Point Processing & Modeling

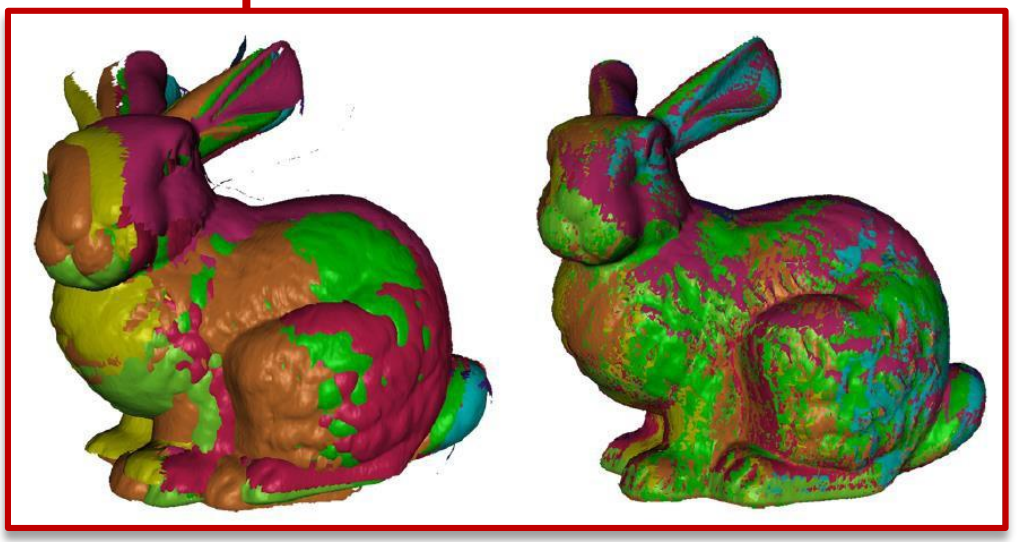
Processus d'acquisition de géométrie

Scan :
nuages de points

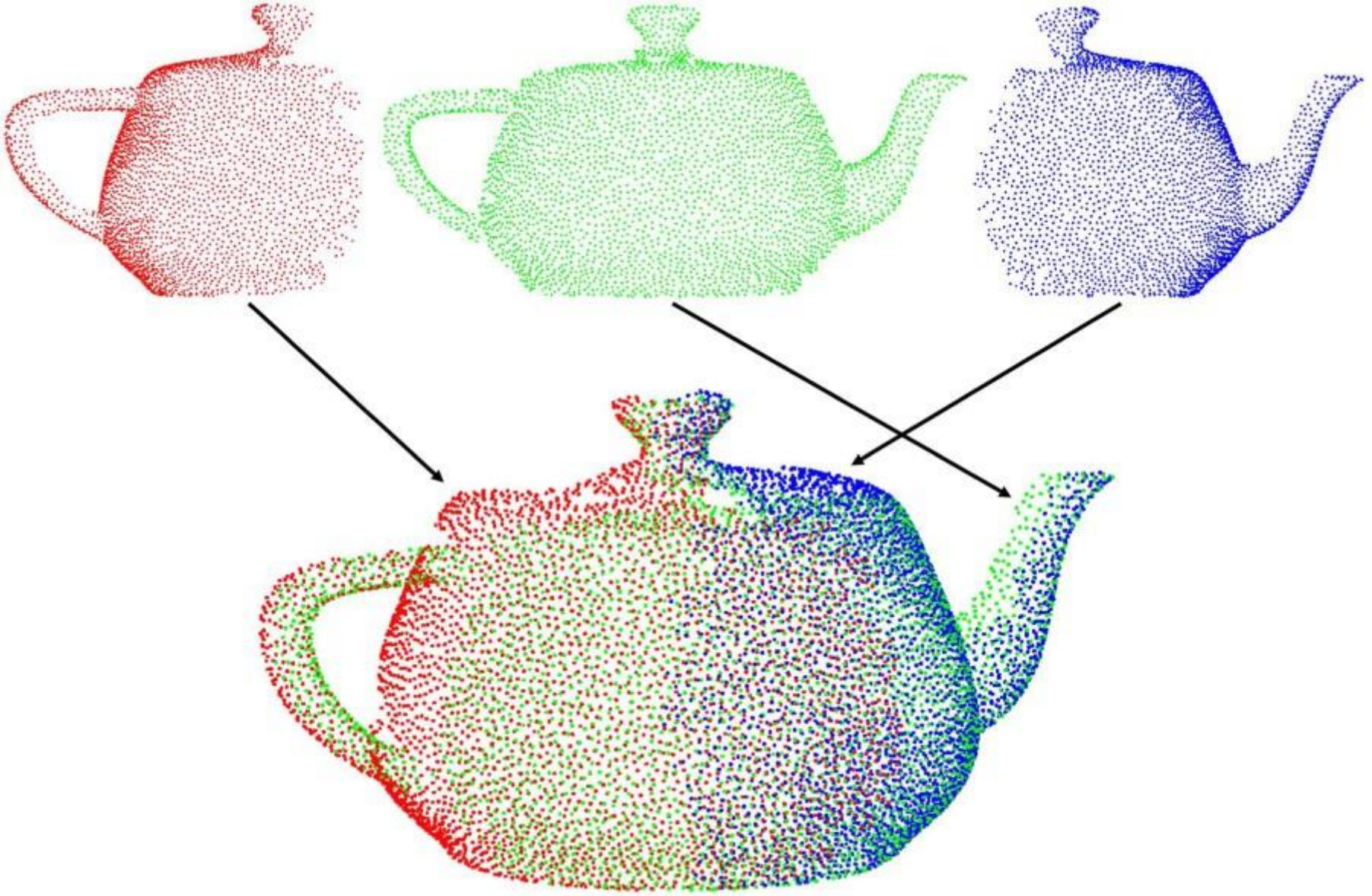
Recalage :
tous les scans dans le même système de coordonnées

Reconstruction :
Integration des scans dans un seul maillage

Postprocess:
- Filtrage géométrique et topologique filtering
- Remaillage…

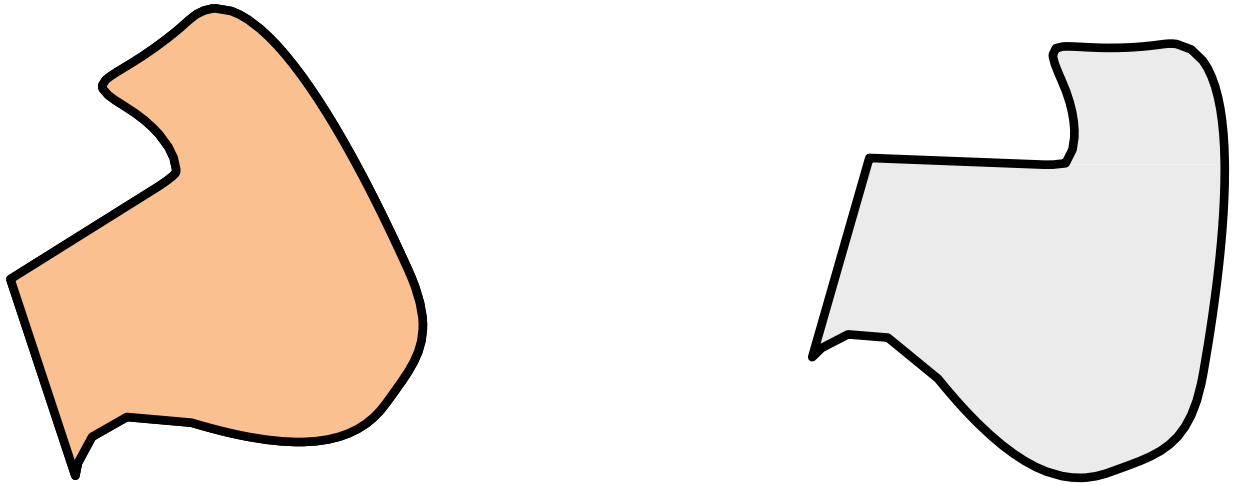


Pourquoi ?



Problème

M_1 M_2

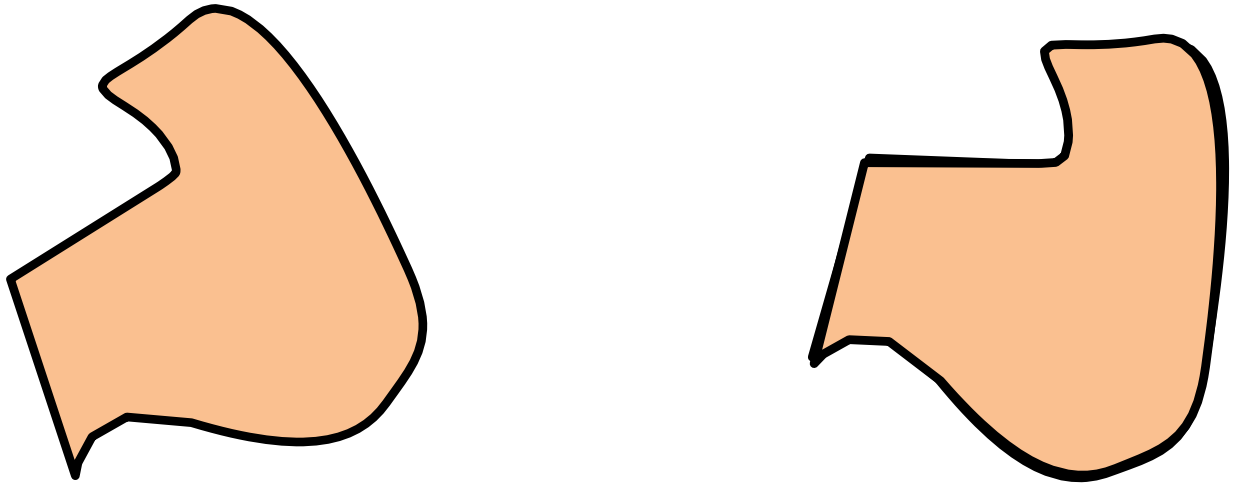


$$M_1 \approx T(M_2)$$

T: Translation + Rotation

Problème

M_1 M_2

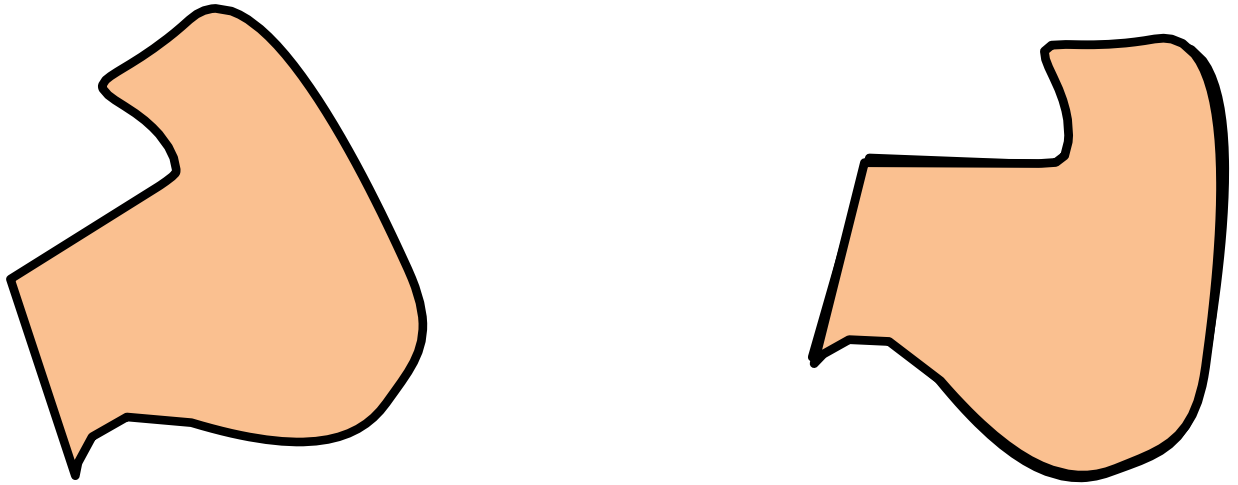


$$M_1 \approx T(M_2)$$

T: Translation + Rotation

Problème

M_1 M_2



Pour $M_1, \ldots, M_n$ trouver $T_2, \ldots, T_n$ tq

$$M_1 \approx T_2(M_2) \approx \cdots \approx T_n(M_n)$$

Difficultés

Différences globales et locales de forme

- Acquisition : décalage et rotation

→ alignement

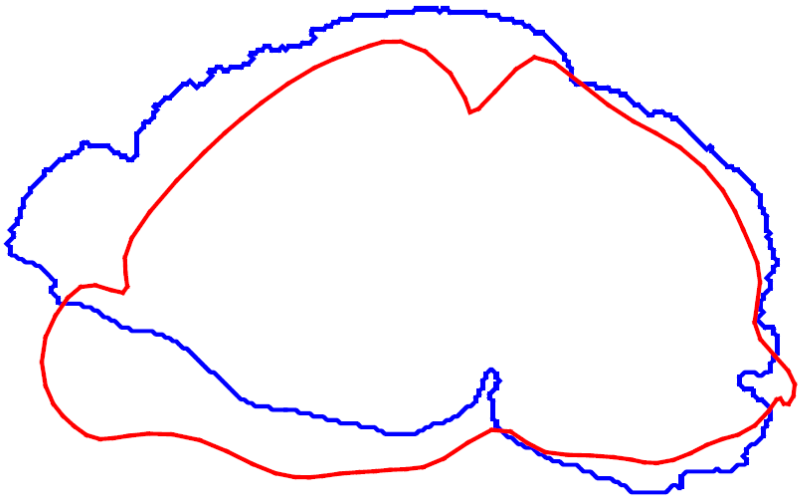
- La forme peut différer entre les modèles ou évoluer au cours du temps

→ déformation

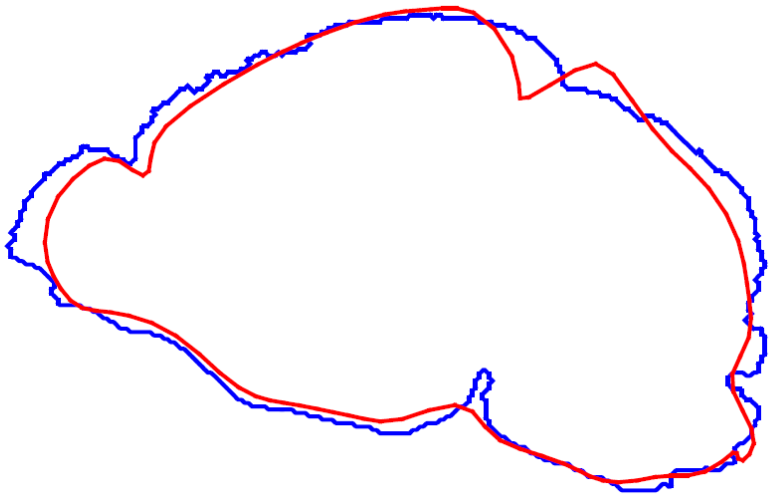
Alignement

Recalage par **translation** et **rotation**

- La structure reste « rigide »
- **Transformation rigide** ou **isométrique** (préservant les distances)
- Mathématiquement représentées par des opérations sur des vecteurs/matrices



Avant recalage



Après recalage

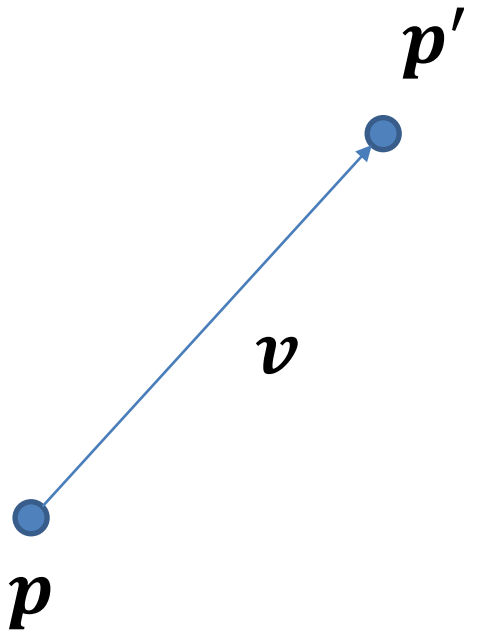
Transformations mathématiques

Translation

Addition de vecteurs : $\boldsymbol{p}' = \boldsymbol{v} + \boldsymbol{p}$

2D : $\begin{pmatrix} {p_x}' \\ {p_y}' \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix}$

3D : $\begin{pmatrix} {p_x}' \\ {p_y}' \\ {p_z}' \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$



Transformations mathématiques

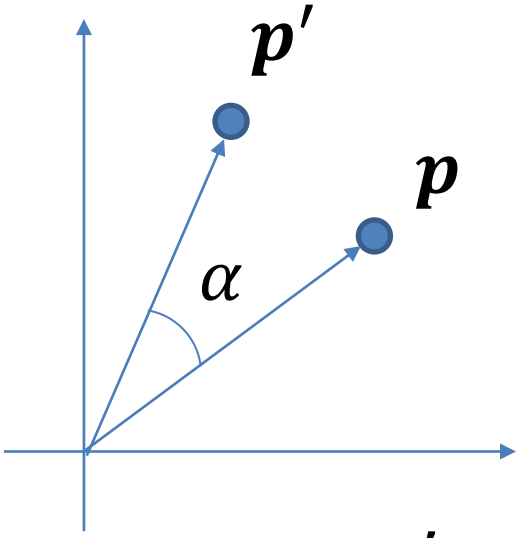
Rotation

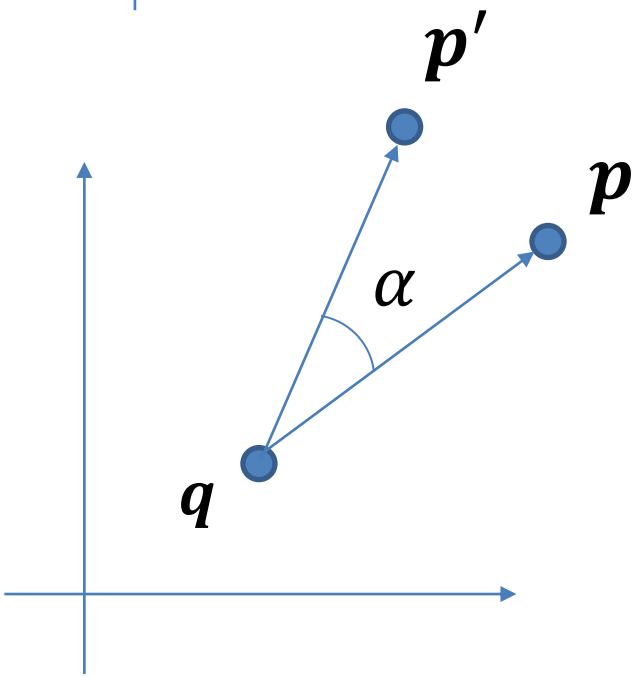
Produit matriciel : $\boldsymbol{p'} = \boldsymbol{R}.\boldsymbol{p}$

2D : $\begin{pmatrix} {p_x}' \\ {p_y}' \end{pmatrix} = \boldsymbol{R}.\begin{pmatrix} p_x \\ p_y \end{pmatrix}$

$\boldsymbol{R} = \begin{pmatrix} \cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & \cos(\alpha) \end{pmatrix}$

Rotation autour du point q :
$\boldsymbol{p'} = \boldsymbol{R}.(\boldsymbol{p} - \boldsymbol{q}) + \boldsymbol{q}$





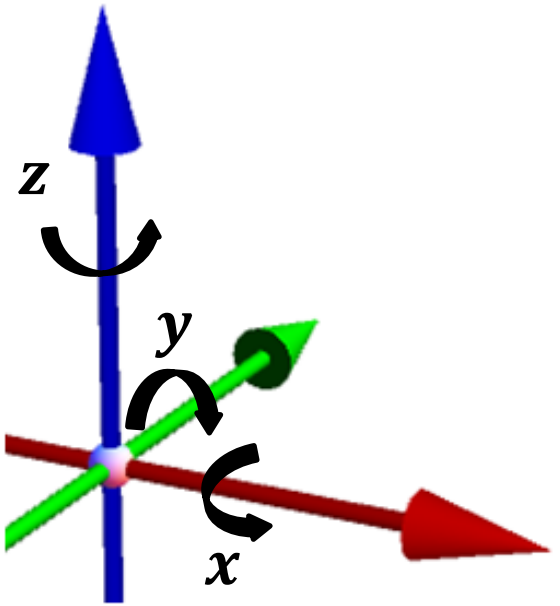
Rotations

Produit matriciel : $\boldsymbol{p'} = \boldsymbol{R}.\boldsymbol{p}$ **en** 3D : $\begin{pmatrix} p_x{}' \\ p_y{}' \\ p_z{}' \end{pmatrix} = \boldsymbol{R}.\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

$$R_y = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



Rotation arbitraire : composée de ces rotations

Rotations : propriétés

Orthogonale :

- $\boldsymbol{R}.\boldsymbol{R}^T = I$
- Vecteurs de norme 1 et orthogonaux 2 à 2

Exemples :

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}. \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Facile à inverser : $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$

Toute matrice orthogonale représente une rotation autour d'un axe (pas limité à $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$)

Rotation arbitraire : propriétés

Etant donnée une matrice orthogonale, l'**angle de rotation** peut être calculé par la **trace** de la matrice

- Trace : somme des valeurs sur la diagonale
- 2D : trace = $2\cos(\alpha)$ avec α l'angle de rotation
- 3D : trace = $1 + 2\cos(\alpha)$

