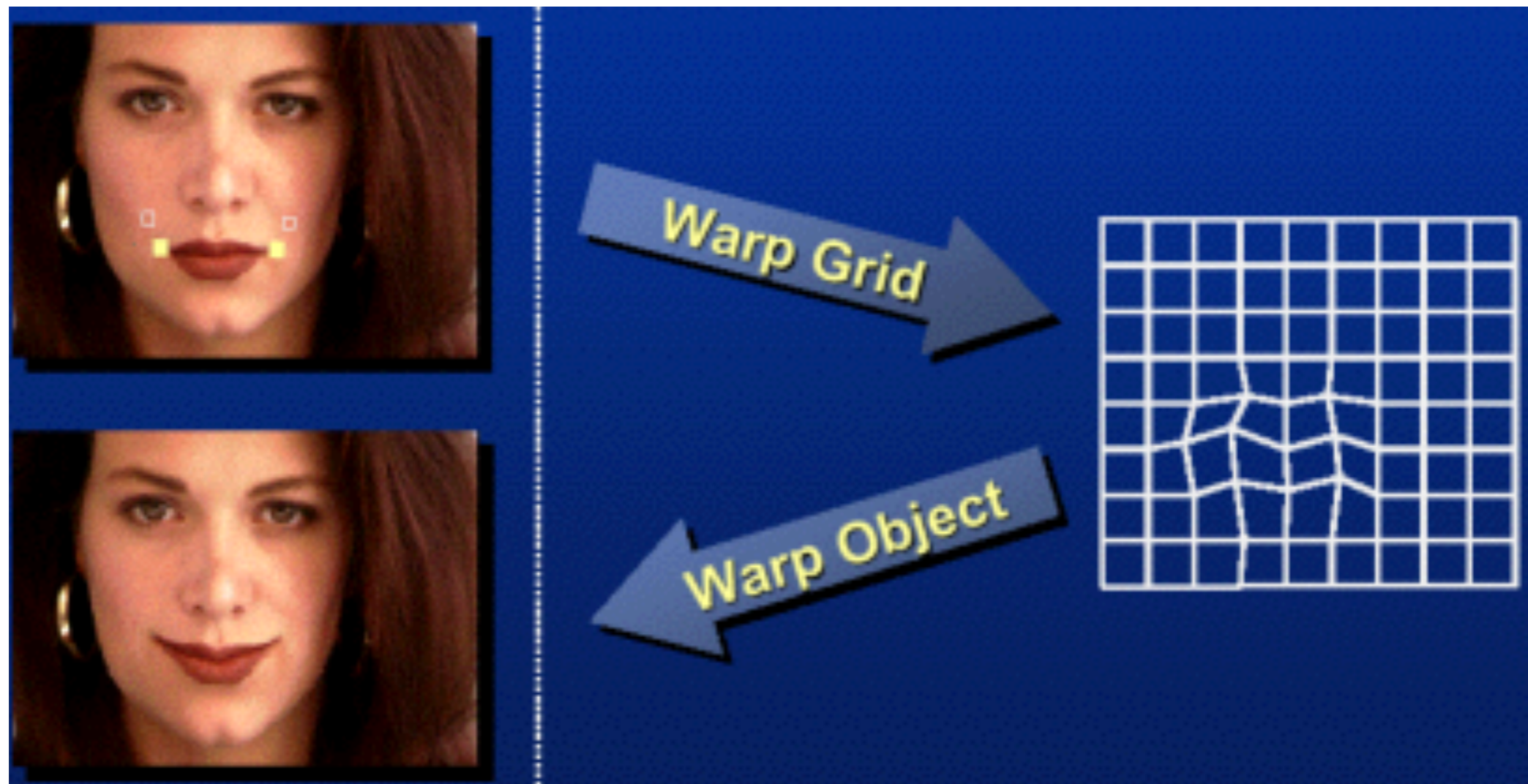
# Transformations locales (non-paramétriques)



- Transformations locales nécessitent beaucoup de paramètres
- Techniquement, on pourrait avoir une transformation  $(u,v)$  pour chaque pixel  $(x,y)$ !
- Lorsqu'on connaît  $(u,v)$  (champ vectoriel), on peut facilement calculer la couleur par interpolation