Plus la trace est grande plus petit est l'angle

Valeurs et vecteurs propres

Soit M une matrice carrée, v est un vecteur propre et λ une valeur propre si :

$$M.v = \lambda * v$$

Si M représente une rotation (i.e. orthogonale), l'axe de rotation est un vecteur propre dont la valeur propre = 1

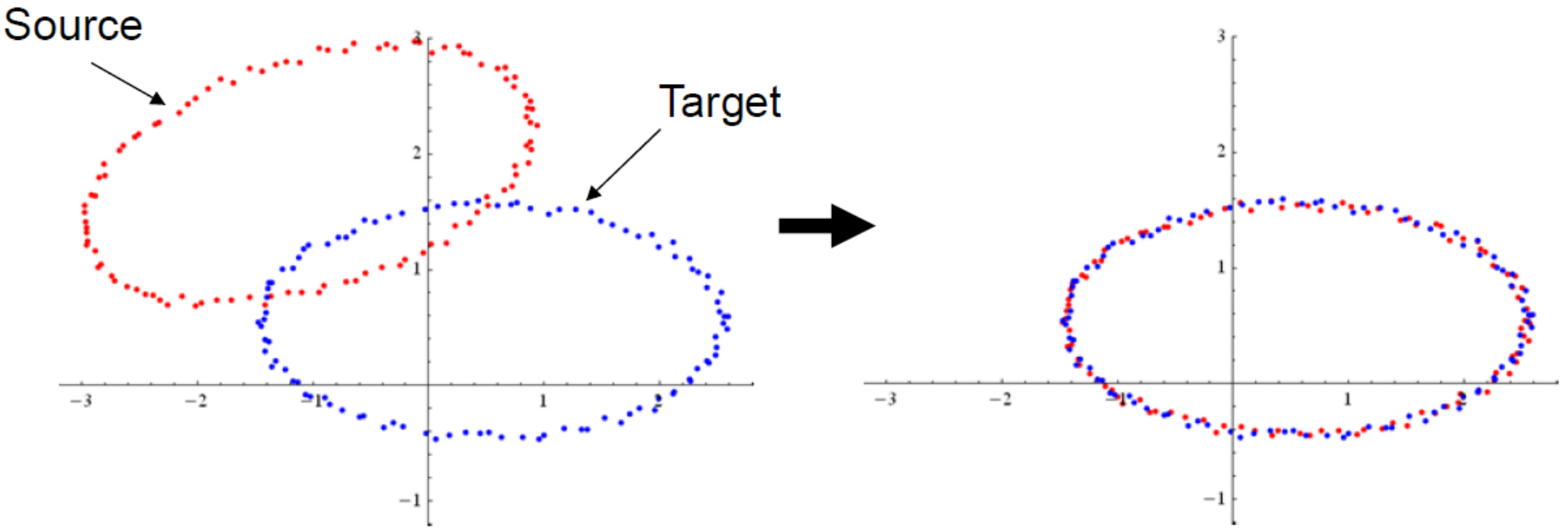
Il y a au maximum m valeurs propres distinctes pour une matrice de m x m

N'importe quel vecteur propre multiplié par un scalaire est aussi un vecteur propre (pour la même valeur propre)

Alignement

Entrée : 2 modèles représentés par des nuages de points
- Source et cible (target)

Sortie : positions des points sources bougés et tournés



Méthodes

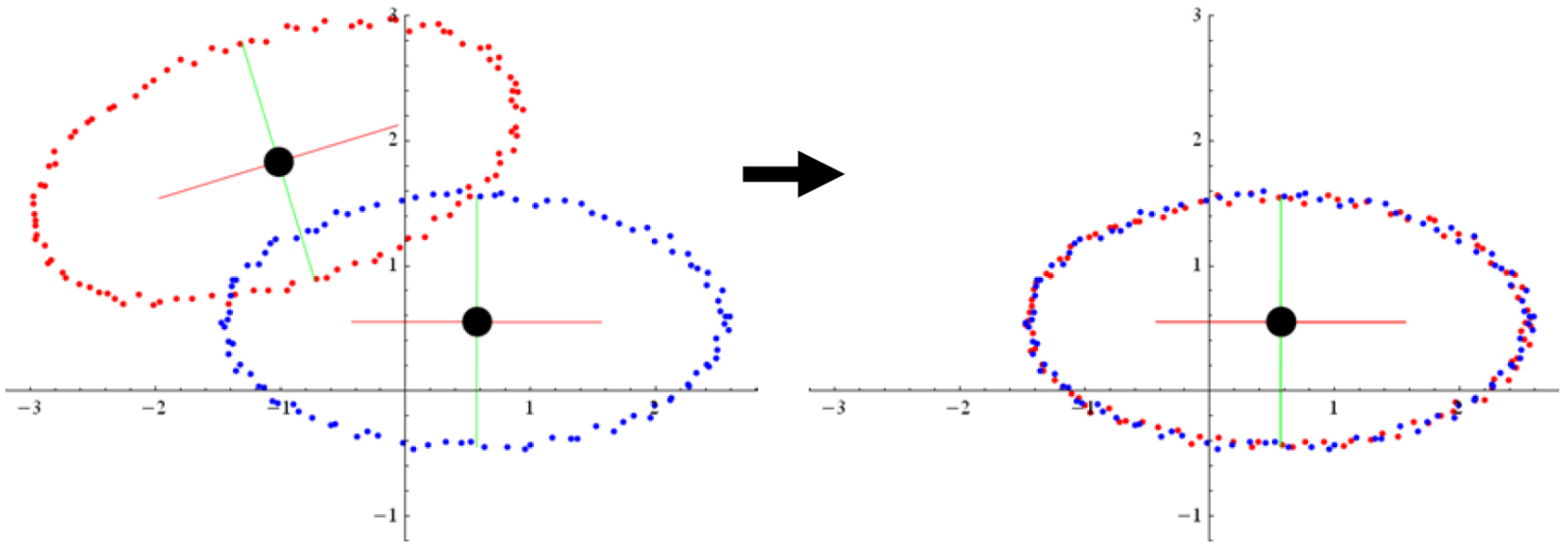
1. Analyse en composante principal (ACP)
   - Aligner les directions principales
2. Décomposition en Valeur Singulières (SVD)
   - Alignement optimal avec des connaissances à priori de correspondances
3. Iterative Closest Point (ICP)
   - Un algorithme SVD itératif avec calcul des correspondances

ACP ou PCA

Système de coordonnées dépendant la forme du modèle :

- **Origine** : centroïde des points
- **Axes** : directions dans lesquels le modèle varie le plus ou le moins

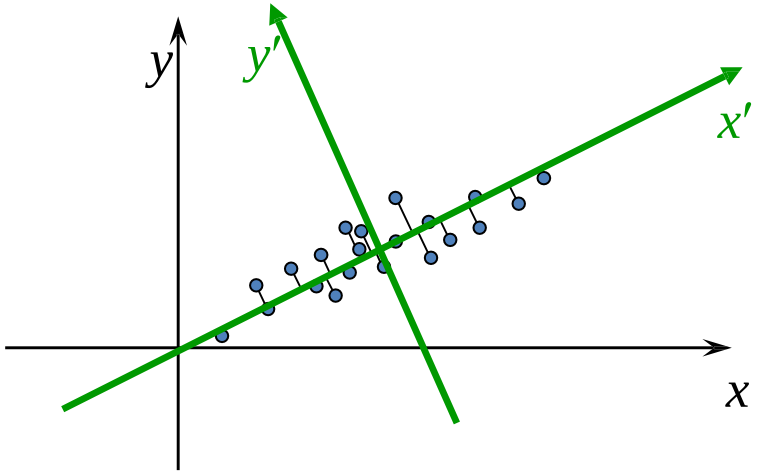
Transformer la source pour l'aligner avec son origine/axes de la cible

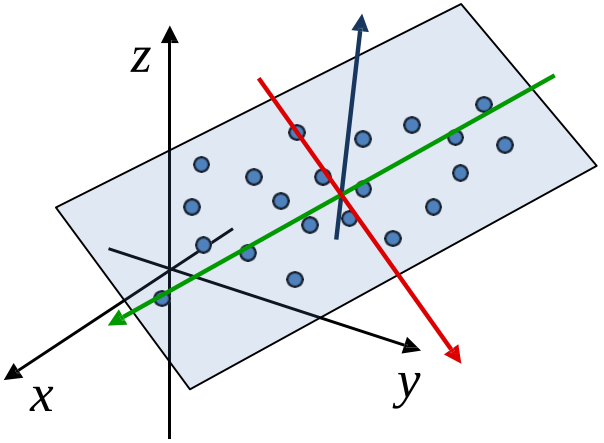


Cours ACP : https://www.youtube.com/watch?v=uV5hmpzmWsU

ACP ou PCA

Trouver la base orthogonale qui représente le mieux le jeu de données



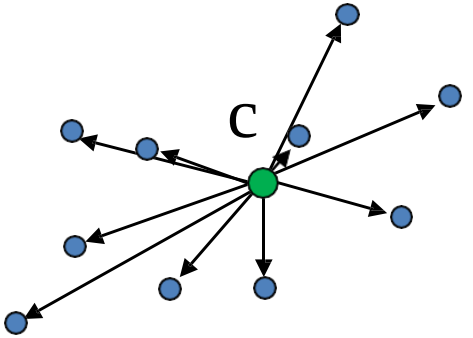


ACP trouve la meilleure approximation en termes des $\Sigma distances^2$

Notations

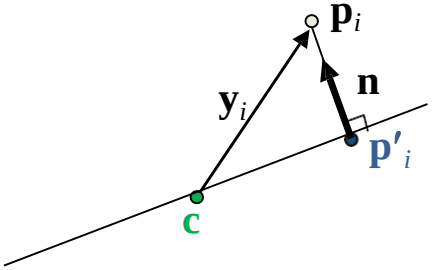
Points d'entrée : $p_1, p_2, \ldots, p_n \quad \in \mathbb{R}^d$

Centroide : $c = \frac{1}{n} \sum_{i=1}^{n} p_i$



Vecteur du centroïde :

$$y_i = p_i - \mathbf{c}$$



$\mathbf{c}$ sera l'origine du plane

Notre problème devient :

$$\min_{\|\mathbf{n}\|=1} \sum_{i=1}^{n} \left(\mathbf{y}_i^T \mathbf{n}\right)^2$$

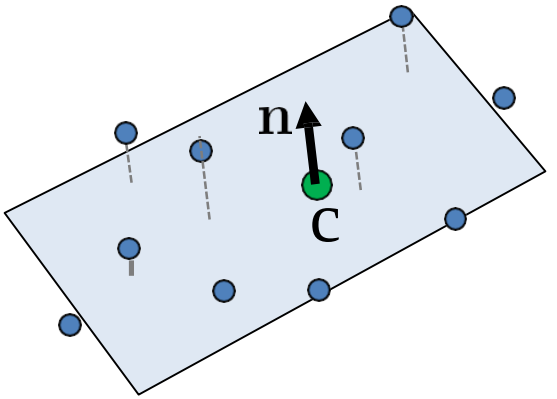
Normale du plan

Minimise

$$\min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \left(\mathbf{y}_i^T \mathbf{n}\right)^2 = \min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \mathbf{n}^T \mathbf{y}_i \mathbf{y}_i^T \mathbf{n} =$$

$$\min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^T \right) \mathbf{n} = \min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\mathbf{Y}\mathbf{Y}^T\right) \mathbf{n}$$

$$\mathbf{Y} = \begin{pmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_n \\ | & | & & | \end{pmatrix}$$



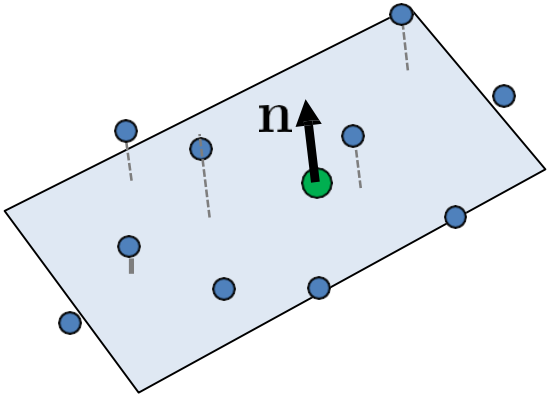
Normale du plan

Minimize!

$$\min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \left(\mathbf{y}_i^T \mathbf{n}\right)^2 = \min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \mathbf{n}^T \mathbf{y}_i \mathbf{y}_i^T \mathbf{n} =$$

$$\min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^T \right) \mathbf{n} = \min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\mathbf{Y}\mathbf{Y}^T\right) \mathbf{n}$$

$$\mathbf{Y} = \begin{pmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_n \\ | & | & & | \end{pmatrix}$$



$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$

$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T\mathbf{n} = 1$

Resumé - trouvé le meilleur plan

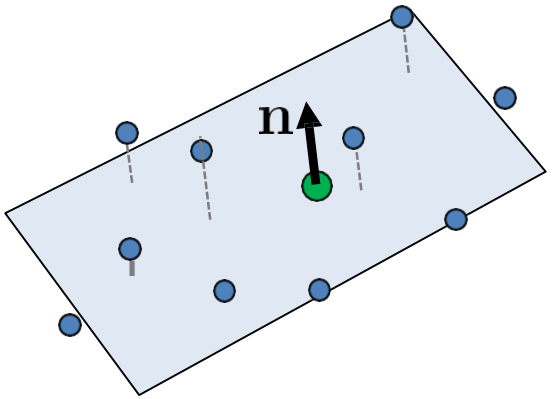
- Entrée : $p_1, p_2, \ldots, p_n \in \mathbb{R}^d$
- Centroïde = origine du plan $c = \frac{1}{n}\sum_{i=1}^{n} p_i$
- Calculer la matrice $\mathbf{S} = \mathbf{Y}\mathbf{Y}^T$

$$\mathbf{Y} = (\mathbf{y}_1\ \mathbf{y}_2\ \ \ldots\ \ \mathbf{y}_n)$$

$$y_i = p_i - c$$

- La normal au plan $\mathbf{n}$ est le vecteur propre de $\mathbf{S}$ avec la valeur propre la plus petite

$$\mathbf{S} = \mathbf{V}\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix}\mathbf{V}^T$$

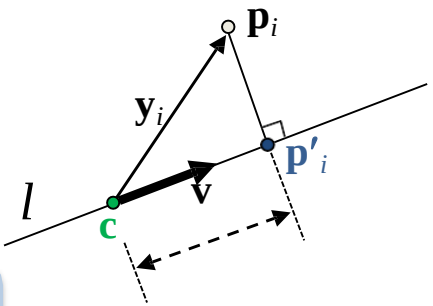


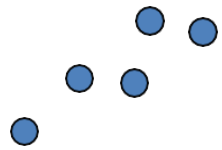
Que dit la matrice de covariance ?

- Pour une ligne l passant par le centre de masse $\mathbf{c}$ avec un vecteur de direction $\mathbf{v}$, projetons nos $\mathbf{p}_i$ La variance des points projetés $\mathbf{p'}_i$ est :

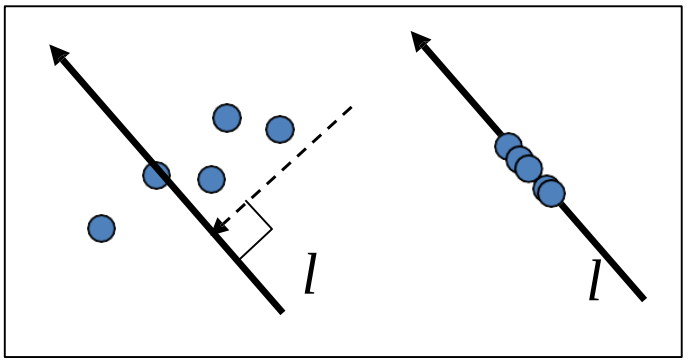
$$\text{var}(\boldsymbol{p}_1, \ldots \boldsymbol{p}_n; \mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n} \| \boldsymbol{p}'_i - \mathrm{c} \|^2 =$$

$$= \frac{1}{n}\sum_{i=1}^{n} \| (\mathbf{c} + \mathbf{v}^T\mathbf{y}_i) - \mathbf{c} \|^2 = \frac{1}{n}\sum_{i=1}^{n} \left(\mathbf{y}_i^T\mathbf{v}\right)^2 = \frac{1}{n}\mathbf{v}^T\mathbf{S}\mathbf{v}$$

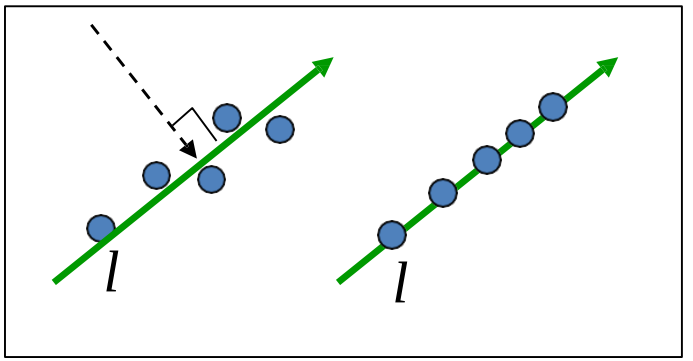




Original set



Small variance



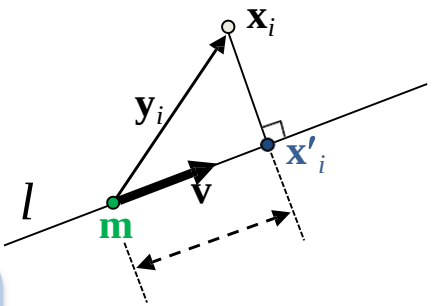
Large variance

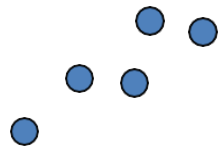
Que dit la matrice de covariance ?

- Mesure la variance des points d'entrée suivant la direction $\mathbf{v}$

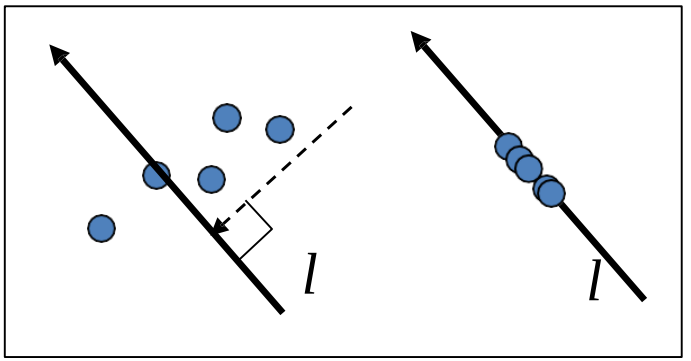
$$\mathrm{var}(\boldsymbol{p}_1, \ldots \boldsymbol{p}_n; \mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n} \| \boldsymbol{p}'_i - \mathrm{c} \|^2 =$$

$$= \frac{1}{n}\sum_{i=1}^{n} \| (\mathbf{c} + \mathbf{v}^T\mathbf{y}_i) - \mathbf{c} \|^2 = \frac{1}{n}\sum_{i=1}^{n} \left(\mathbf{y}_i^T\mathbf{v}\right)^2 = \frac{1}{n}\mathbf{v}^T\mathbf{S}\mathbf{v}$$

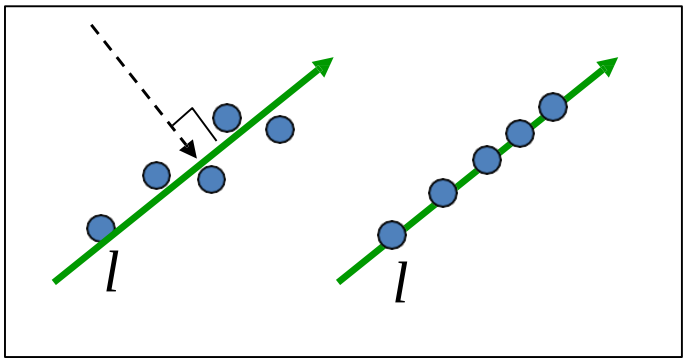




Original set



Small variance



Large variance

Principal Components

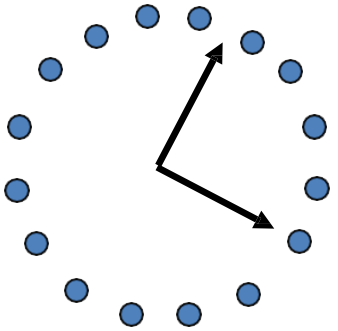
- Mesure la variance des points d'entrée suivant la direction $\mathbf{v}$

Les vecteurs propres de $\mathbf{S}$ avec une grande valeur propre sont les directions avec des composants forts (= grande variance).

Si les vp sont équivalentes il n'y a pas de direction privilégiée.

$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix} \mathbf{V}^T$$

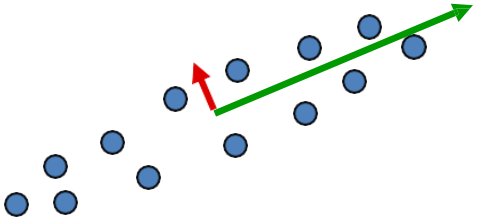
Principal Components



- There's no preferable direction
- S looks like this:

$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda & \\ & \lambda \end{pmatrix} \mathbf{V}^T$$

- Any vector is an eigenvector



- There's a clear preferable direction
- S looks like this:

$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda & \\ & \mu \end{pmatrix} \mathbf{V}^T$$

- μ is close to zero, much smaller than λ

ACP

Nuage de points $p_1, \dots, p_n$ avec leur centroïde c

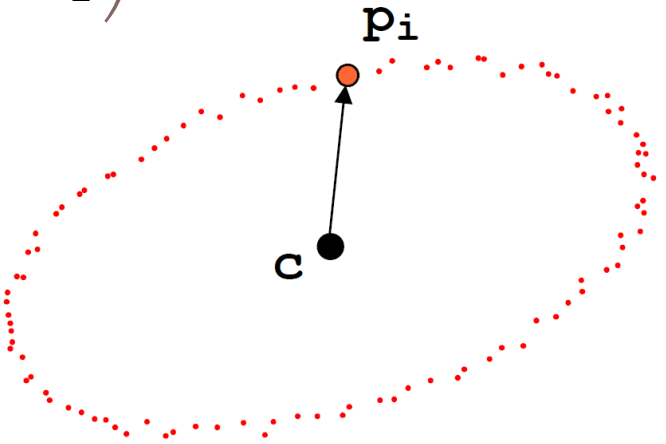
On construit la matrice Y : i^{ème} colonne = vecteur $p_i - c$

Matrice 2xn : $P = \begin{pmatrix} p_{1_x} - c_x & p_{2_x} - c_x & \cdots & p_{n_x} - c_x \\ p_{1_y} - c_y & p_{2_y} - c_y & \cdots & p_{n_y} - c_y \end{pmatrix}$

Matrice 3xn : $P = \begin{pmatrix} p_{1_x} - c_x & p_{2_x} - c_x & \cdots & p_{n_x} - c_x \\ p_{1_y} - c_y & p_{2_y} - c_y & \cdots & p_{n_y} - c_y \\ p_{1_z} - c_z & p_{2_z} - c_z & \cdots & p_{n_z} - c_z \end{pmatrix}$

La matrice de covariance : $S = P.P^T$

- 2D : une matrice 2x2
- 3D : une matrice 3x3



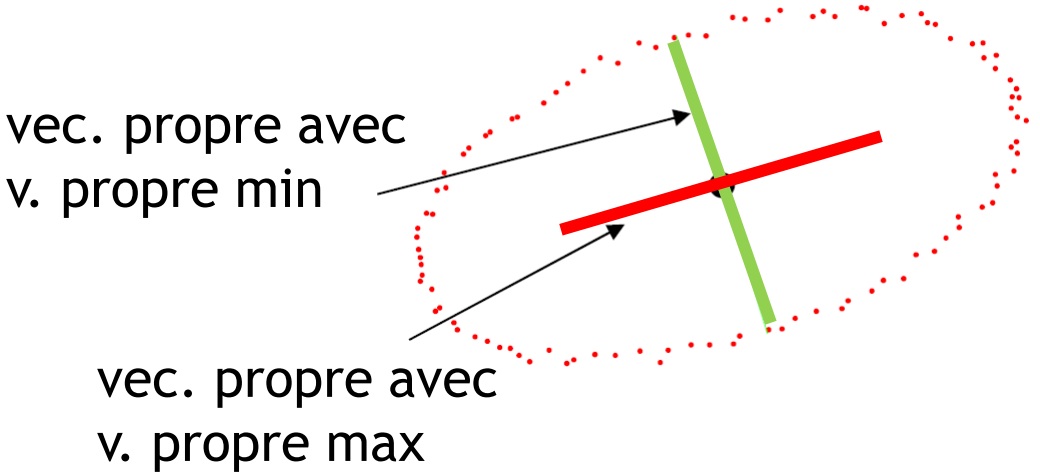
ACP

Vecteurs propres de la matrice de covariance : directions principales de variation de forme

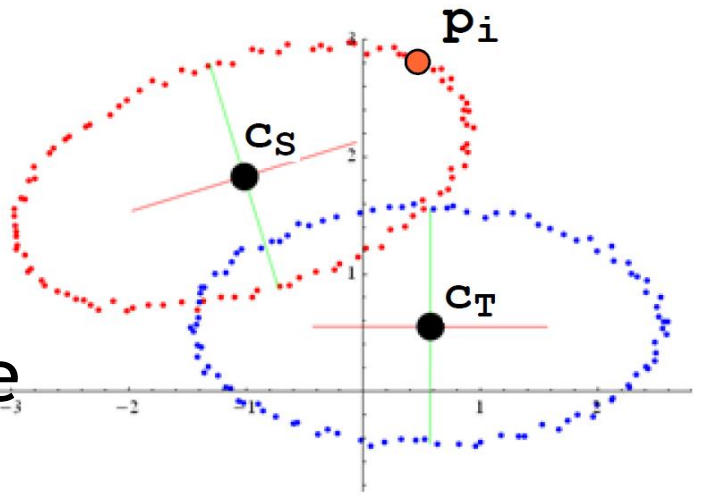
- Non signés et orthogonaux

Valeurs propres : amplitude de la variation suivant chaque vecteur propre

- plus grande (petite) vp = direction de plus grande (petite) variation de forme.



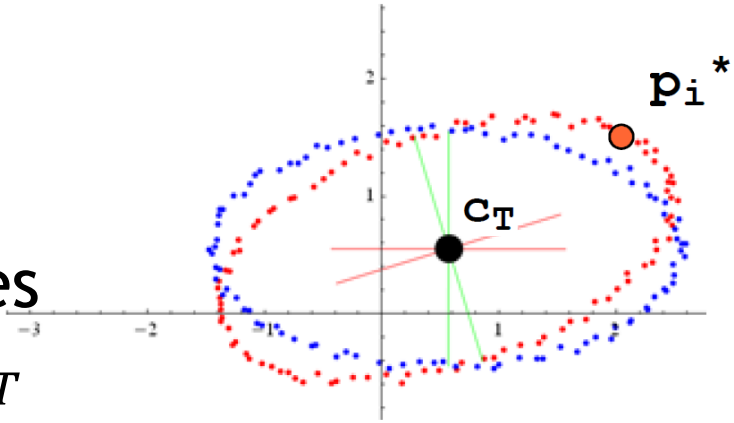
Alignement : ACP



Avec c_S, c_T les centroïdes de la source et la cible

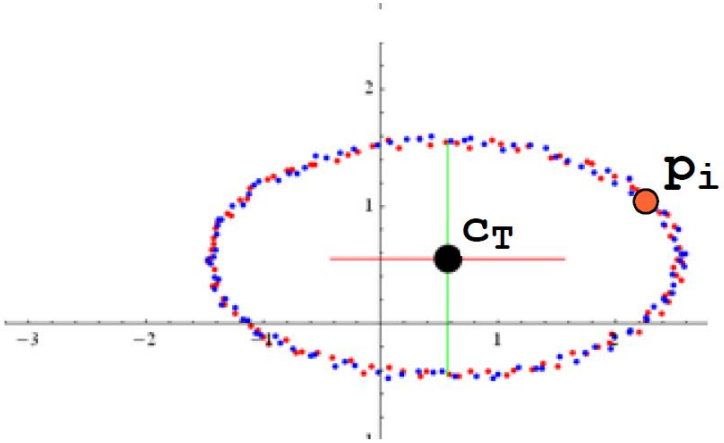
1 - déplacer pour aligner c_S et c_T avec

$$p_i{}^* = p_i + (c_T - c_S)$$



2 - trouver la rotation R alignant les axes de ACP et tourner la source autour de c_T

$$p_i{}' = c_T + R.(p_i{}^* - c_T)$$



Combiné :

$$p_i{}' = c_T + R.(p_i - c_S)$$

Alignement : ACP

Trouver la rotation entre des **axes orientés** :

Soit les matrices A, B dont les colonnes sont les axes

- axes orthogonaux et normalisés (ie A et B sont orthogonales)

On veut calculer une matrice de rotation R tq :

$$R.A = B$$

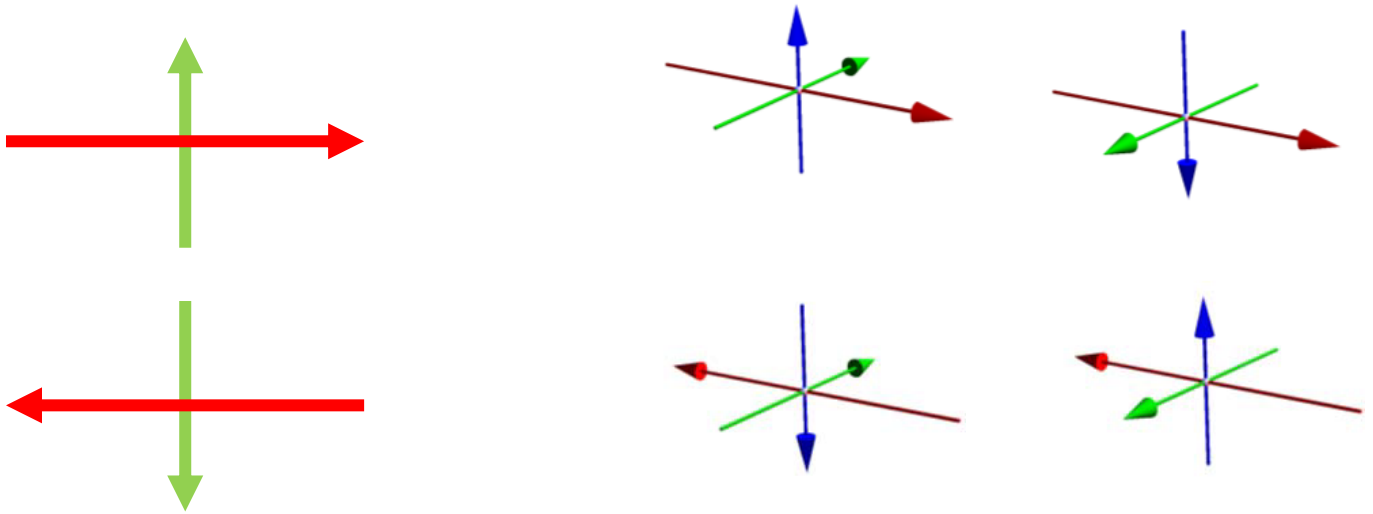
Comme $A\ et\ B$ sont orthogonales :

$$R = B.A^{-1} = B.A^{T}$$

Alignement : ACP

Assigner une orientation aux axes ACP

- 2 possibilités en 2D
- En 3D, 4 possibilités (loi de la main droite)



1er vecteur propre

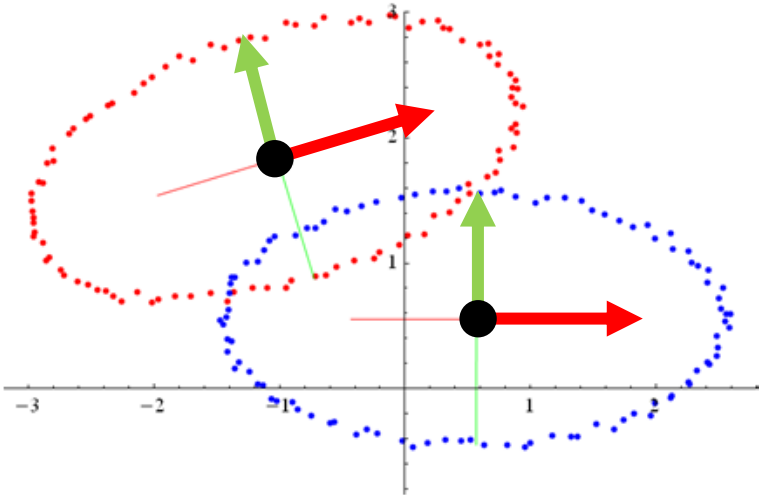
2ème vecteur propre

3ème vecteur propre

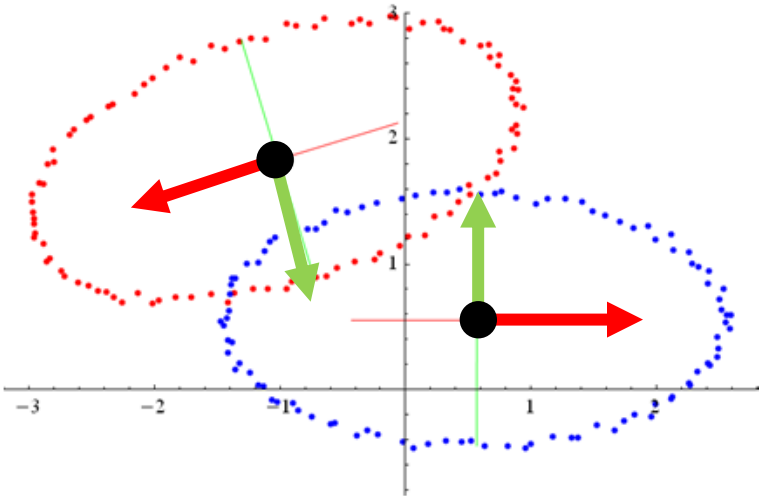
Alignement : ACP

Trouver la rotation entre des axes orientés :

- Fixer l'orientation des axes cibles
- Pour chaque orientation possible de la source, calculer R
- Choisir le R avec l'angle de rotation le plus petit (trace de R)



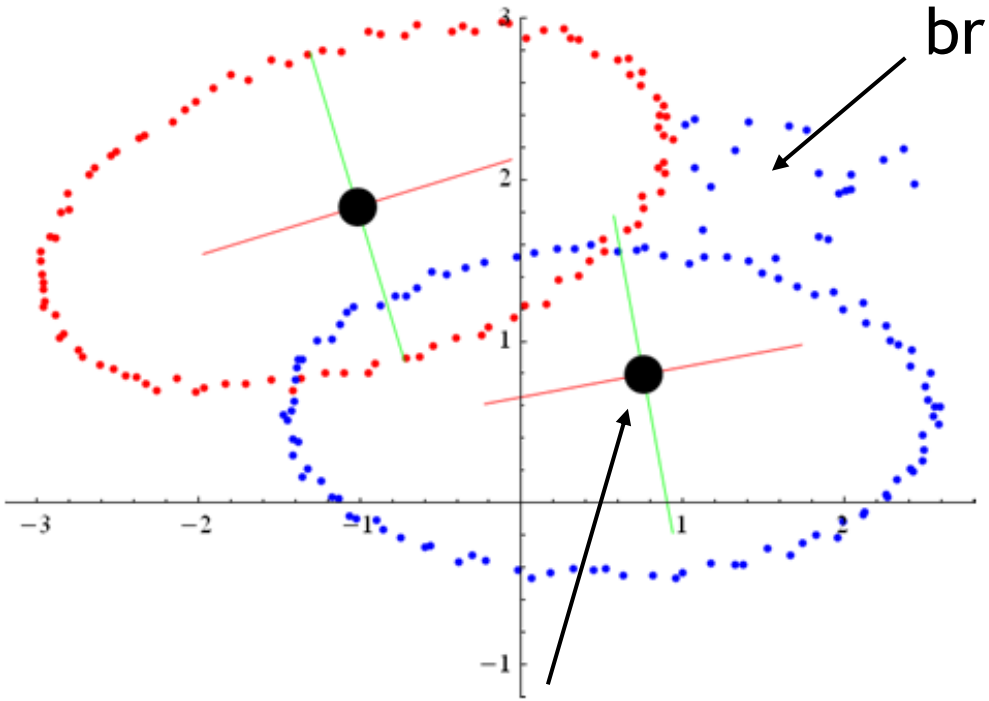
Plus petite rotation

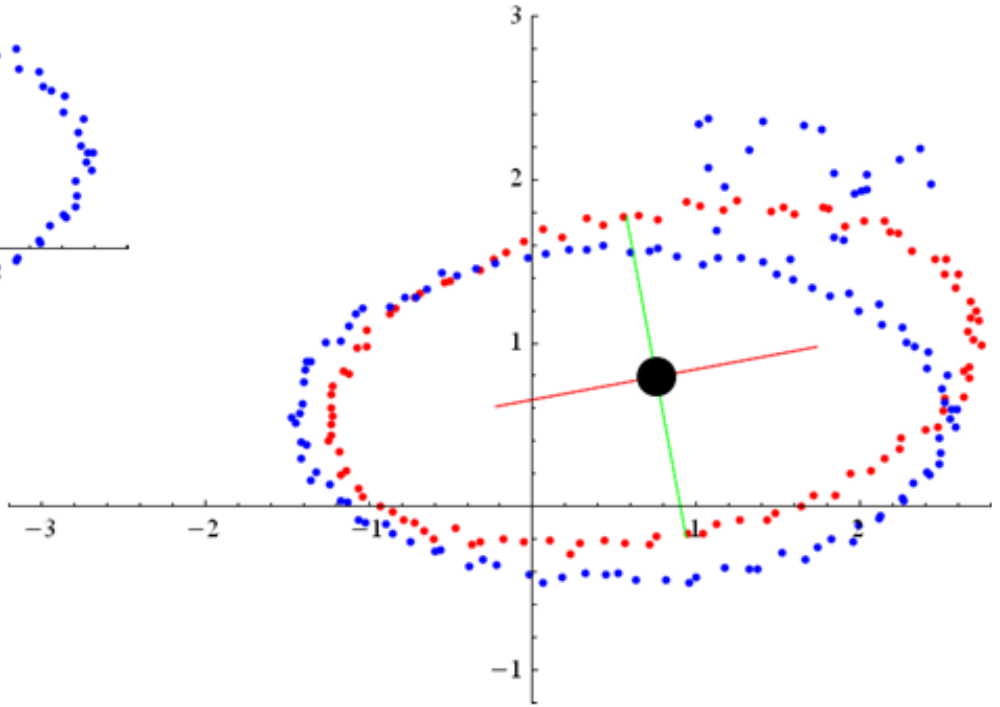


Plus grande rotation

Limitations

Centroïdes sensibles au bruit

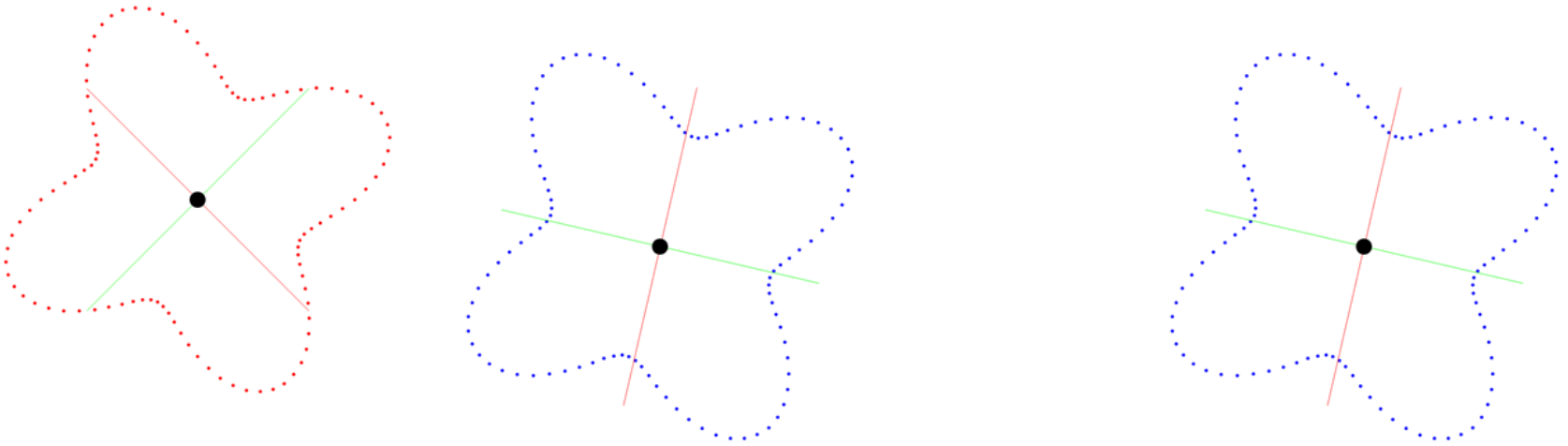




Résultat de l'ACP

Limitations

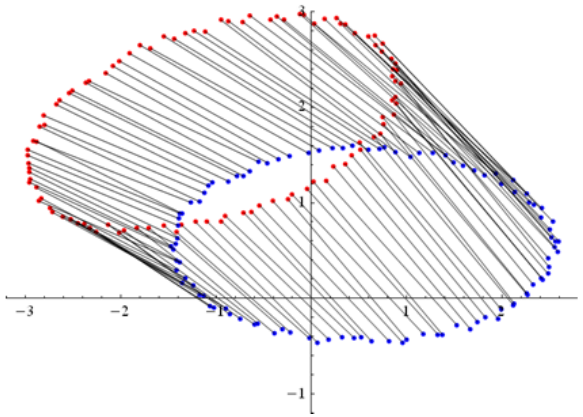
Les axes ne sont pas fiables pour les objets circulaires
Les valeurs propres deviennent similaires
Les vecteurs propres instables

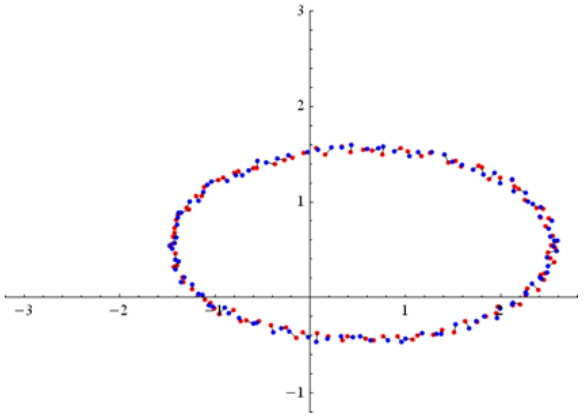


SVD

Alignement optimal entre des points correspondants

- Supposons qu'on connaisse la correspondance entre les points de la source et de la cible