# Déformation non-paramétriques d'images

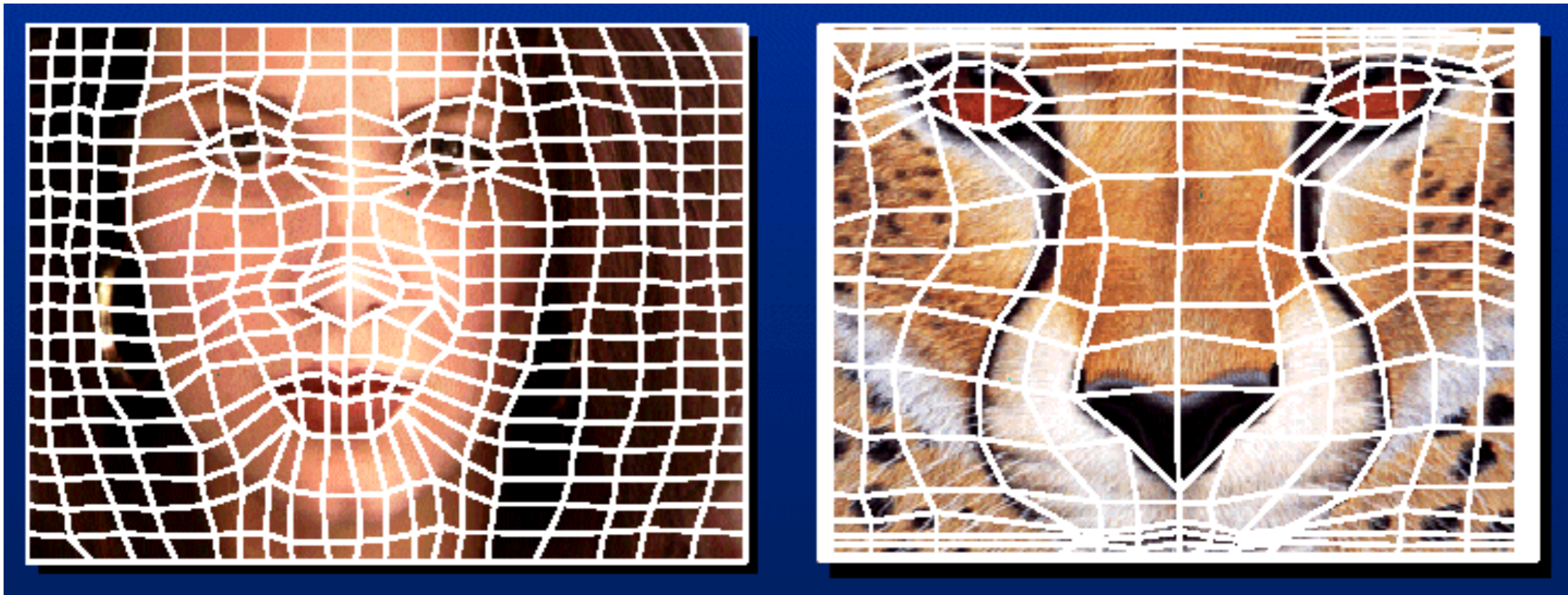
- Déplacer des points de contrôle vers la position désirée
- Calculer un champ vectoriel en interpolant
- Spécifie une déformation (vecteur) pour chaque pixel





# Spécifier la déformation (dense)

- Comment spécifier la déformation?
  - Déterminer les valeurs à des points de contrôle
    - et on interpole pour calculer la déformation à chaque pixel

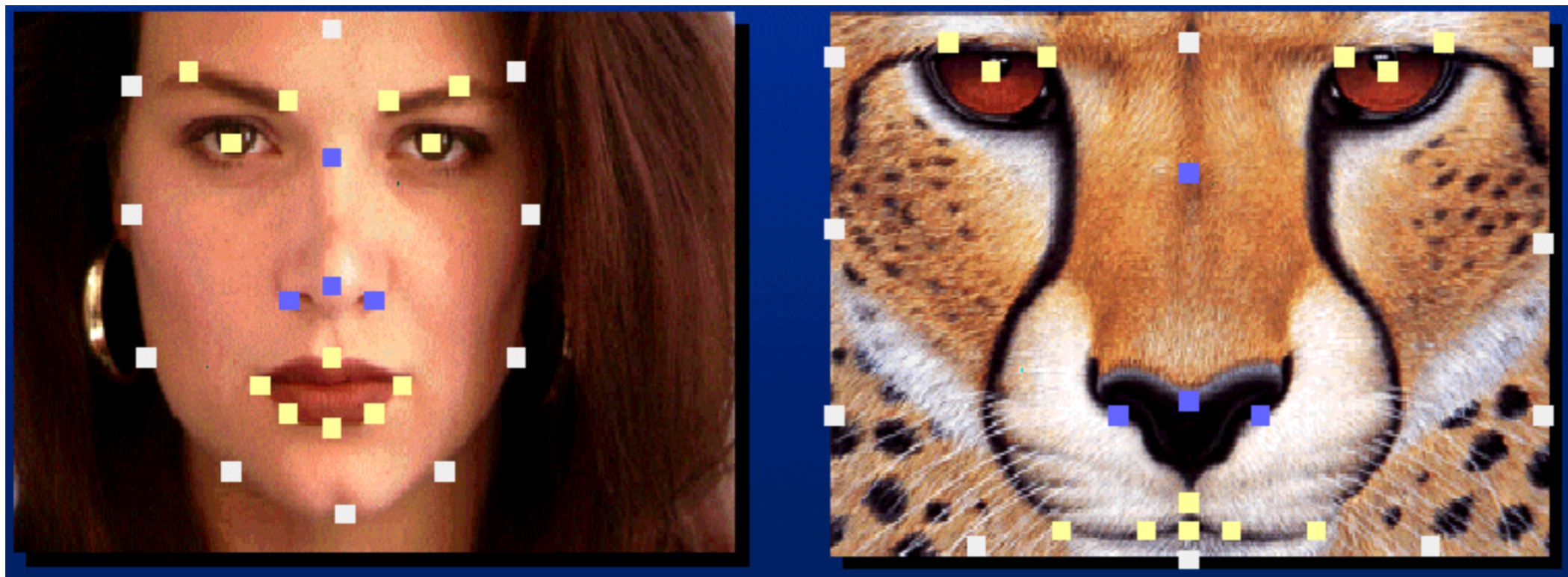


Ça en fait beaucoup à spécifier!