SVD

Formulation du problème :

- Points sources $p_1, \dots, p_n$ avec leur centroïde c_S
- Points cibles $q_1, \dots, q_n$ avec leur centroïde c_T
- q_i correspond au point p_i

- Après centrage et rotation par R, un point source transformer se trouve à :

$$p_i' = c_T + R.(p_i - c_S)$$

Trouver R minimisant la somme des paires de distance :

$$E = \sum_{i=1}^{n} \|q_i - p_i'\|^2$$

SVD

Equivalent à
Soit P une matrice dont la i^{ème} colonne est le vecteur $p_i - c_S$
Soit Q une matrice dont la i^{ème} colonne est le vecteur $q_i - c_T$

Considérons la matrice de covariance croisée :

$$M = P.Q^T$$

Trouver la matrice de rotation orthogonale qui maximise la trace :

$$Tr(R.M)$$

SVD

Résoudre le problème de minimisation
Singular Value Decomposition (SVD) d'une matrice mxm M :

$$M = U.W.V^T$$

- U, V sont des matrice mxm orthogonales (i.e. rotations)
- W est une matrice mxm diagonale sans entrées négatives

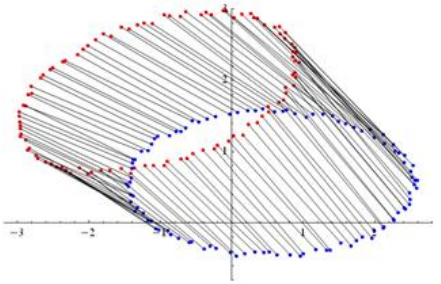
La matrice orthogonale (rotation) $R = V.U^T$ est R maximisant $Tr(R.M)$

SVD disponible dans de nombreuse librairies C++

Alignement par SVD

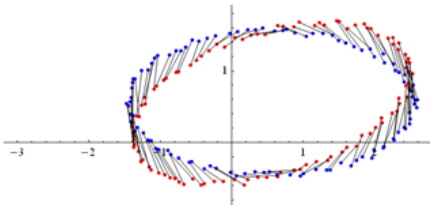
Former la matrice de covariance croisée :

$$M = P.Q^T$$



Calculer la SVD :

$$M = U.W.V^T$$

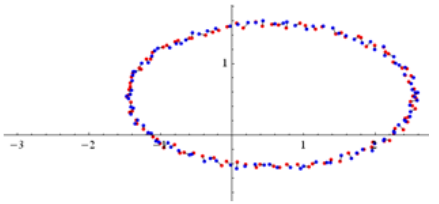


La matrice de rotation est :

$$R = V.U^T$$

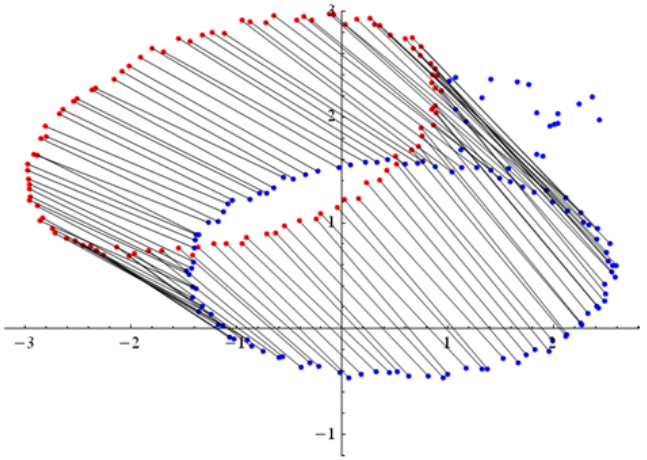
Déplacer et tourner la source :

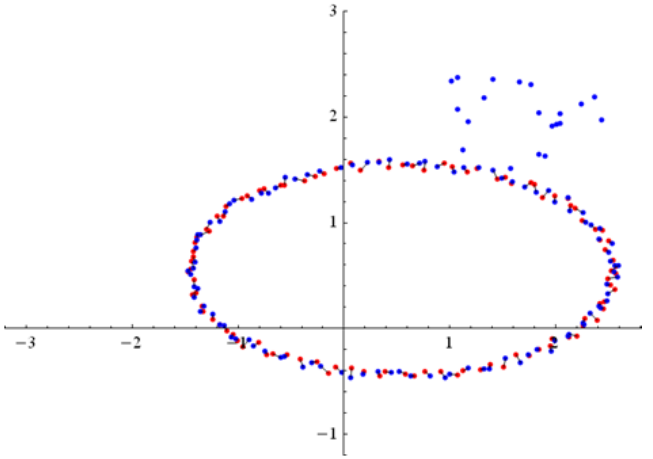
$$p_i{}' = c_T + R.(p_i - c_S)$$



Alignement par SVD

Plus stable que l'ACP

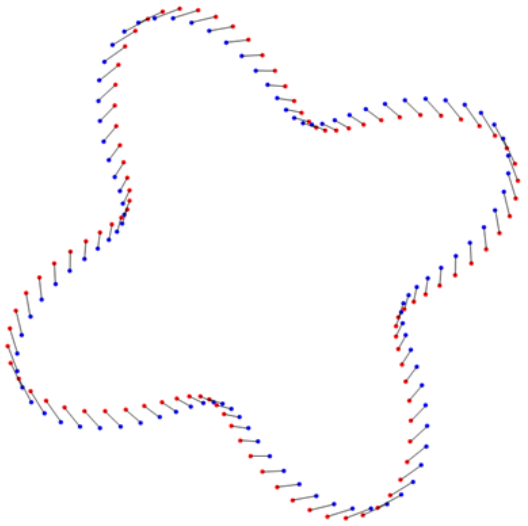


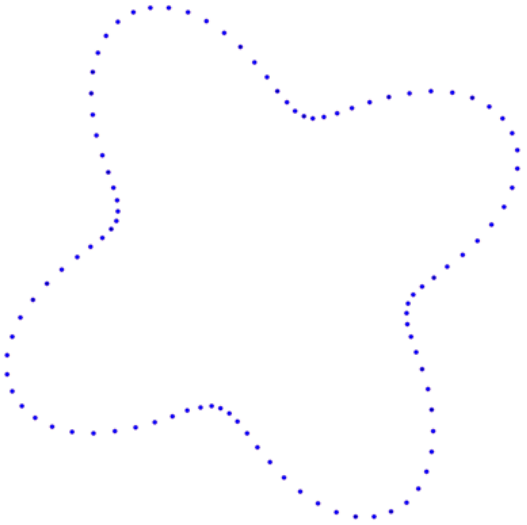


Si les correspondances sont correctes…

Alignement par SVD

Plus stable que l'ACP





Si les correspondances sont correctes…

Alignement par SVD

Limitation :

Nécessite des correspondances précises la plus part du temps non disponibles

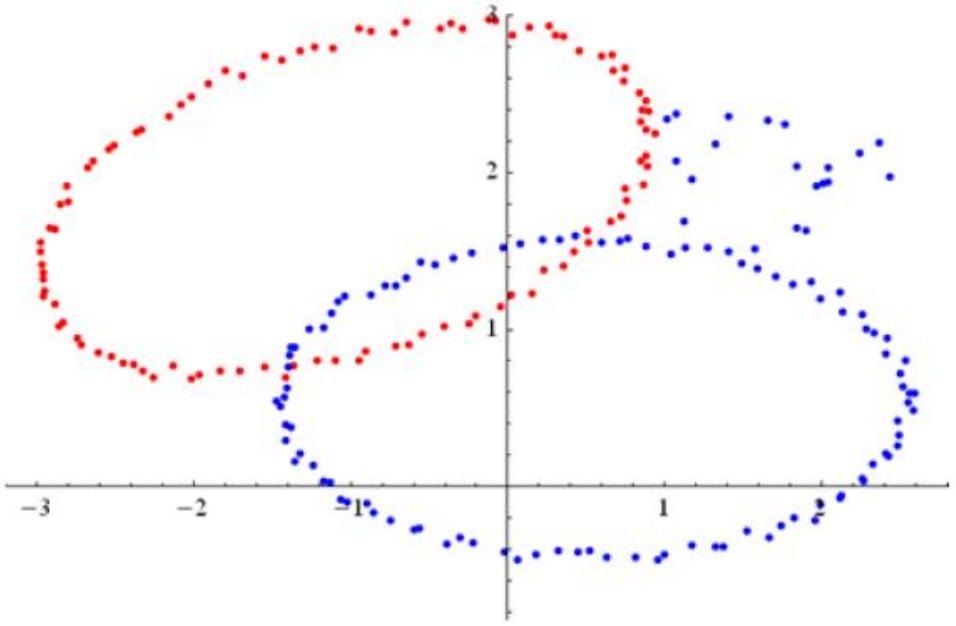
→ ICP

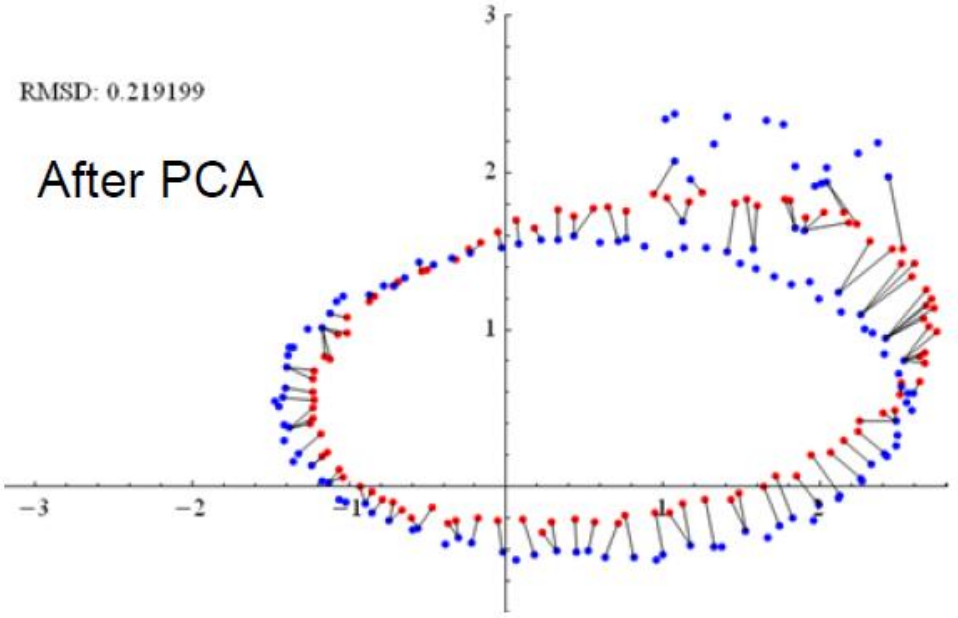
ICP : Iterative Closest Point

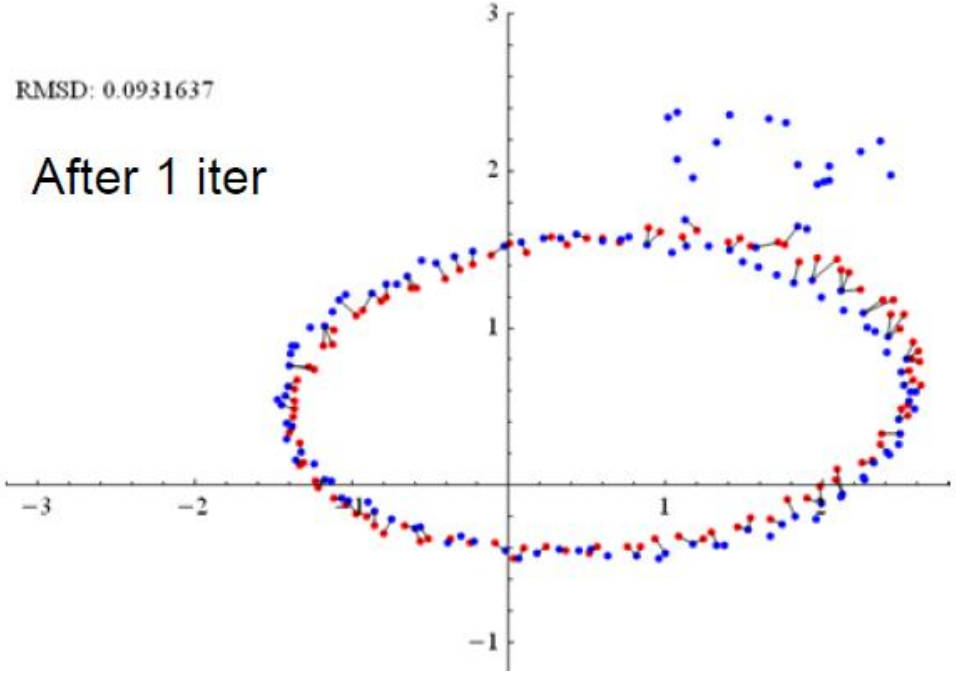
Idée :
Utiliser l'alignement ACP pour obtenir une première initialisation
Améliorer itérativement les correspondances après des SVD répétées

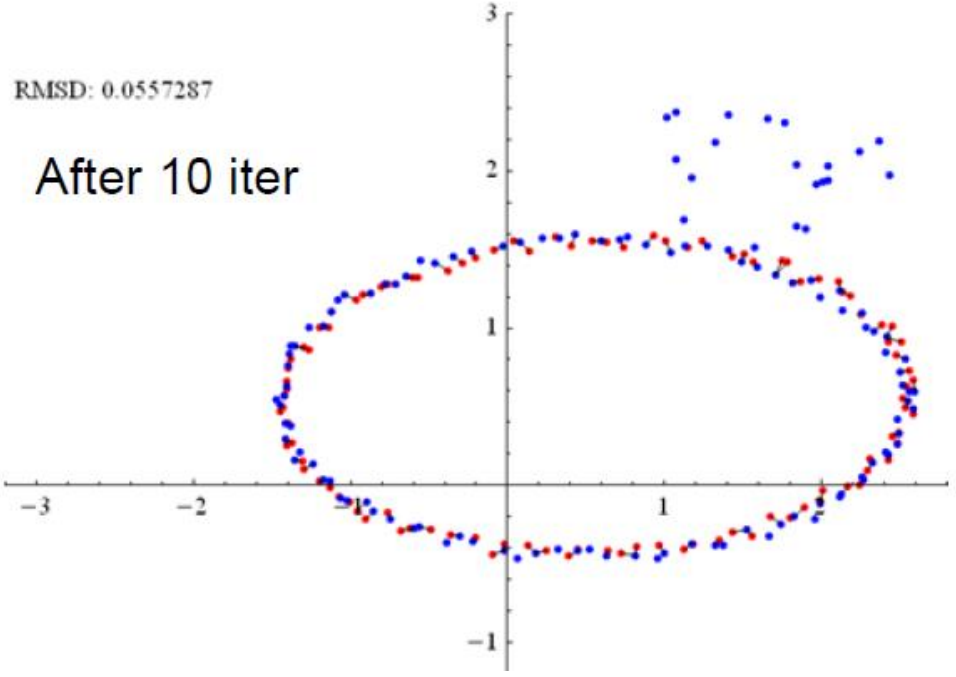
1 – Transformer la source avec ACP
2 – Pour chaque point transformer, allouer le point cible le plus proche. Aligner source et cible avec une SVD
Tous les points cibles n'ont pas besoin d'être utilisés
3 – Répéter 2 jusqu'à atteindre un critère d'arrêt

ICP




RMSD: 0.219199
After PCA


RMSD: 0.0931637
After 1 iter


RMSD: 0.0557287
After 10 iter

ICP

Critère de fin

- nombre d'itération max
- l'amélioration devient faible :

Root Mean Squared Distance (RMSD)

$$\sqrt{\frac{\sum_{i=1}^{n} \left\| q_i - p_i' \right\|^2}{n}}$$

Mesure de la déviation moyenne de toutes les correspondances

Arrêter quand la différence des RMSD avant et après < seuil utilisateur

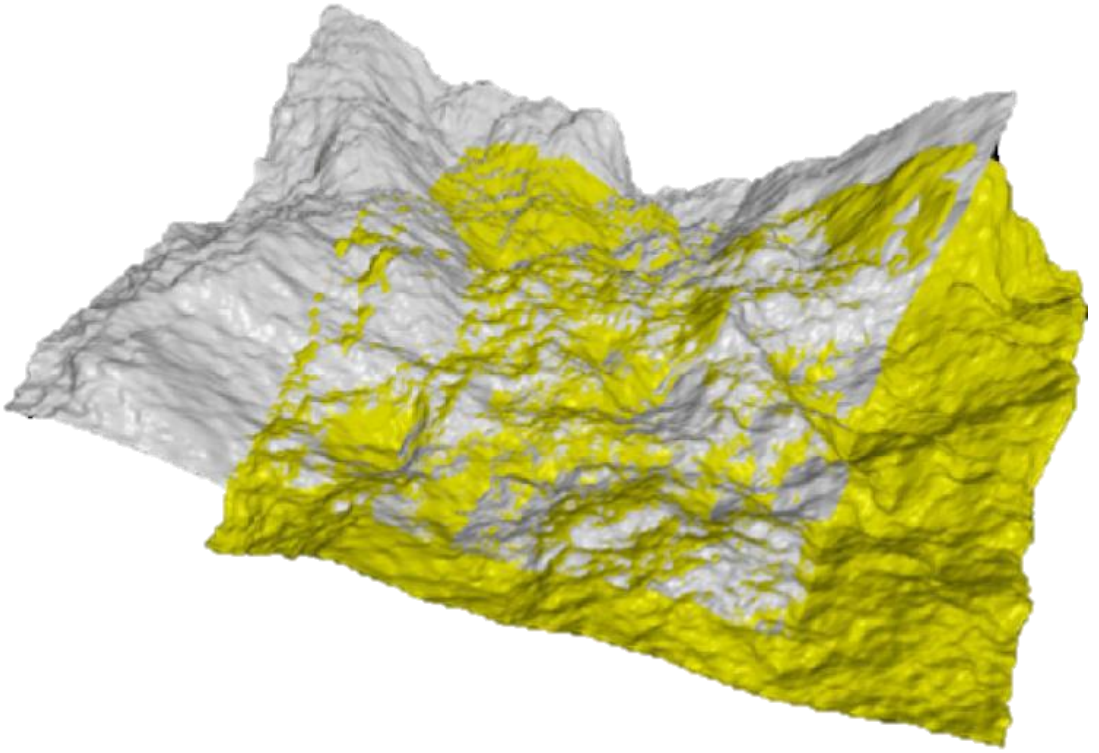
Correspondances

Combien de points sont nécessaires pour trouver la bonne transformation ?

Le premier problème est de trouver les paires

$$\mathbf{p}_1 \rightarrow \mathbf{q}_1$$
$$\mathbf{p}_2 \rightarrow \mathbf{q}_2$$
$$\mathbf{p}_3 \rightarrow \mathbf{q}_3$$

$$R\mathbf{p}_i + t \approx \mathbf{q}_i$$



ICP : Iterative Closest Point

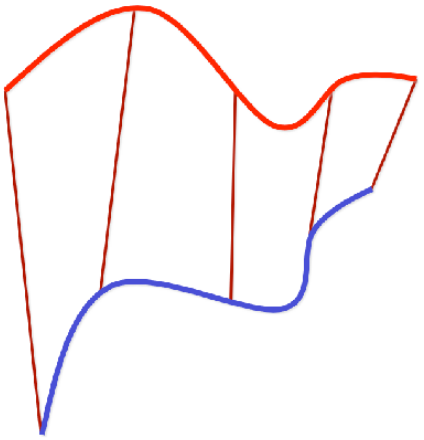
Intuition :

Correspondances correctes ⇒ pb résolu !

Idée :

(1) Trouver les correspondances

(2) Les utiliser pour trouver la transformation



ICP : Iterative Closest Point

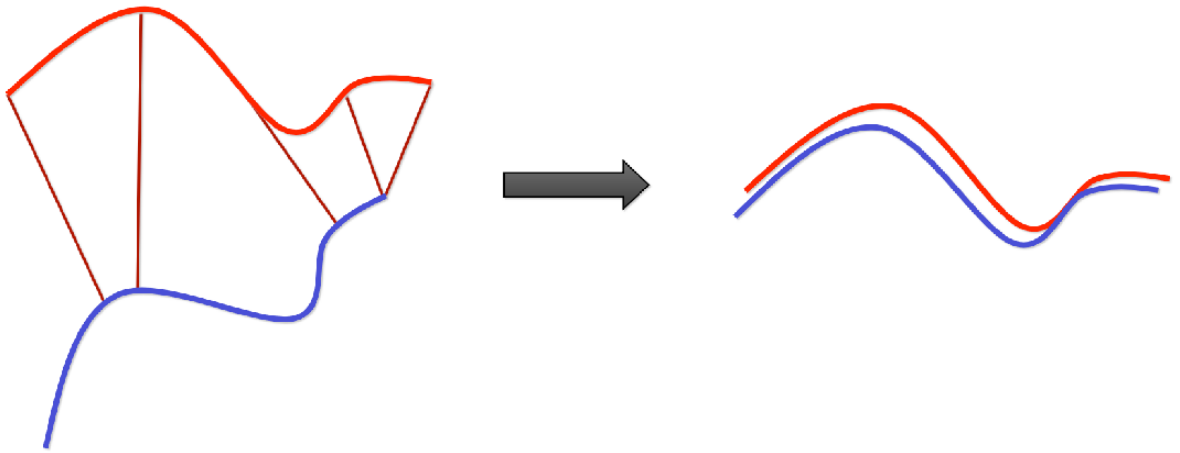
Intuition :

Correspondances correctes ⇒ pb résolu !

Idée :

(1) Trouver les correspondances

(2) Les utiliser pour trouver la transformation



ICP : Iterative Closest Point

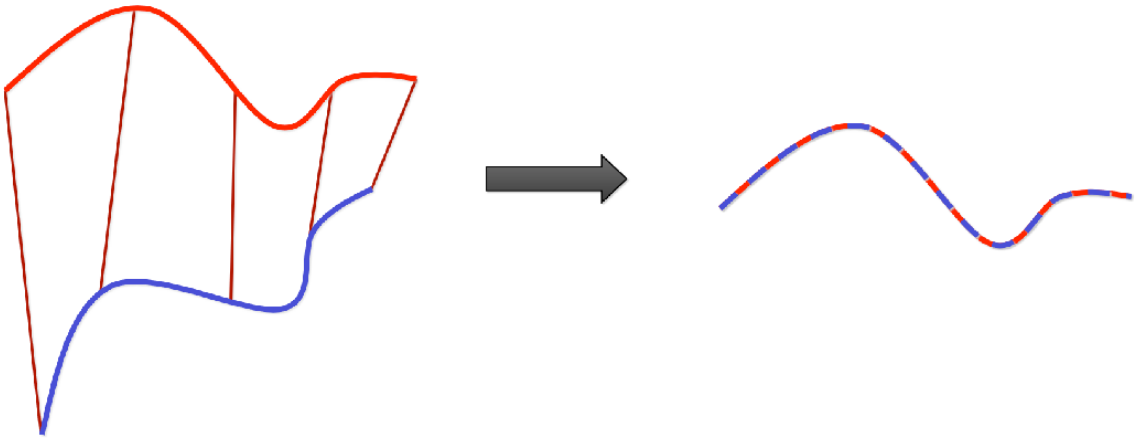
Intuition :

Correspondances correctes ⇒ pb résolu !

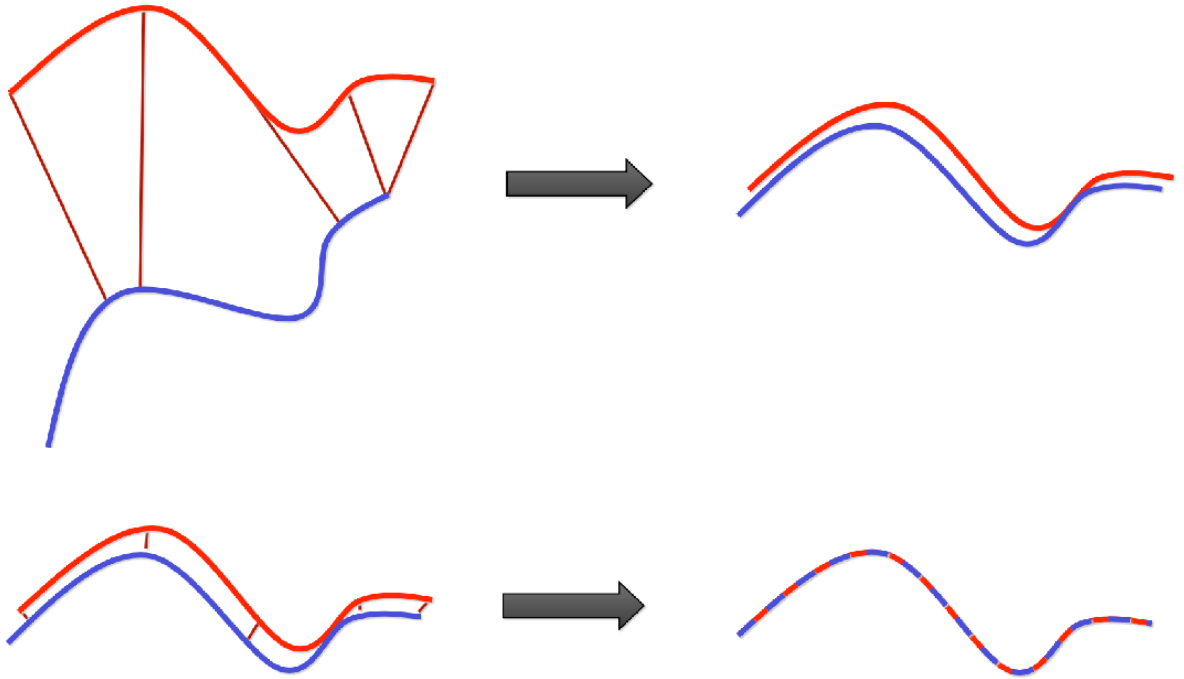
Idée :

(1) Trouver les correspondances

(2) Les utiliser pour trouver la transformation



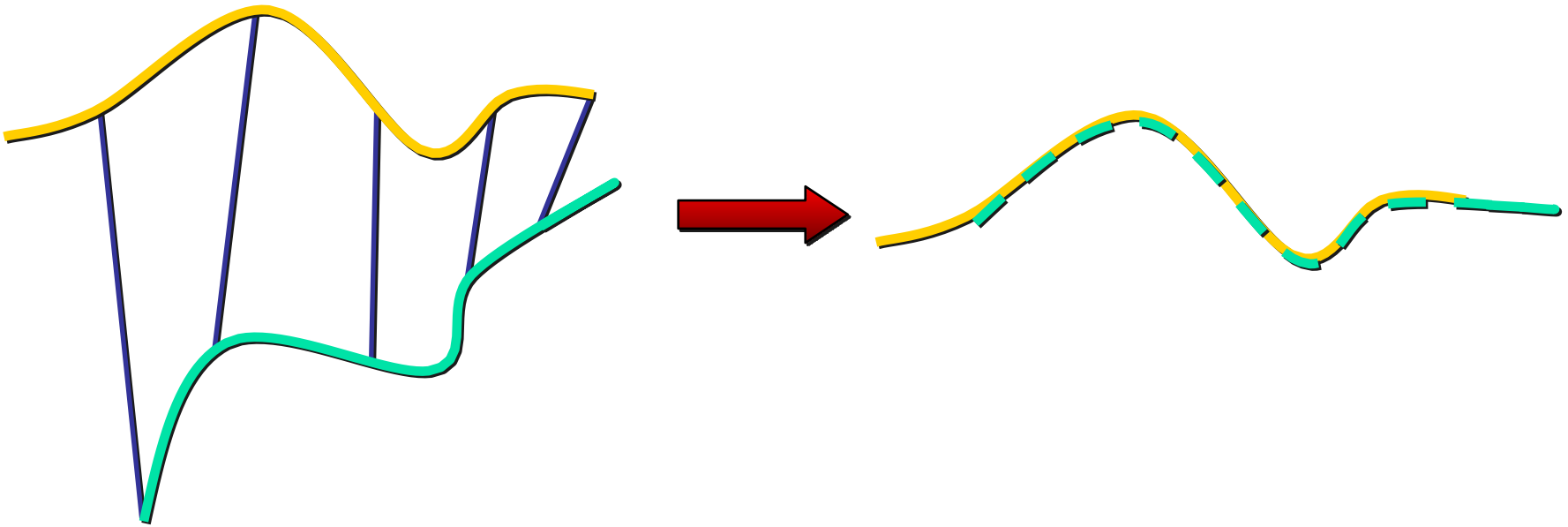
ICP : Iterative Closest Point



Cet algorithme converge vers la bonne solution
si les scans sont « assez proches »

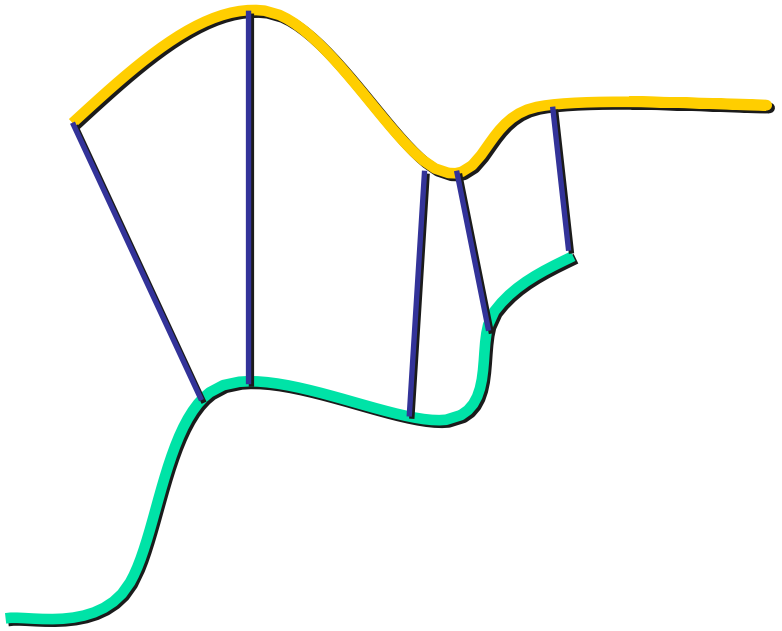
ICP : Iterative Closest Points

Si on connaît une correspondance entre deux pointsets, on peut les aligner (trouver une rotation et une translation pour aligner un sur l'autre).



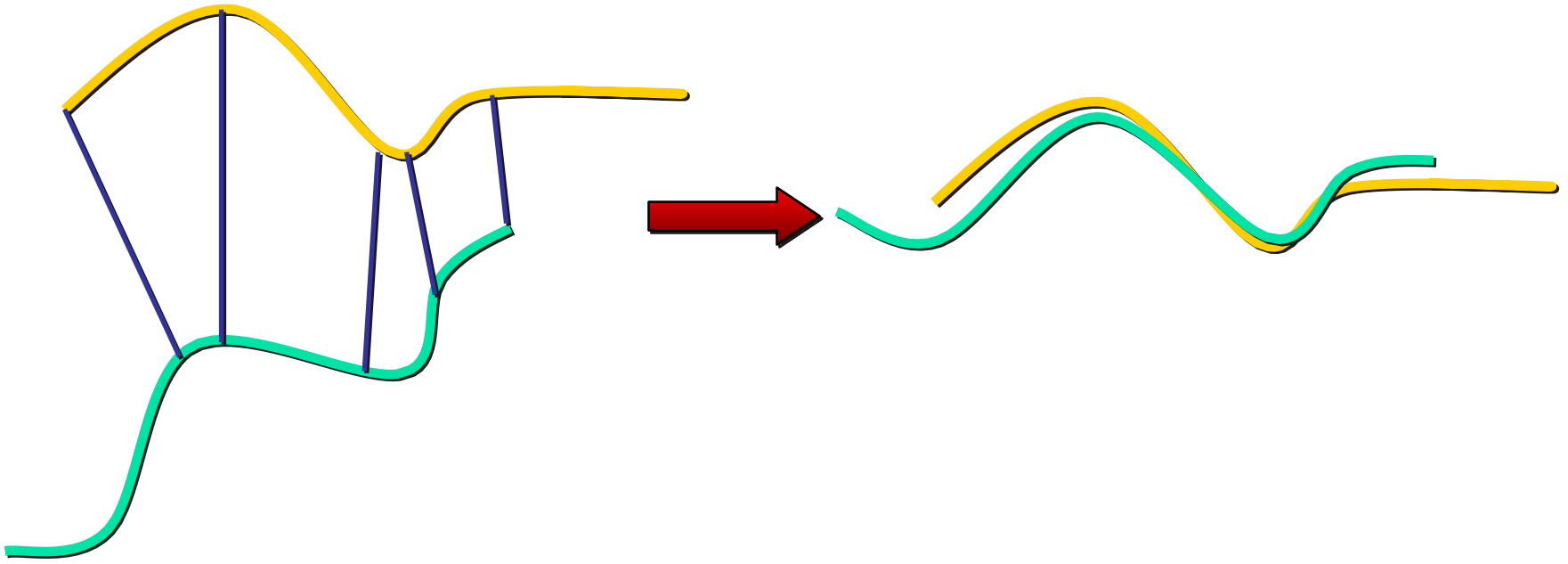
ICP : Iterative Closest Points

Si les deux pointsets sont relativement proches, alors on peut dire que le point le plus proche dans le pointset vert est un bonne correspondance avec un point dans le pointset jaune.



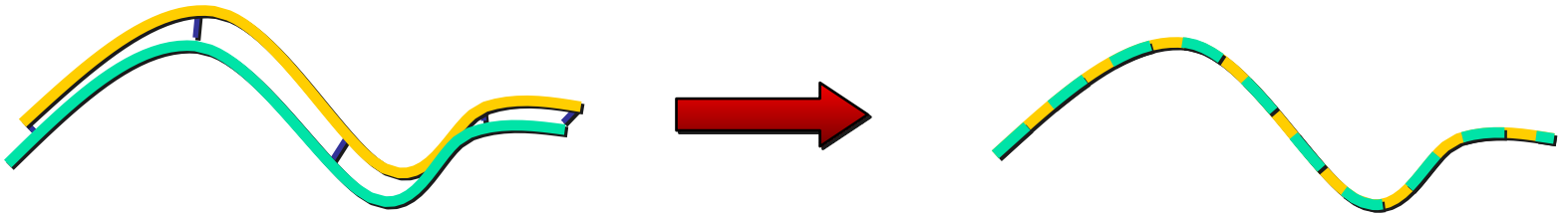
ICP : Iterative Closest Points

Ce ne sera pas parfait, mais l'alignement sera meilleur



ICP : Iterative Closest Points

Et en iterant, cela finit par converger (si il y a suffisamment d'overlap dans les deux pointsets, et que l'alignement initial n'est pas non plus trop mauvais)



Algorithme de base

Séléctionner (e.g.1000) des points aléatoires

Associer au plus proche voisin

Rejeter paires avec une distance trop grande

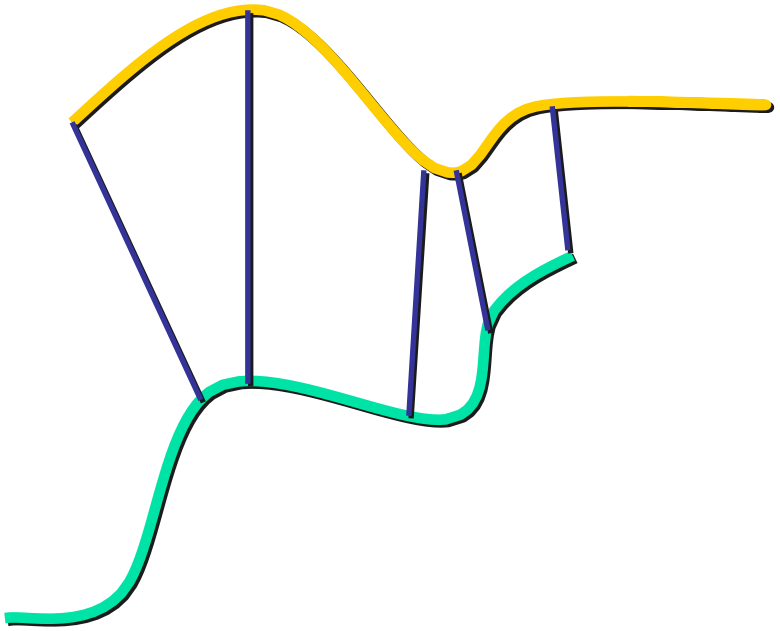
Minimiser

$$E := \sum_i (R\mathbf{p}_i + t - \mathbf{q}_i)^2$$

Solution :
http://dl.acm.org/citation.cfm?id=250160

Correspondance

1) Trouver, pour chaque point $p_i \in P$ le point le plus proche dans $q_i \in Q$ et enregistrer la correspondance : $\varphi(p_i) = q_i$

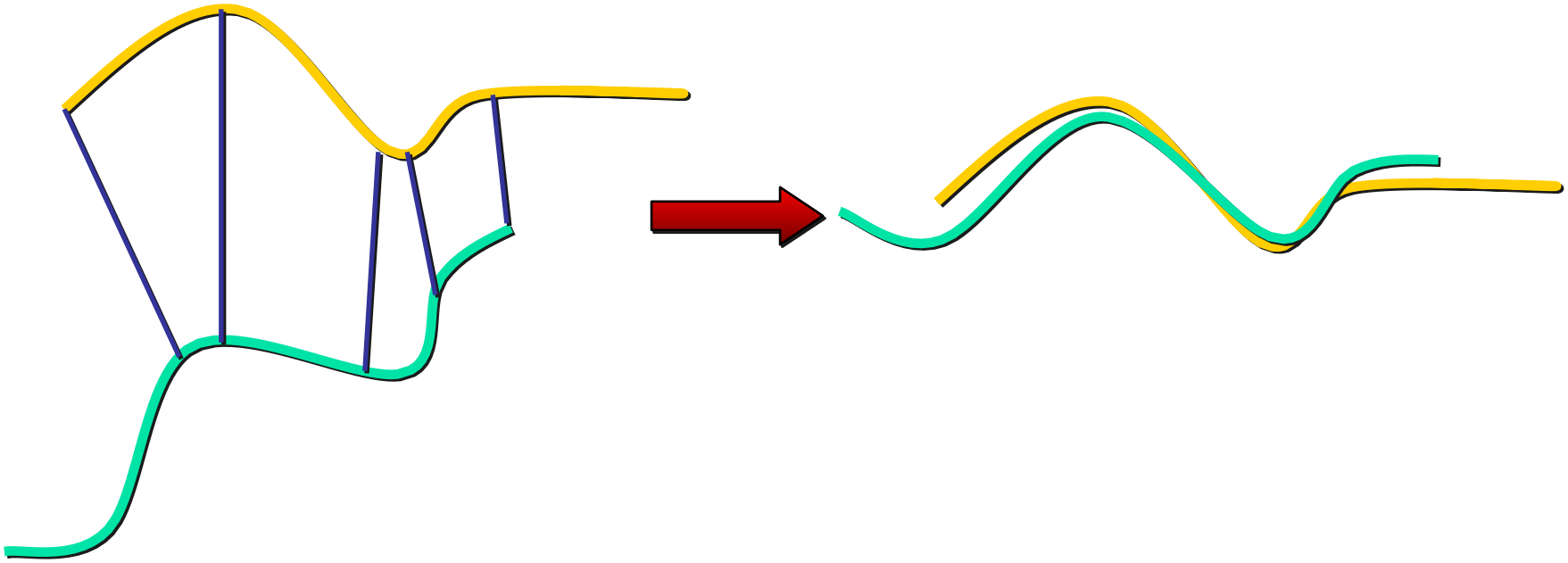


Trivial :
plus proche voisin (kd-tree par ex)

Alignement

2) Étant donnée une correspondance, trouver la transformation rigide (R,t) qui aligne au mieux les points correspondants

$$(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^2$$



Alignement

2) Étant donnée une correspondance, trouver la transformation rigide (R,t) qui aligne au mieux les points correspondants

$$(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^2$$

Théorème : la transformation optimale aligne les centroides.

$p=\sum_i w_i p_i / \sum_i w_i$

$q=\sum_i w_i q_i / \sum_i w_i$

centroides

$p_i'=p_i-p$

$q_i'=q_i-q$

recentrés

$\sum_i w_i p_i'=0$

$\sum_i w_i q_i'=0$

Alignement

2) Étant donnée une correspondance, trouver la transformation rigide (R,t) qui aligne au mieux les points correspondants

$$(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^2$$

Théorème : la transformation optimale aligne les centroides.

$$p=\sum_i w_i p_i / \sum_i w_i$$
$$q=\sum_i w_i q_i / \sum_i w_i$$

centroides

$$p_i'=p_i-p$$
$$q_i'=q_i-q$$

recentrés

$$\sum_i w_i p_i'=0$$
$$\sum_i w_i q_i'=0$$

$$\|R.p_i+t-q_i\|^2=\|(R.p_i'-q_i')+(R.p+t-q)\|^2$$
$$=\|(R.p_i'-q_i')\|^2+\|(R.p+t-q)\|^2+2(R.p_i'-q_i')^T.(R.p+t-q)$$

Alignement

2) Étant donnée une correspondance, trouver la transformation rigide (R,t) qui aligne au mieux les points correspondants

$$(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^2$$

Théorème : la transformation optimale aligne les centroides.