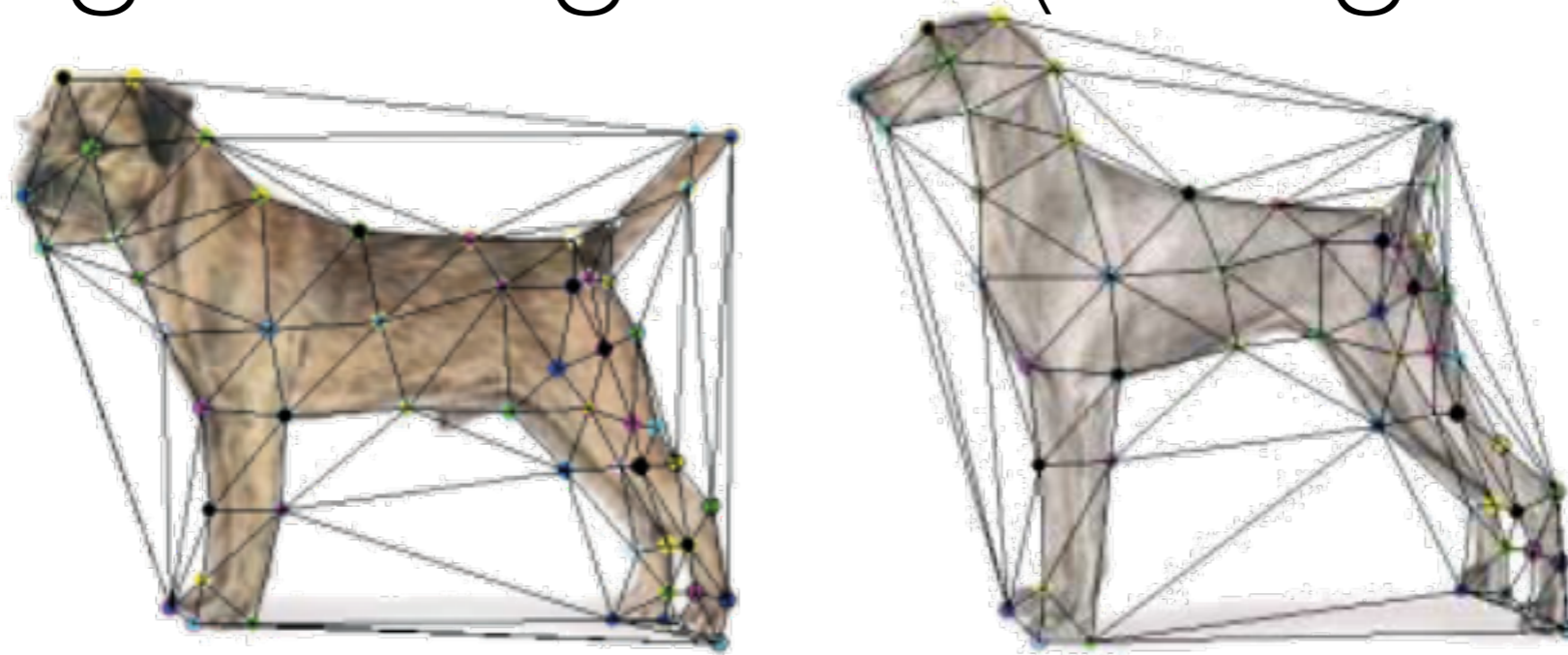


# Spécifier la déformation (par points)

- Comment spécifier la déformation?
  - Indiquer des points d'intérêts
    - et on interpole pour déterminer la valeur pour chacun des pixels



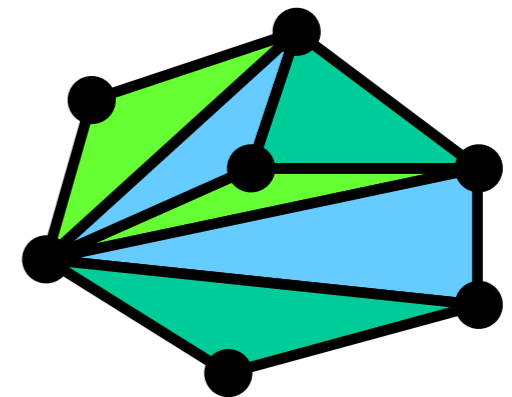
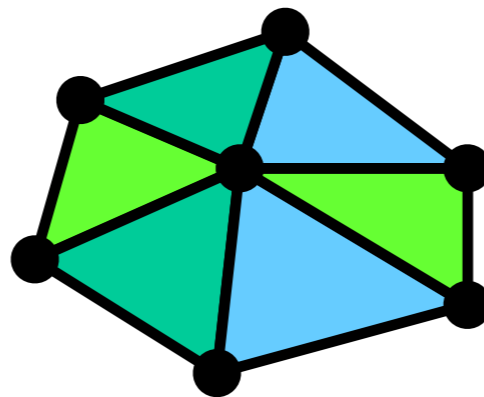
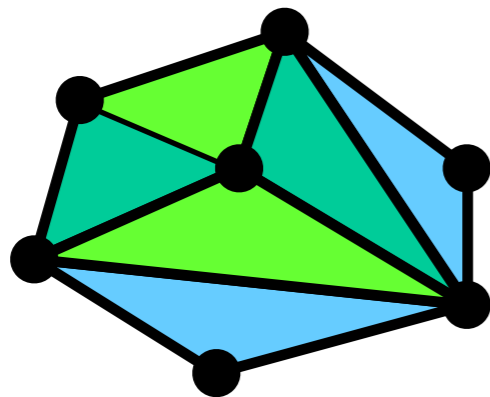
# Maillage triangulaire (triangulation)



- À partir des correspondances spécifiées manuellement
- Calculer un maillage (mesh) triangulaire
  - Le même pour les deux images. Qu'est-ce qu'on obtient?
    - Une correspondance des triangles dans les deux images
- Calculer la déformation pour chaque paire de triangle
  - Quelle type de transformation?

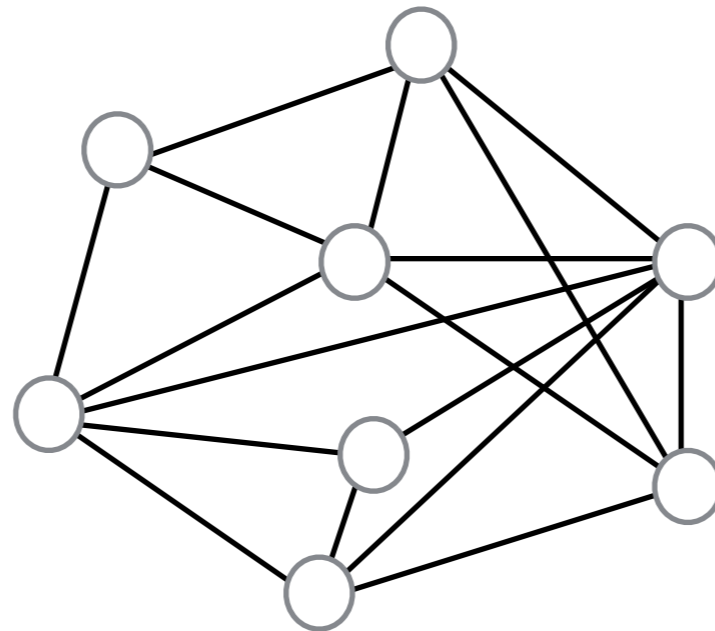
# Triangulations

- Une triangulation d'un groupe de points est une partition de son enveloppe convexe en un groupe de triangles dont les vertex sont les points, et qui ne contiennent aucun autre point.
- Combien de triangulations possibles?
  - Un nombre exponentiel!