$p=\sum_i w_i p_i / \sum_i w_i$ centroides

$q=\sum_i w_i q_i / \sum_i w_i$ centroides

$p_i'=p_i-p$ recentrés

$q_i'=q_i-q$ recentrés

$\sum_i w_i p_i'=0$

$\sum_i w_i q_i'=0$

$$\|R.p_i+t-q_i\|^2=\|(R.p_i'-q_i')+(R.p+t-q)\|^2$$
$$=\|(R.p_i'-q_i')\|^2+\|(R.p+t-q)\|^2+2(R.p_i'-q_i')^T.(R.p+t-q)$$

$$\sum_i w_i\|R.p_i+t-q_i\|^2=\sum_i w_i(\|(R.p_i'-q_i')\|^2+\|(R.p+t-q)\|^2)+2(R.(\sum_i w_i p_i')-(\sum_i w_i q_i'))^T.(R.p+t-q)$$
$$=\sum_i w_i(\|(R.p_i'-q_i')\|^2+\|(R.p+t-q)\|^2)$$

Alignement

$$(R,t)=argmin \sum_i w_i(\|(R.p_i'-q_i')\|^2+\|(R.p+t-q)\|^2)$$

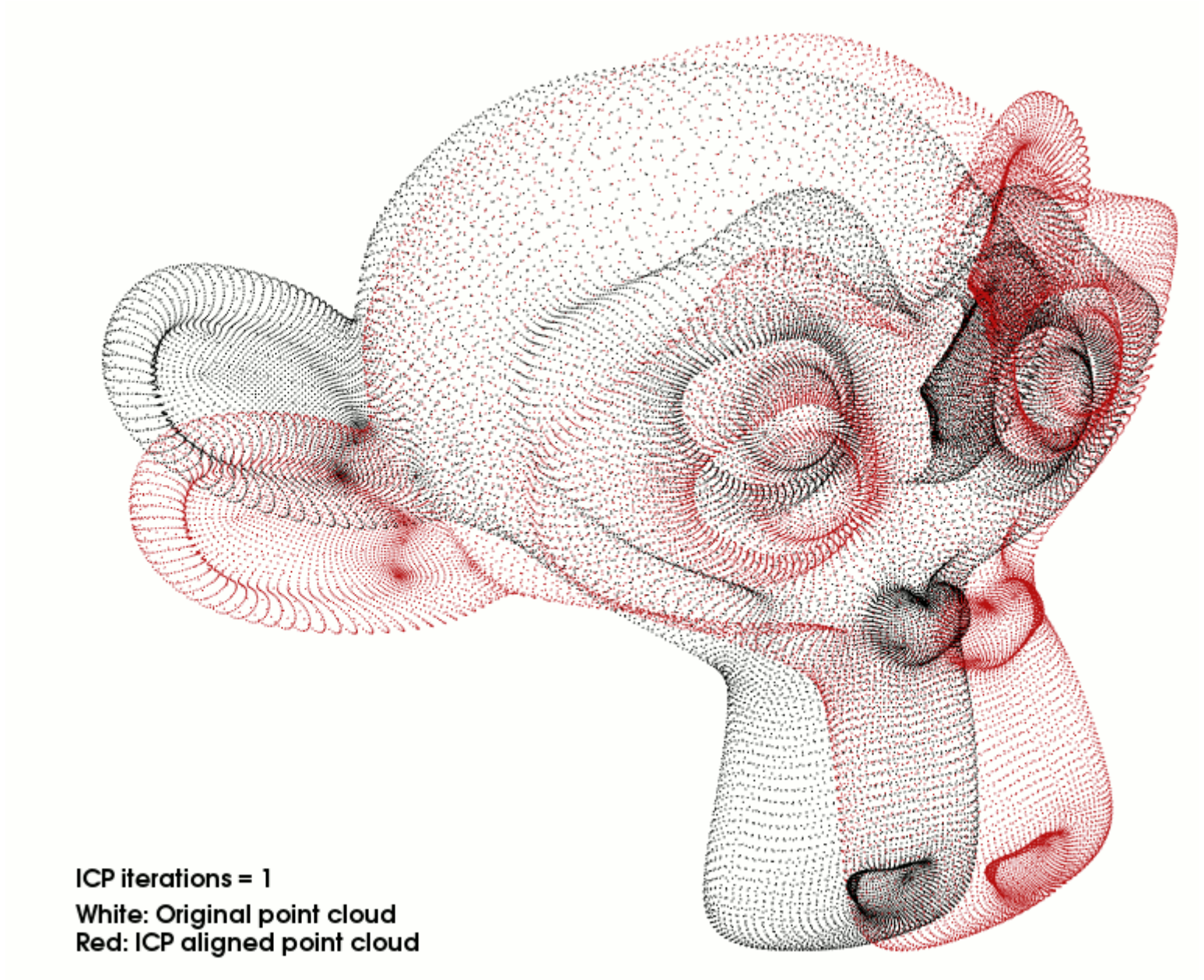
avec $p=\sum_i w_i p_i / \sum_i w_i$ $q=\sum_i w_i q_i / \sum_i w_i$ centroides, $p_i'=p_i-p$ $q_i'=q_i-q$ recentrés

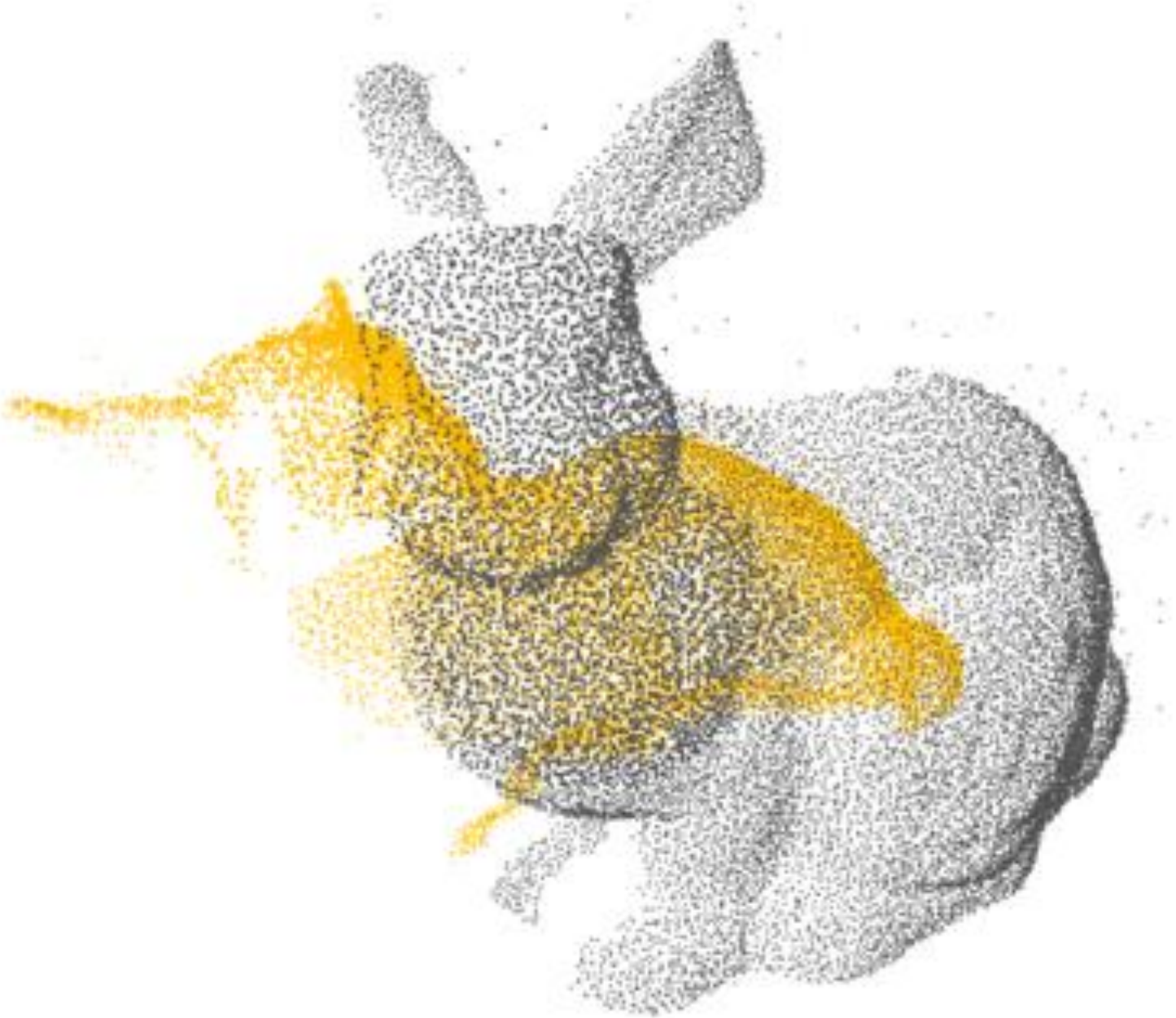
Procédure :

Trouver R qui minimise : $\sum_i w_i\|(R.p_i'-q_i')\|^2$ (Problème de Procustes)

$$\begin{cases} C=\sum_i w_i q_i'.p_i'^T \\ C=U.S.V^T \quad \text{(décomposition en valeurs singulières)} \\ R=U.diag(1,1,sign(U.V^T)).V^T \end{cases}$$

Puis t est donné par : $t=q-R.p$


ICP iterations = 1
White: Original point cloud
Red: ICP aligned point cloud

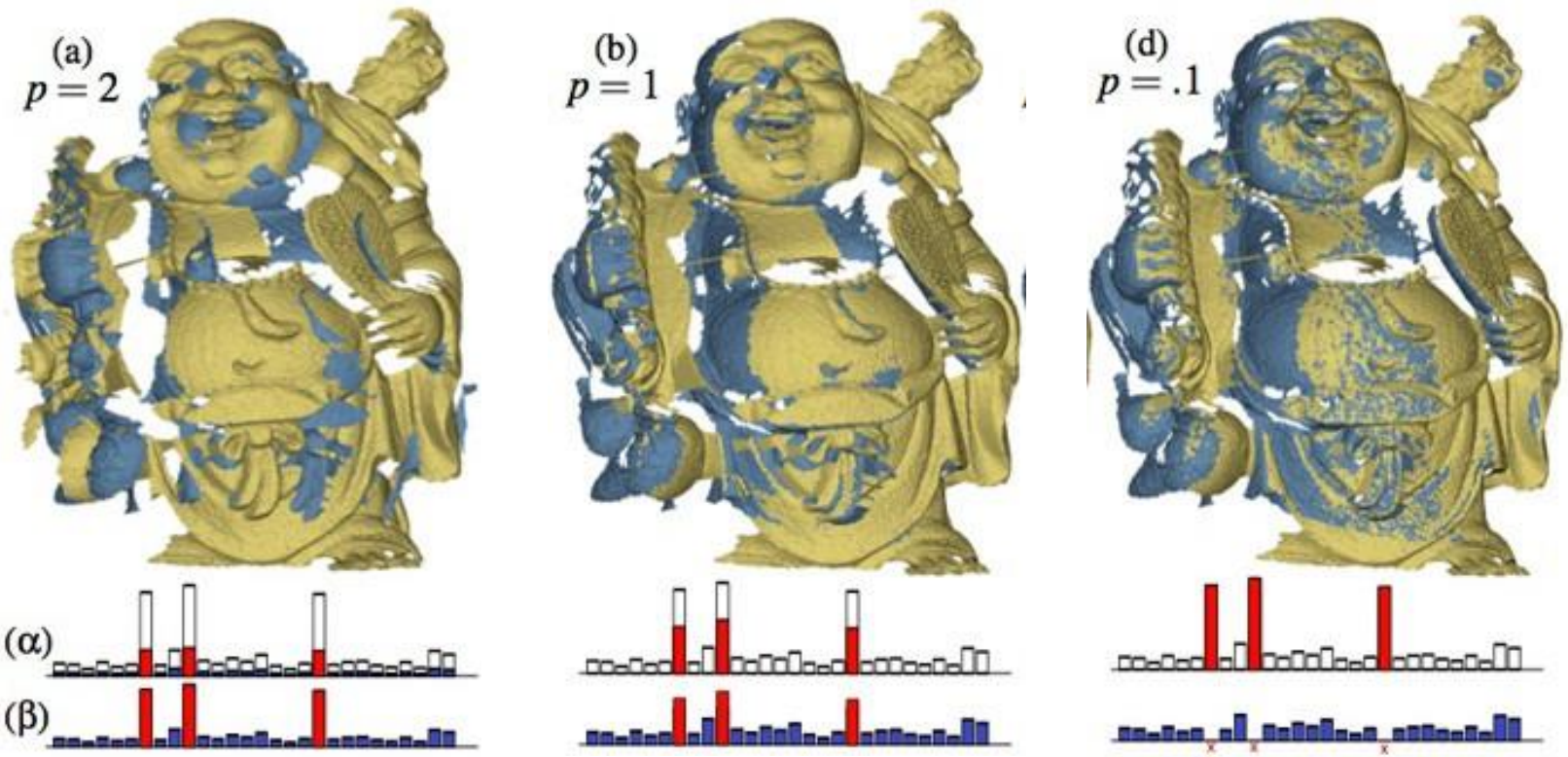


Variantes

$$(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^2$$

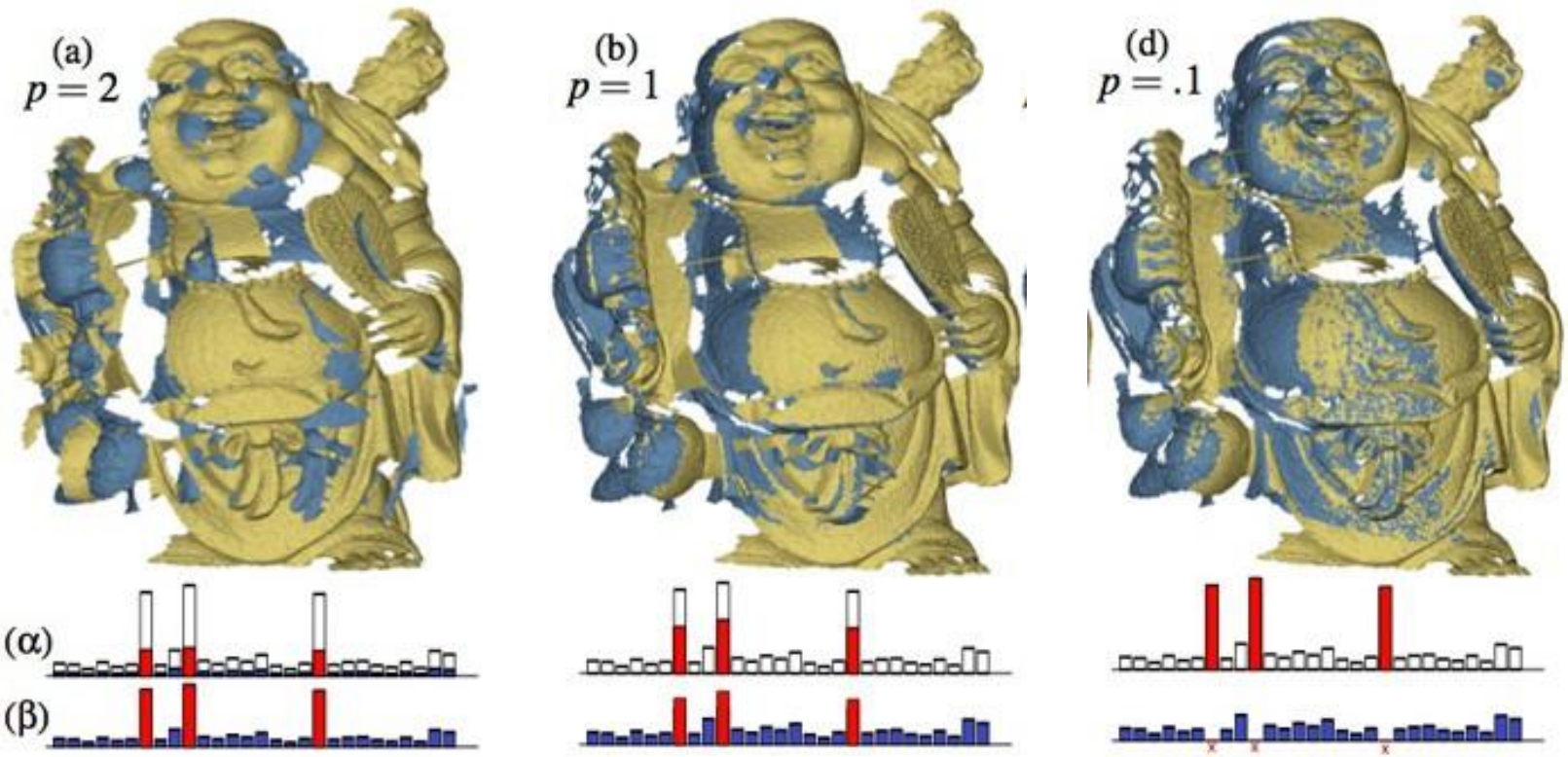
- Quels sont des poids pertinents ?
- Doit-on considérer tous les points de P ?
- Doit-on associer un point de Q nécessairement ?
- Peut-on aligner P sur Q ET Q sur P en même temps ?
- En présence d'outliers, peut-on ignorer certains points ?
  - RANSAC
  - Normes insensibles aux outliers : $(R,t)=argmin \sum_i w_i \|R.p_i+t-q_i\|^p$

Application : « Sparse » ICP



[Bouaziz et al. 2013] : Sparse Iterative Closest Point

Vidéo démo

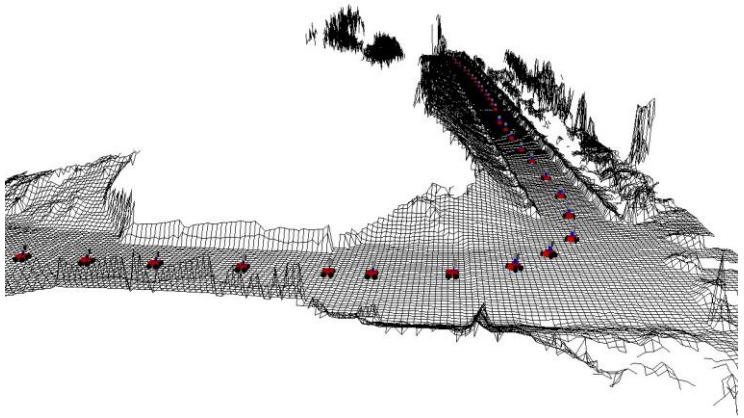


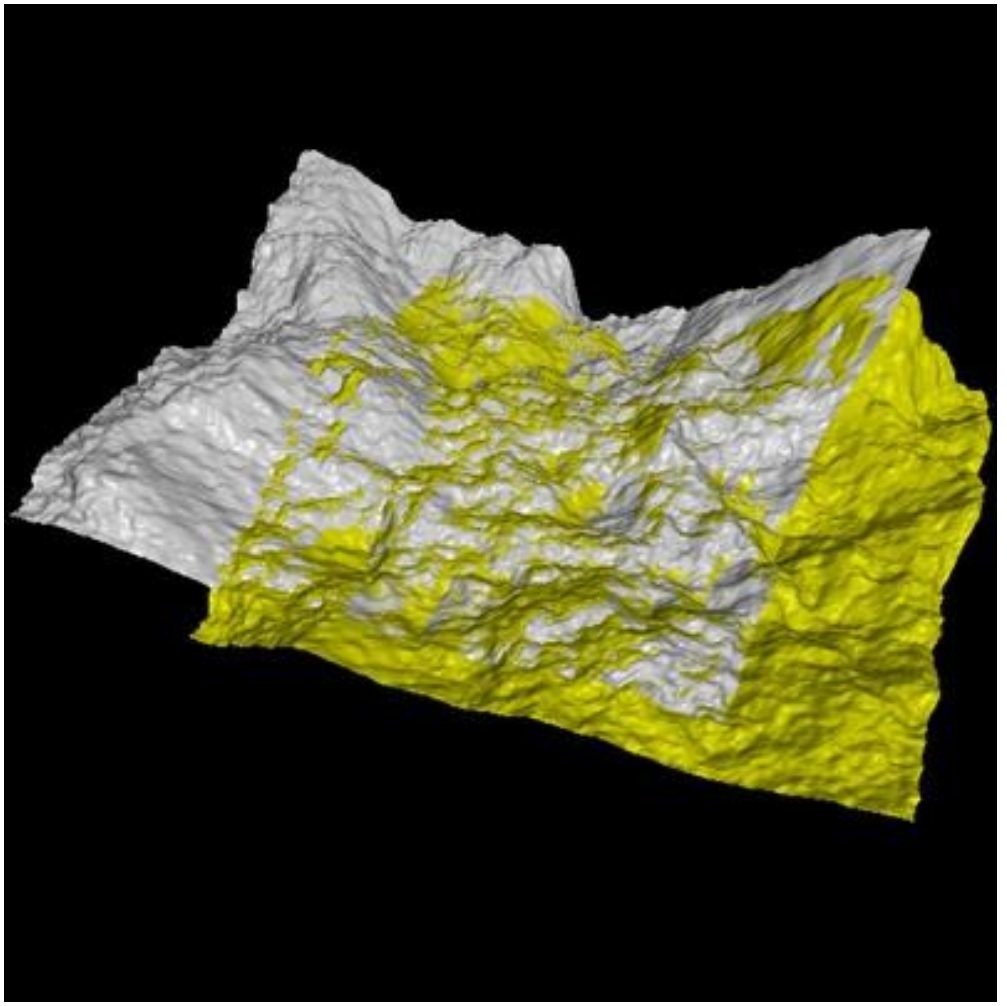
https://www.youtube.com/watch?v=ii2vHBwlmo8

Questions ?

Sources cours de Jean-Marc Thiery :
https://perso.telecom-paristech.fr/jthiery/
Roi Pooran ETH

Motivation





Goal: Find local transformation to align points

The Problem

- Given two corresponding point sets:

$$X = \{x_1, ..., x_{N_x}\}$$

$$P = \{p_1, ..., p_{N_p}\}$$

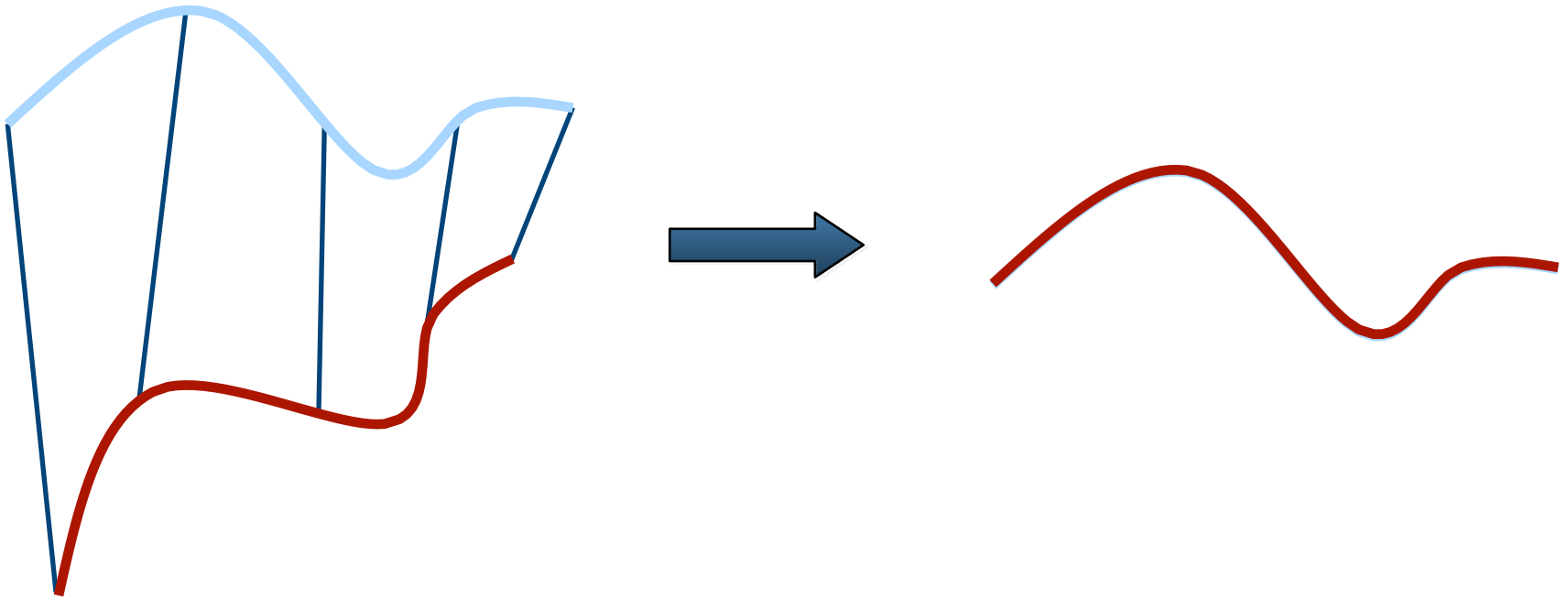
- Wanted: Translation *t* and rotation *R* that minimize the sum of the squared error:

$$E(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2$$

Where x_i and p_i are corresponding points

Key Idea

- If the correct correspondences are known, the correct relative rotation/translation can be calculated in **closed form**



Center of Mass

$$\mu_x = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i \quad \text{and} \quad \mu_p = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i$$

are the centers of mass of the two point sets

Idea:

- Subtract the corresponding center of mass from every point in the two point sets before calculating the transformation
- The resulting point sets are:

$$X' = \{x_i - \mu_x\} = \{x'_i\}$$
$$P' = \{p_i - \mu_p\} = \{p'_i\}$$

and

Singular Value Decomposition

Let $W = \sum_{i=1}^{N_p} x'_i p'^T_i$

denote the singular value decomposition (SVD) of W by:

$$W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

where $U, V \in \mathbb{R}^{3\times3}$ are unitary, and

$\sigma_1 \geq \sigma_2 \geq \sigma_3$ are the singular values of W

SVD

Theorem (without proof):

If rank(*W*) = 3, the optimal solution of E(*R*,*t*) is unique and is given by:

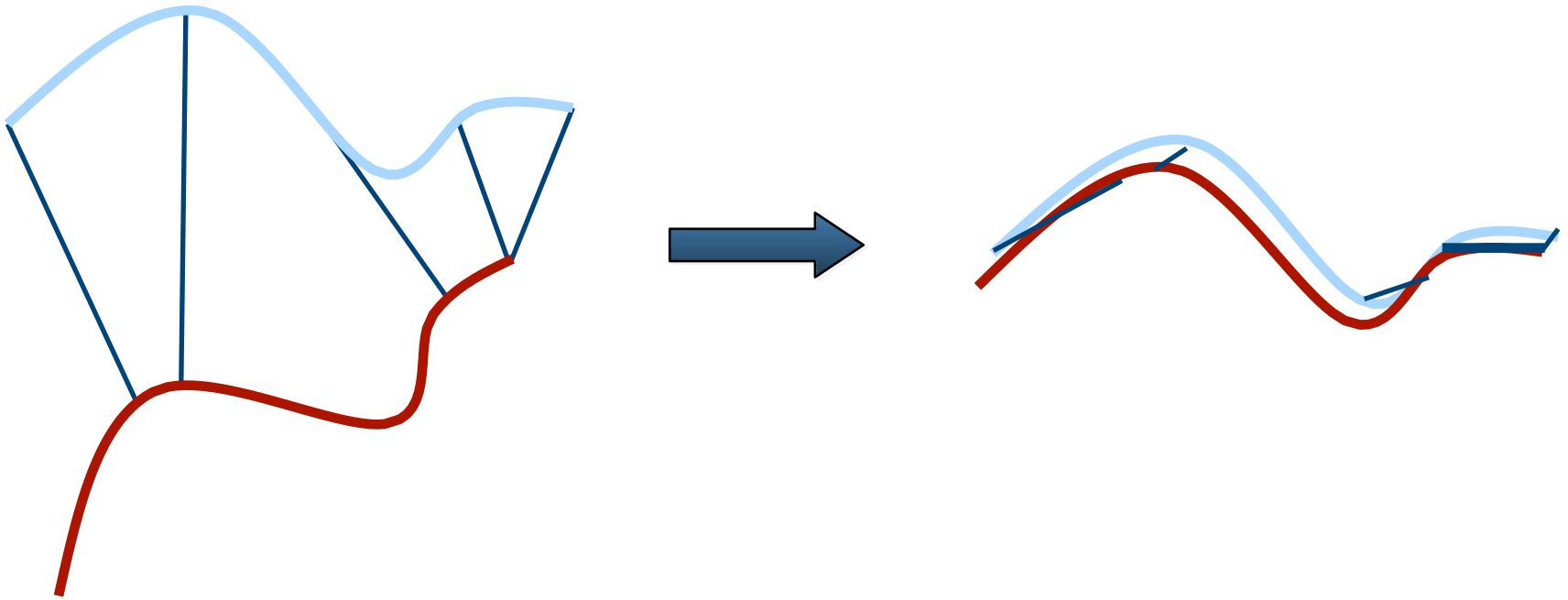
$$R = UV^T$$
$$t = \mu_x - R\mu_p$$

The minimal value of error function at (*R*,*t*) is:

$$E(R,t) = \sum_{i=1}^{N_p} (||x_i'||^2 + ||y_i'||^2) - 2(\sigma_1 + \sigma_2 + \sigma_3)$$

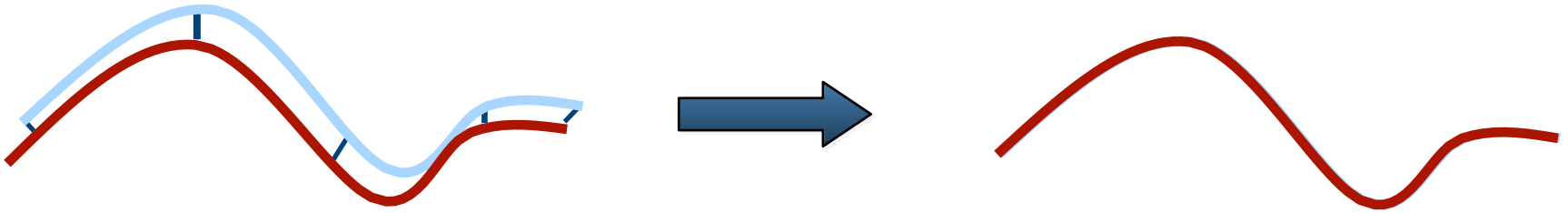
ICP with Unknown Data Association

- If the correct correspondences are **not known**, it is generally impossible to determine the optimal relative rotation/ translation in one step



Iterative Closest Point (ICP) Algorithm

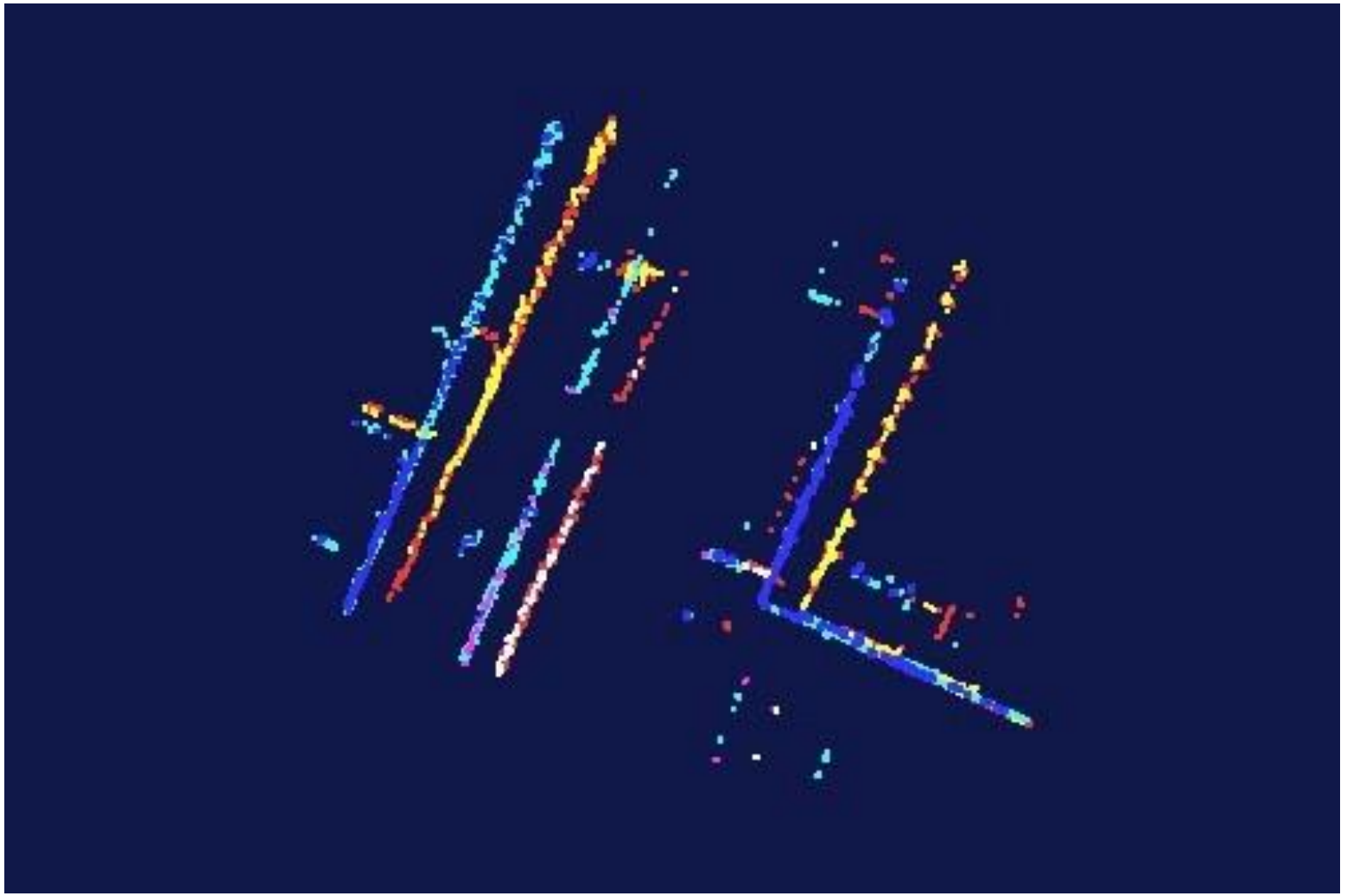
- Idea: Iterate to find alignment
- Iterative Closest Points
  [Besl & McKay 92]
- Converges if starting positions are "close enough"



Basic ICP Algorithm

- Determine corresponding points
- Compute rotation R, translation t via SVD
- Apply R and t to the points of the set to be registered
- Compute the error $E(R,t)$
- If error decreased and error > threshold
  - Repeat these steps
  - Stop and output final alignment, otherwise

ICP Example



ICP Variants

Variants on the following stages of ICP have been proposed:

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Performance of Variants

- Various aspects of performance:
  - Speed
  - Stability (local minima)
  - Tolerance wrt. noise and outliers
  - Basin of convergence
    (maximum initial misalignment)

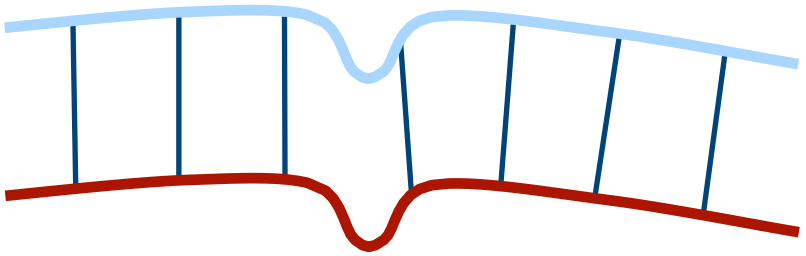
ICP Variants

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

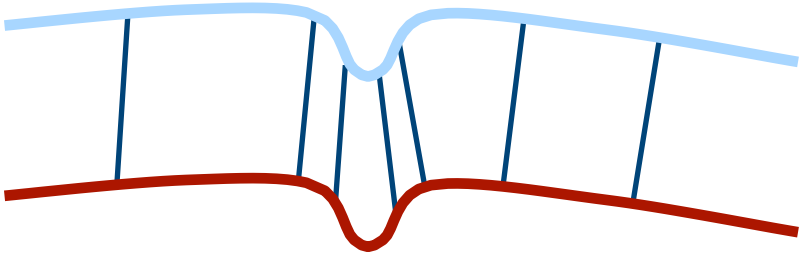
Selecting Source Points

- Use all points
- Uniform sub-sampling
- Random sampling
- Feature based sampling
- Normal-space sampling
  (Ensure that samples have normals distributed as uniformly as possible)

Normal-Space Sampling



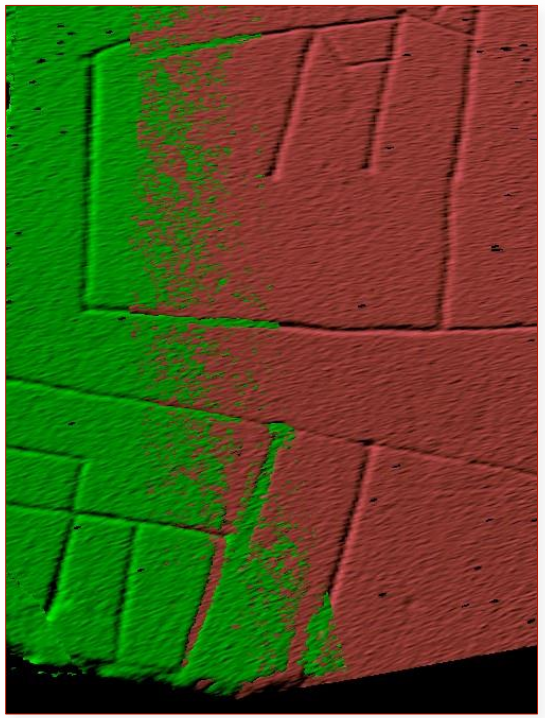
uniform sampling



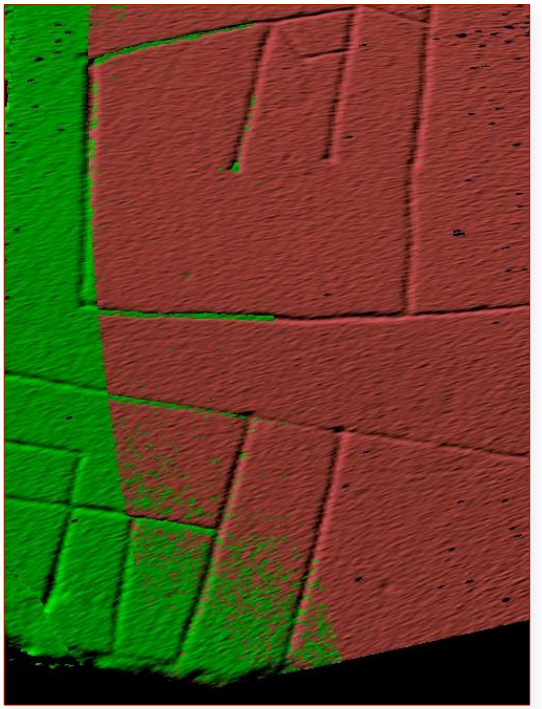
normal-space sampling

Comparison

- Normal-space sampling better for mostly smooth areas with sparse features
  [Rusinkiewicz et al., 01]



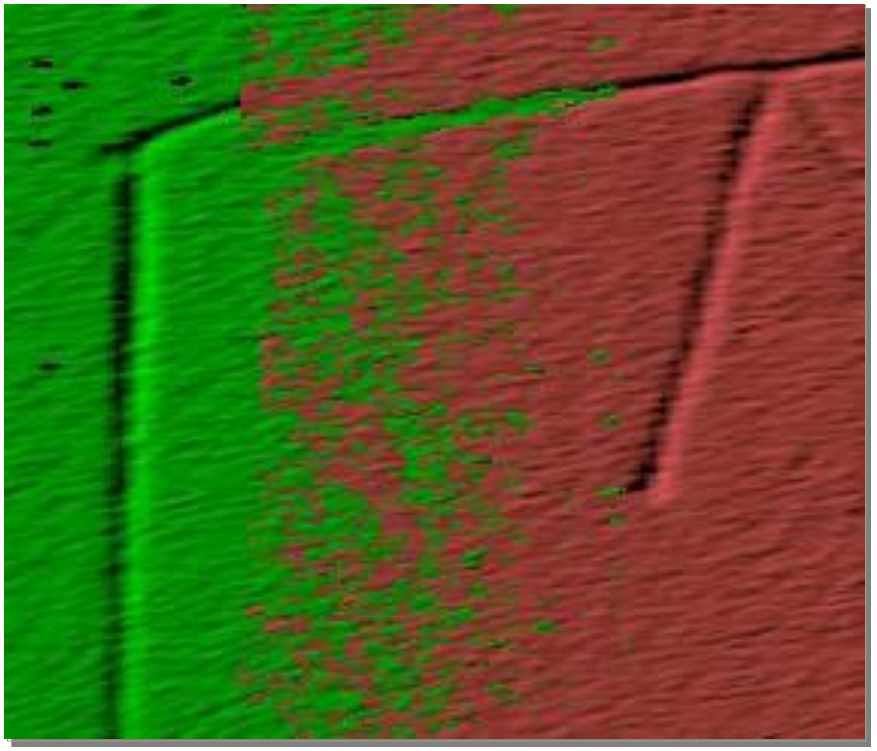
Random sampling



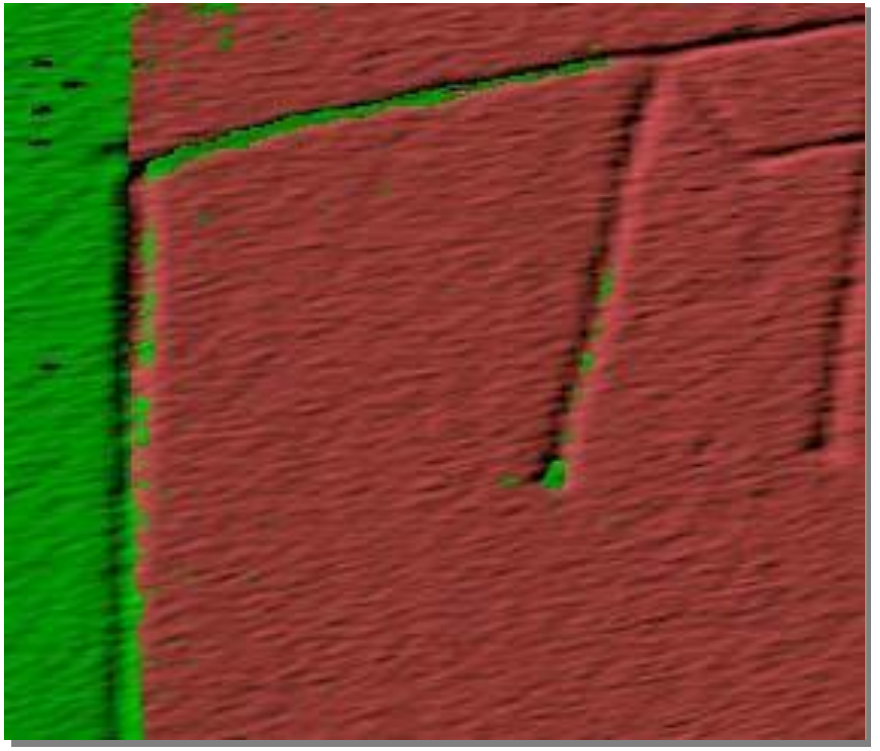
Normal-space sampling

Comparison

- Normal-space sampling better for mostly smooth areas with sparse features
  [Rusinkiewicz et al., 01]