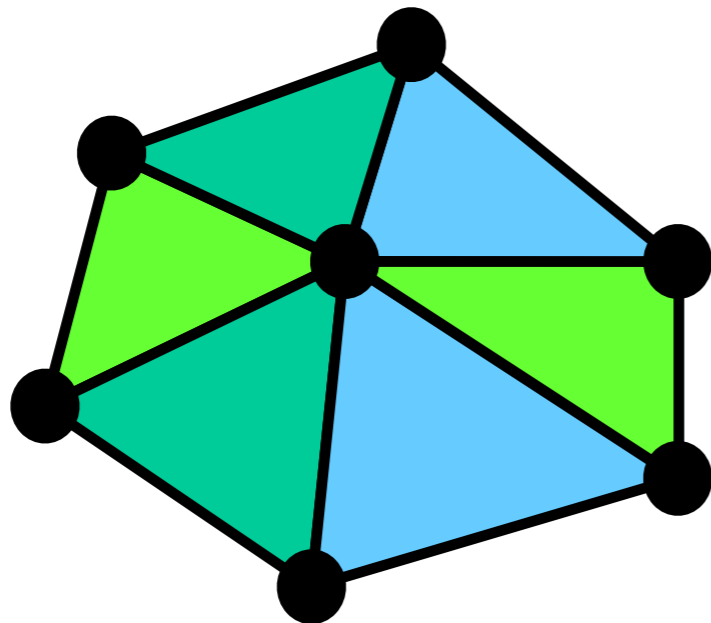
# Algorithme de triangulation (un peu naïf)

- Répéter:
  - Sélectionner deux points (qui n'ont pas d'arête entre eux)
  - Si l'arête entre ces deux points n'intersecte pas une autre arête, rajouter l'arête à la triangulation

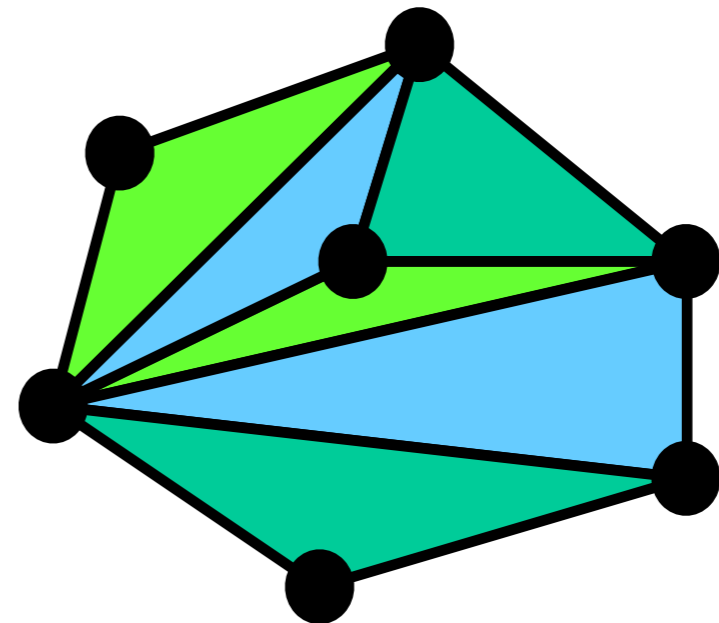


# Qualité des triangulations

- Calculons le plus petit angle d'une triangulation
  - Une triangulation T1 est "meilleure" que T2 si le plus petit angle de T1 est plus grand que celui de T2
  - La triangulation Delaunay est la "meilleure" (elle maximise le plus petit angle)



bonne

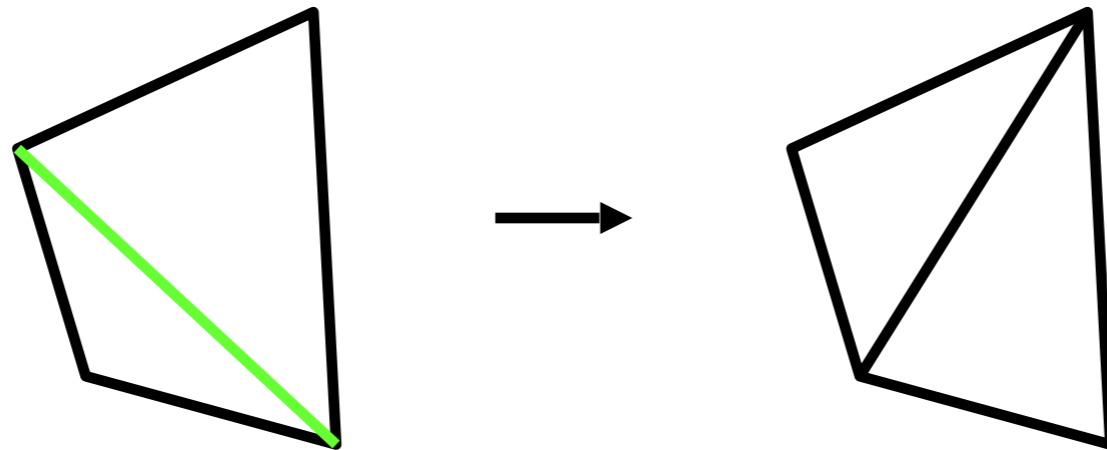


moins bonne



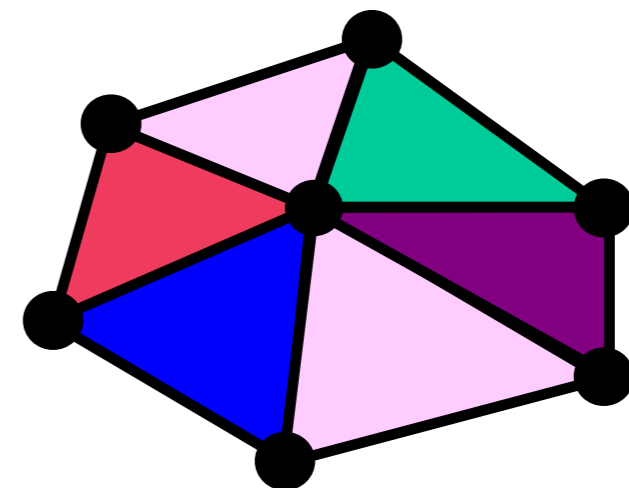
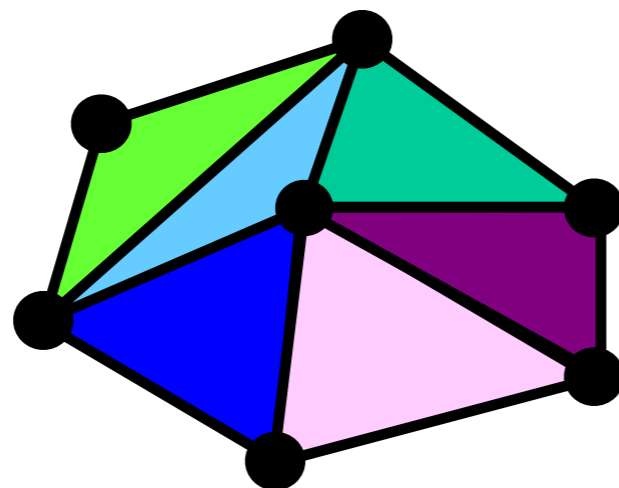
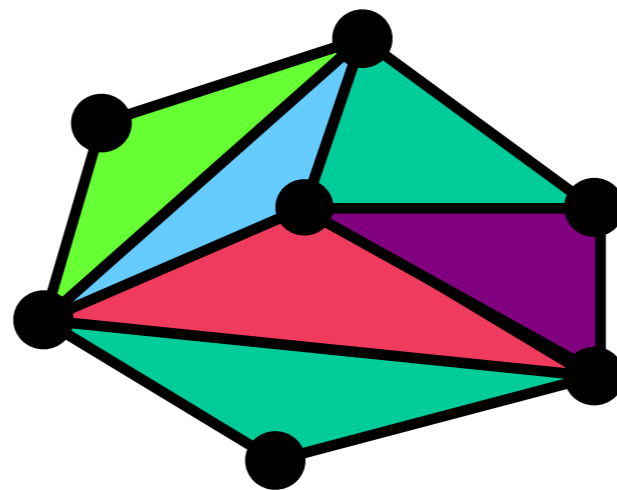
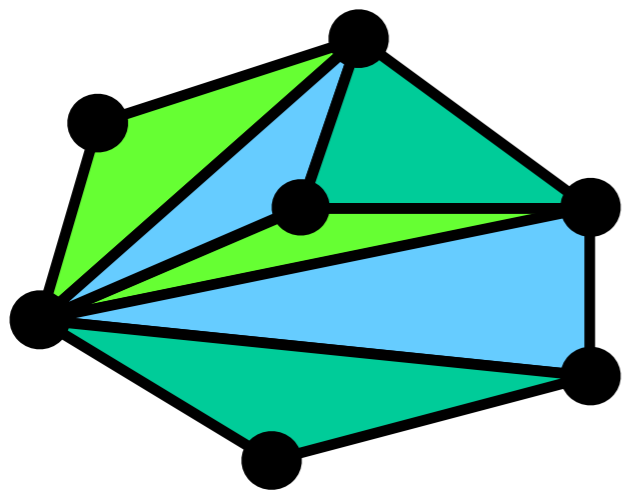
# Améliorer une triangulation

- Pour chaque quadrilatère convexe, nous pouvons inverser l'arête le traversant
- Si l'inversion améliore la triangulation localement
  - elle l'améliore aussi globalement
  - l'arête initiale est "illéegale"



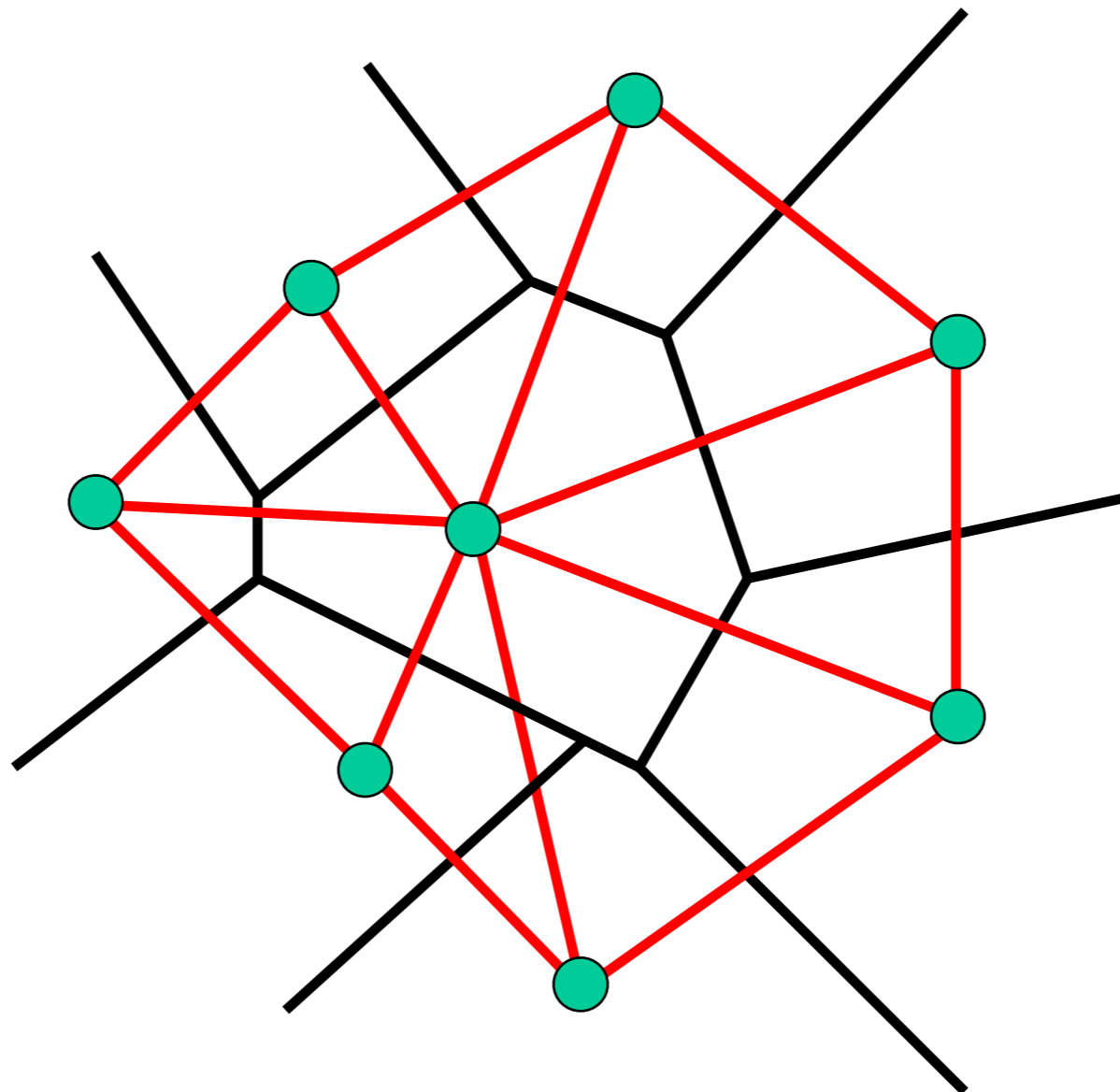
# Algorithme de Delaunay, version naïve

- Commencer avec triangulation arbitraire.
  - Tant qu'on trouve une arête illégale, l'inverser
- Pourrait prendre un bon bout de temps avant de converger!



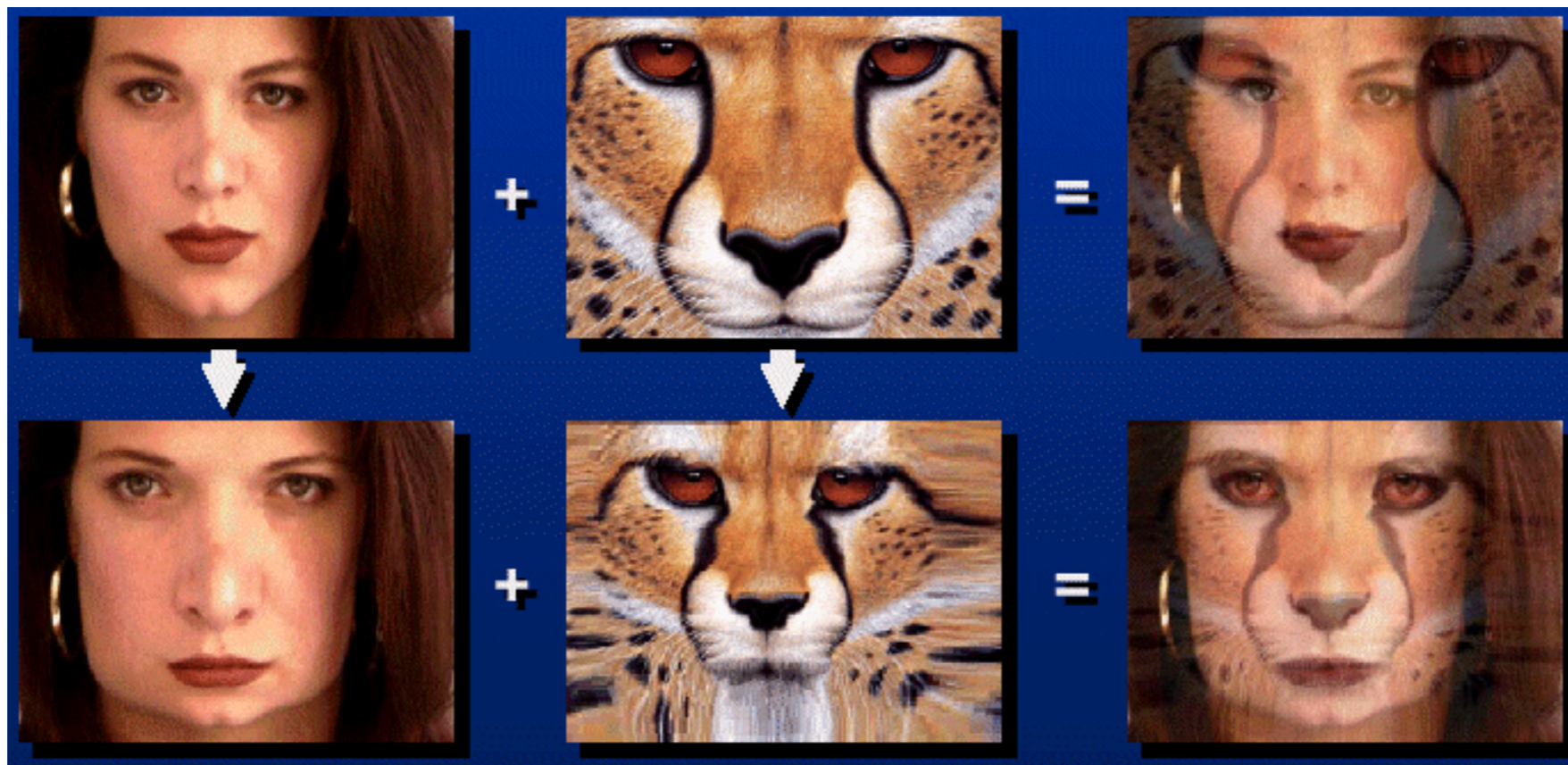
# Triangulation Delaunay par dualité

- Calculer le diagramme de Voronoi
- Graphe dual = triangulation Delaunay!
  - $O(n \log n)$



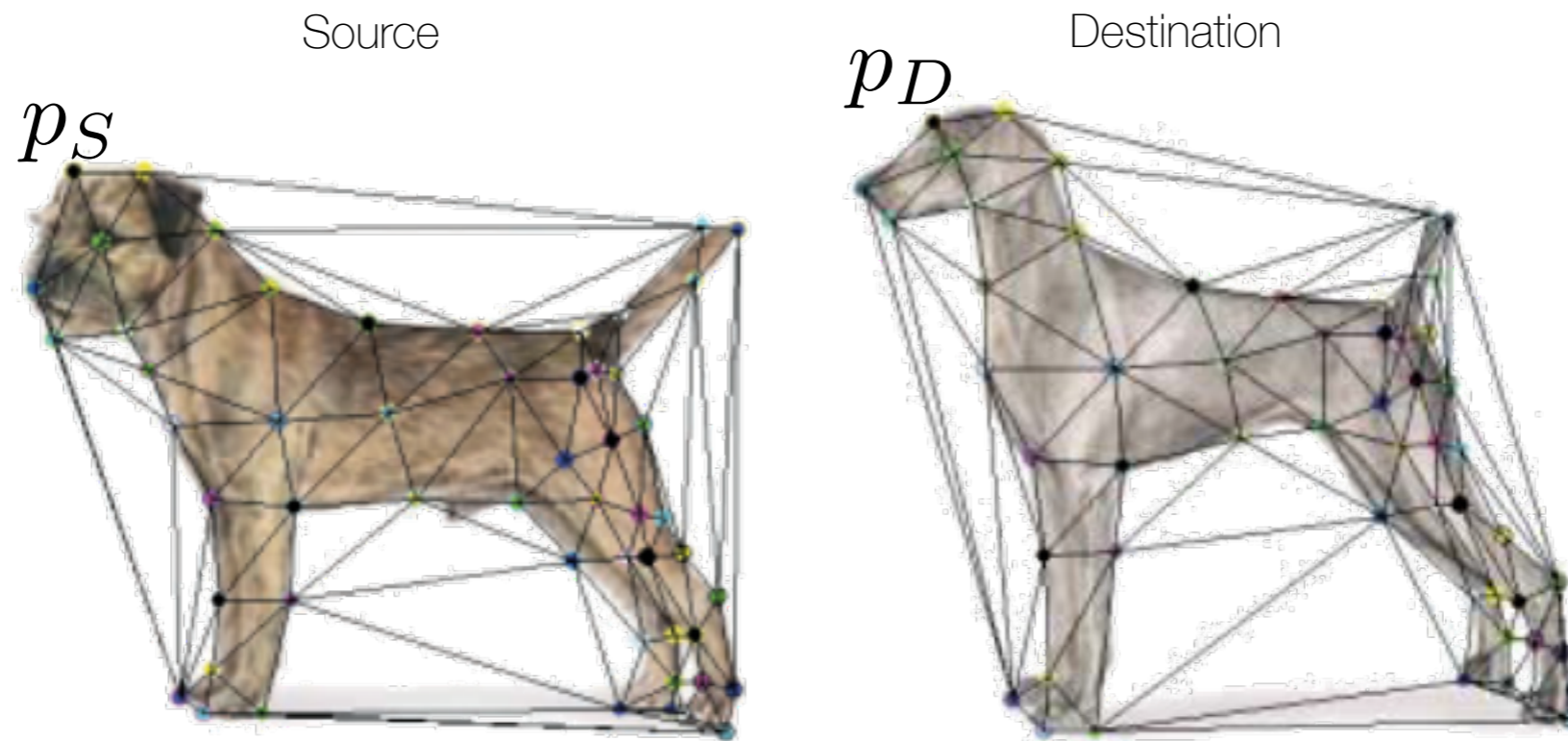
# Morphage d'images

- Créer une triangulation intermédiaire (interpolation)
- Calculer la déformation des deux images vers l'image intermédiaire
- Déformer les images
- Calculer la moyenne des deux images déformées



# Triangulation intermédiaire

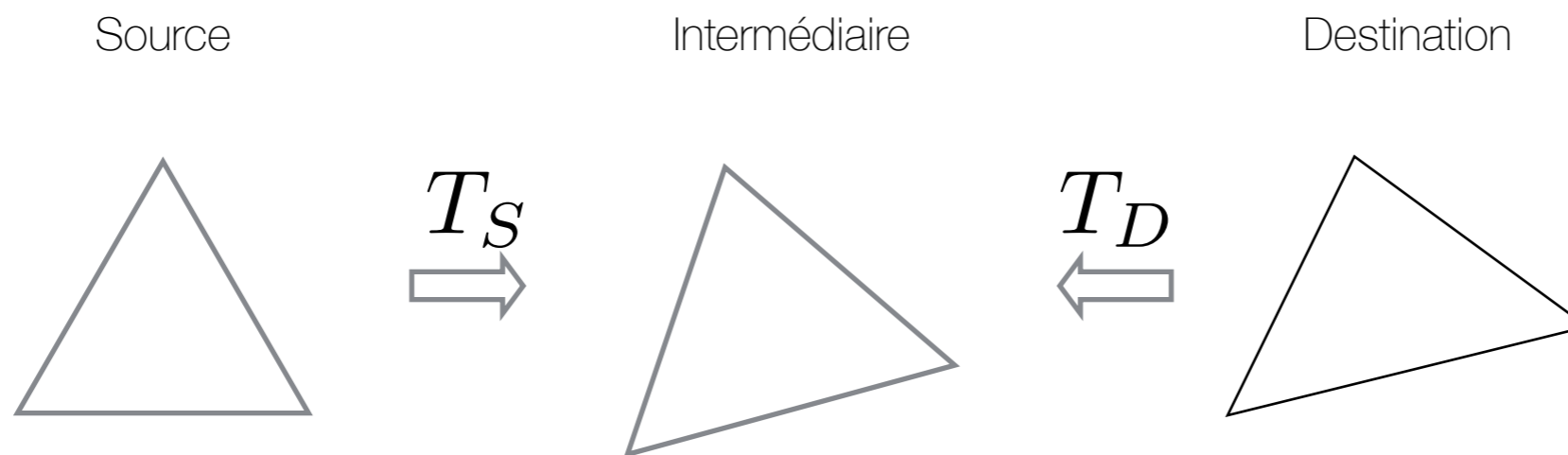
- Déterminer facteur  $\alpha = [0, 1]$
- Pour chaque point:
  - Calculer la moyenne pondérée  $(1 - \alpha)p_S + \alpha p_D$





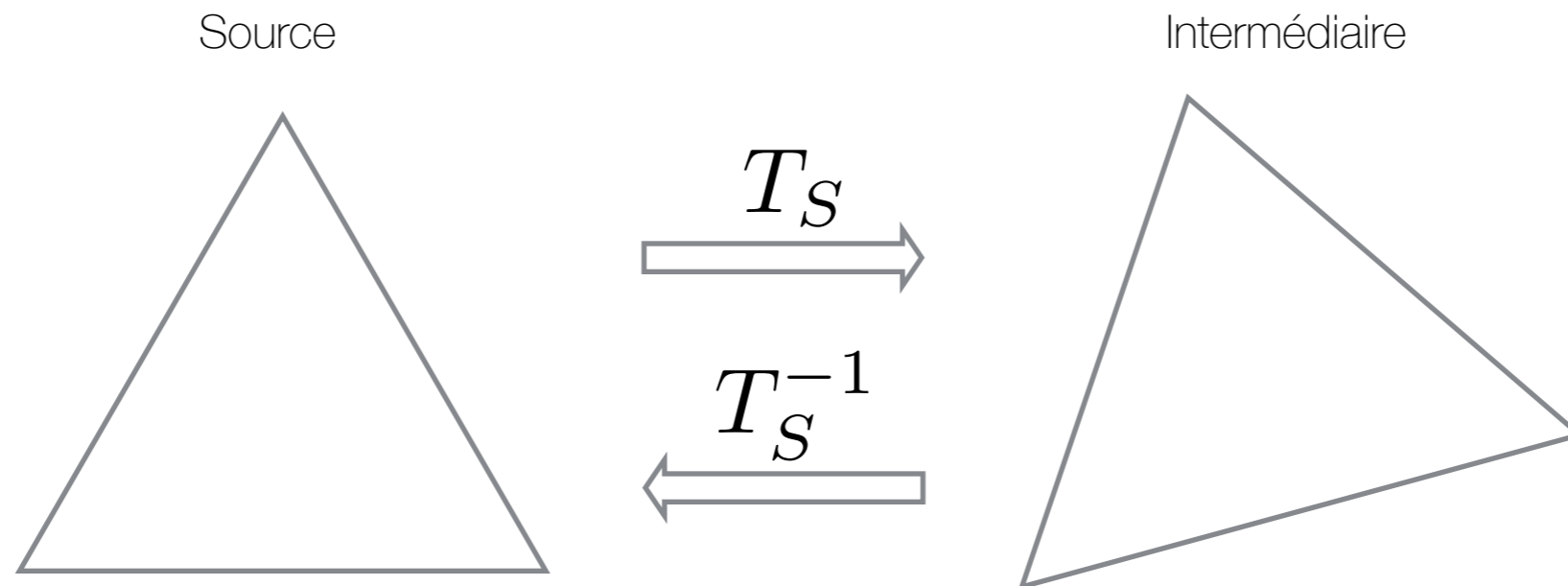
# Calculer la déformation

- Pour chaque triangle
  - Calculer la déformation à partir de la source et de la destination, selon les points correspondants
  - Quelle type de transformation?



# Déformer les images

- Pour chaque triangle
  - Appliquer la déformation
  - Directe ou inverse?



# Résumé

- Définir les points correspondants
- Calculer une triangulation
  - Utiliser la même pour les deux images!
- Pour chaque  $\alpha = 0, \dots, 1$ 
  - Calculer la triangulation intermédiaire (moyenne pondérée de chaque point)
  - Pour chaque triangle dans la triangulation intermédiaire
    - Calculer la transformation affine vers l'image source et destination
    - Pour chaque pixel dans le triangle, calculer leur provenance dans l'image source et destination, et calculer leur moyenne pondérée
  - Sauvegarder l'image intermédiaire
- Avec toutes les images, générer une animation!

# Morphage dynamique



Black or White (MJ):

<http://www.youtube.com/watch?v=R4kLKv5gtxc>

Willow: <http://www.youtube.com/watch?v=uLUyuWo3pG0>