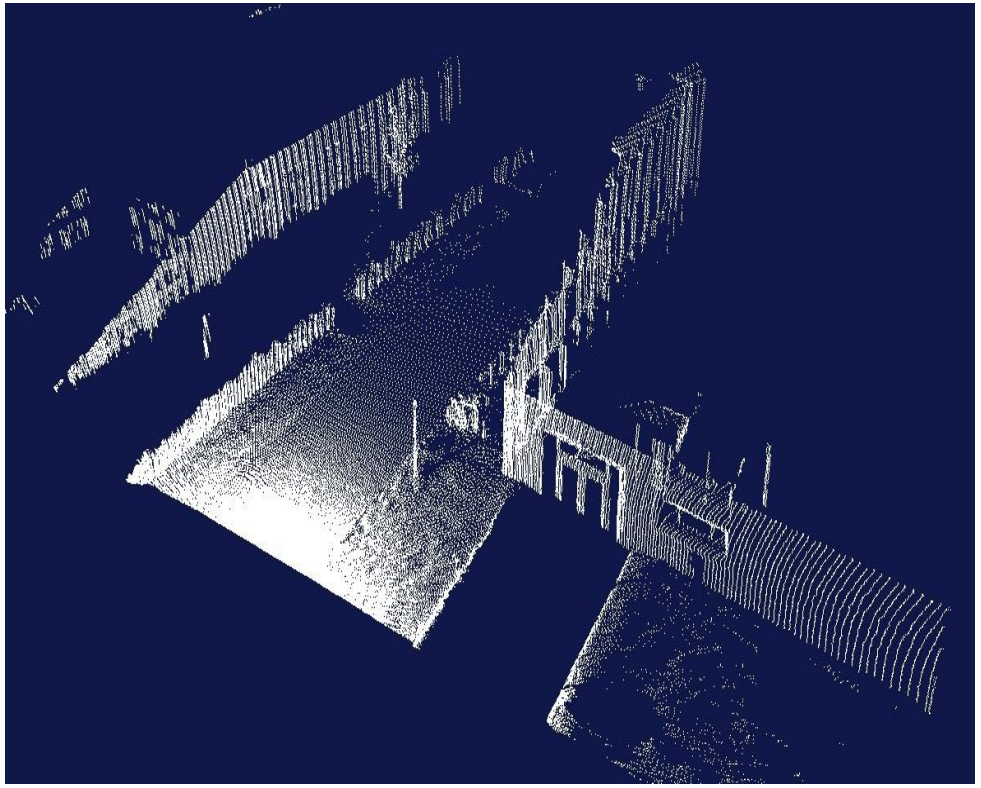
Random sampling



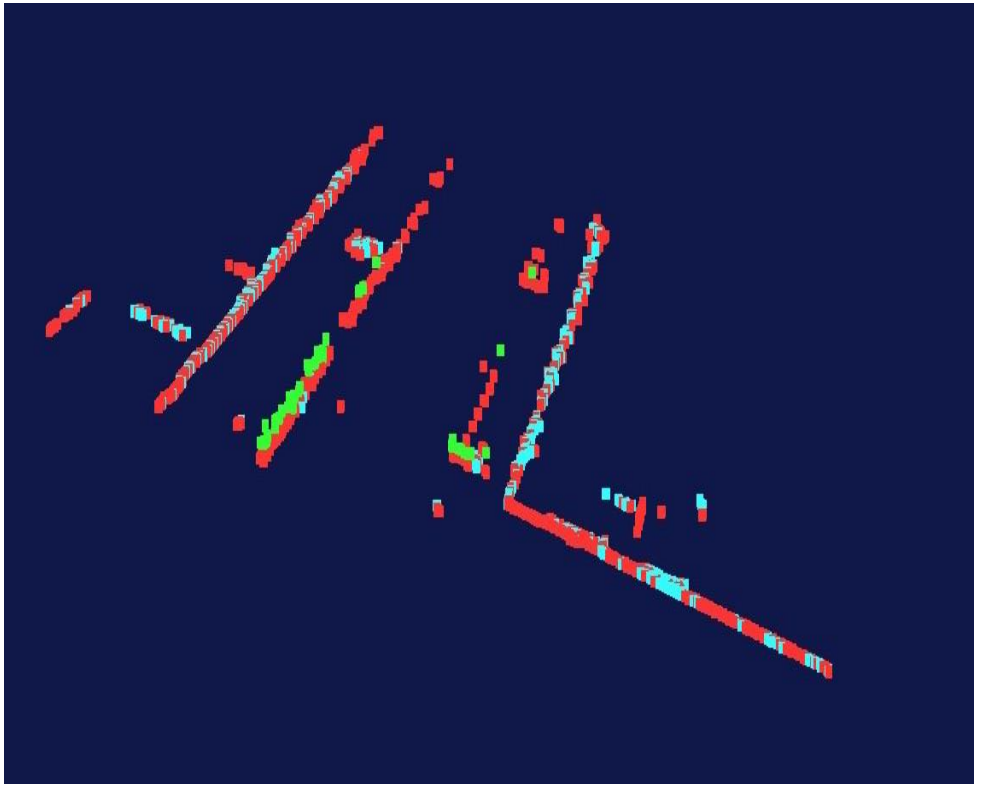
Normal-space sampling

Feature-Based Sampling

- Try to find “important” points
- Decreases the number of correspondences to find
- Higher efficiency and higher accuracy
- Requires preprocessing

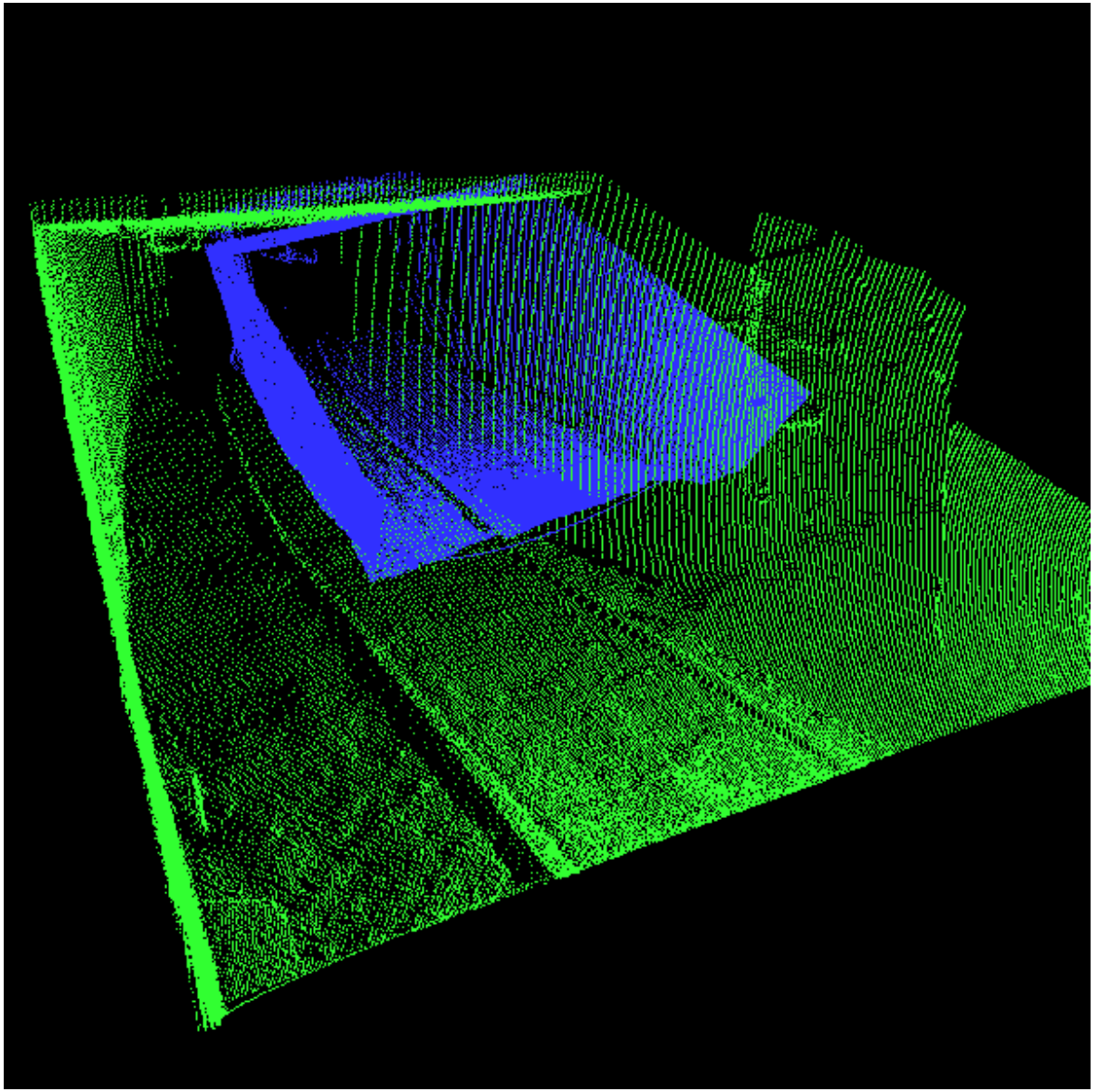


3D Scan (~200.000 Points)



Extracted Features (~5.000 Points)

ICP Application (With Uniform Sampling)



[Nuechter et al., 04]

ICP Variants

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Weighting

- Select a set of points for each set
- Match the selected points of the two sets
- **Weight the corresponding pairs**
- E.g., assign lower weights for points with higher point-point distances
- Determine transformation that minimizes the error function

ICP Variants

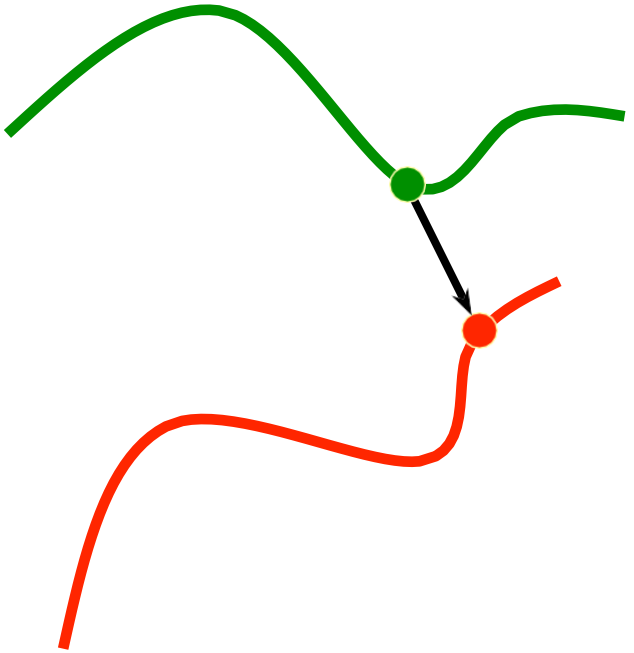
1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Data Association

- Has greatest effect on convergence and speed
- Matching methods:
  - Closest point
  - Normal shooting
  - Closest compatible point
  - Projection-based

Closest-Point Matching

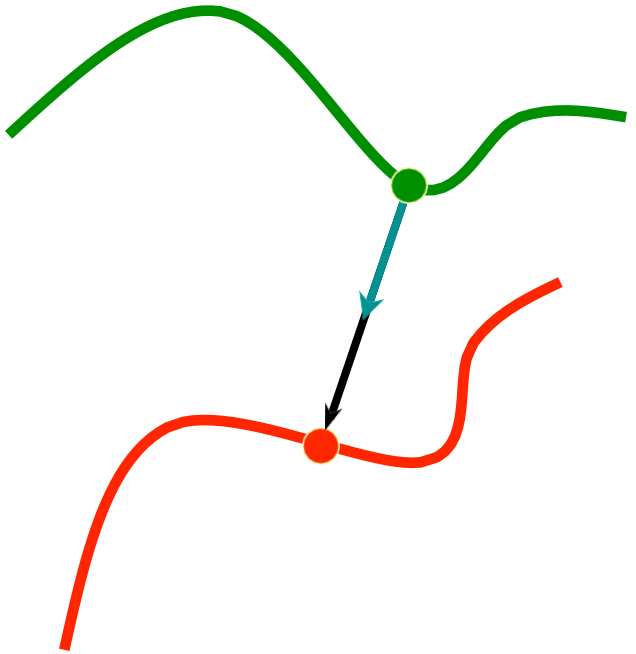
- Find closest point in other the point set (using kd-trees)



Generally stable, but slow convergence and requires preprocessing

Normal Shooting

- Project along normal, intersect other point set



Slightly better convergence results than closest point for smooth structures, worse for noisy or complex structures

Closest Compatible Point

- Improves the two previous variants by considering the **compatibility** of the points
- Only match compatible points
- Compatibility can be based on
  - Normals
  - Colors
  - Curvature
  - Higher-order derivatives
  - Other local features

Point-to-Plane Error Metric

- Minimize the sum of the squared distance between a point and the tangent plane at its correspondence point [Chen & Medioni 91]

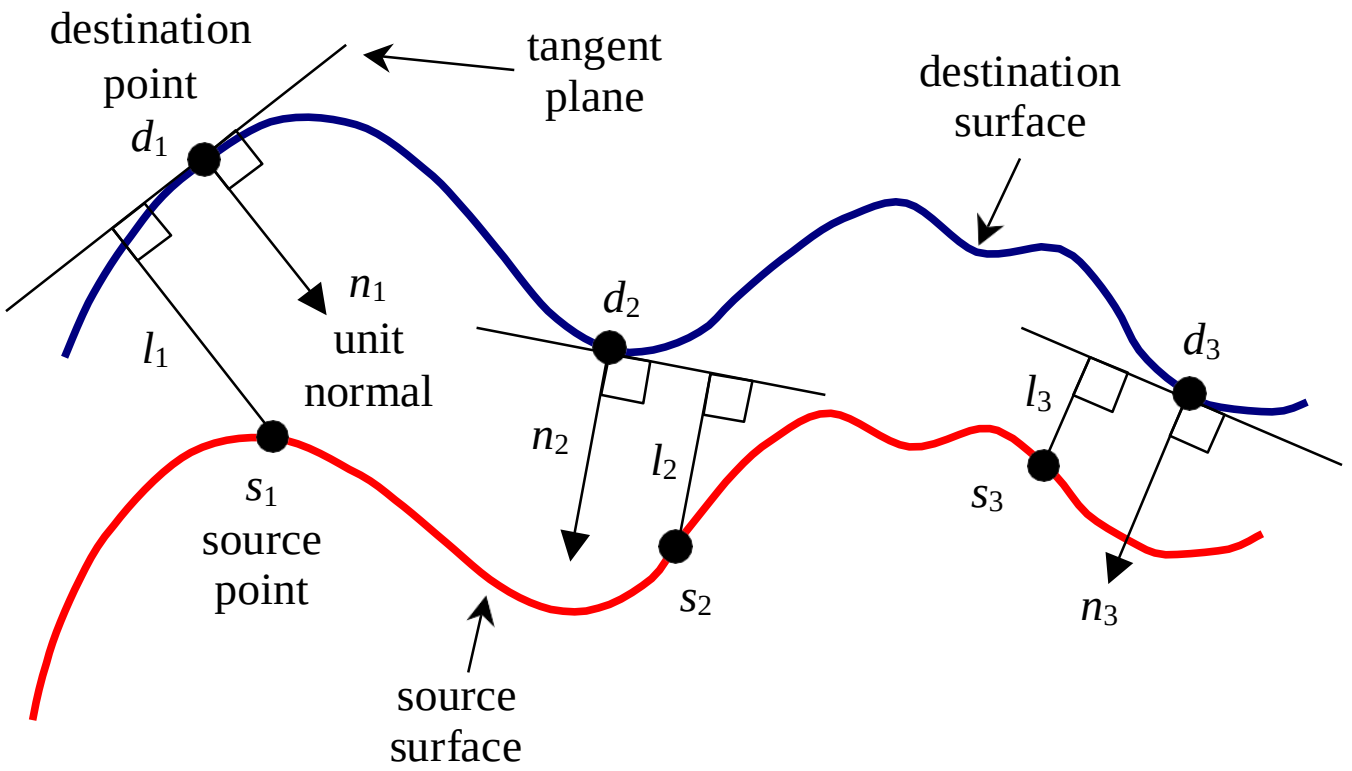


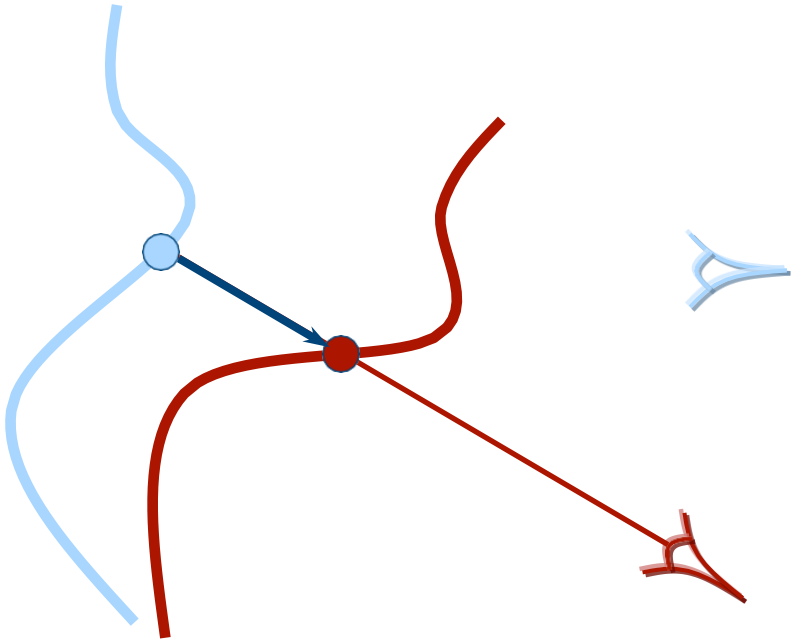
image from [Low 04]

Point-to-Plane Error Metric

- Solved using standard nonlinear least squares methods (e.g., Levenberg-Marquardt method [Press92]).
- Each iteration generally slower than the point-to-point version, however, often significantly better convergence rates [Rusinkiewicz01]
- Using point-to-plane distance instead of point-to-point lets flat regions slide along each other [Chen & Medioni 91]

Projection

- Finding the closest point is the most expensive stage of the ICP algorithm
- Idea: Simplified nearest neighbor search
- For range images, one can project the points according to the view-point [Blais 95]



Projection-Based Matching

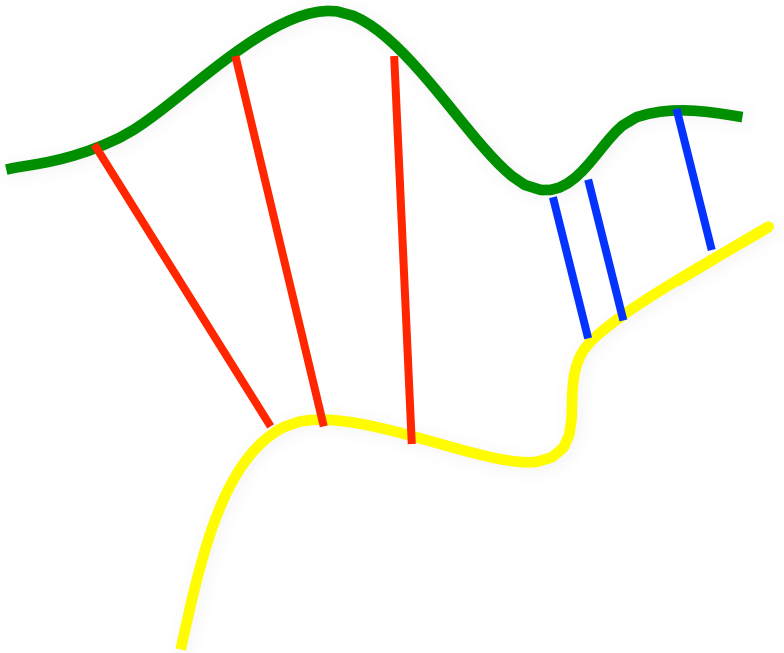
- Constant time
- Does not require precomputing a special data structure
- Requires point-to-plane error metric
- Slightly worse alignments per iteration

ICP Variants

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

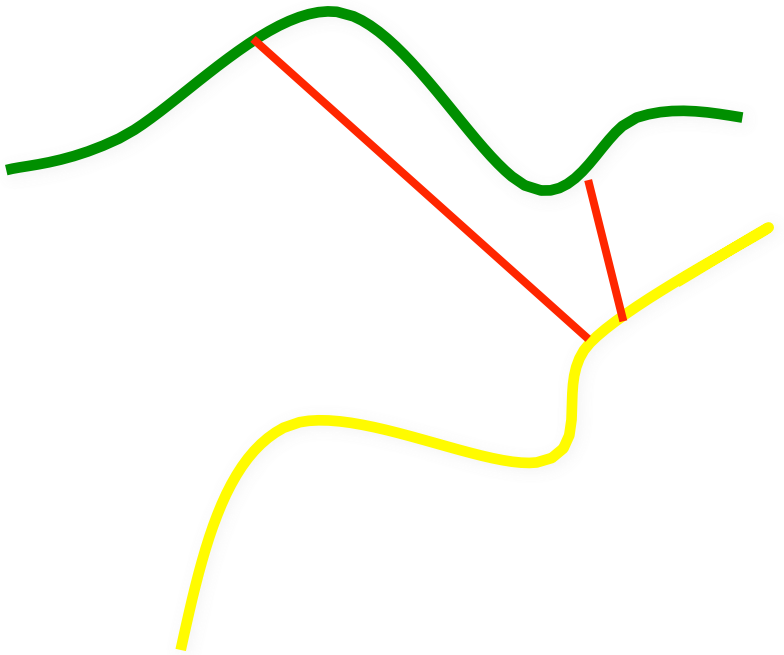
Rejecting (Outlier) Point Pairs

- Corresponding points with point to point distance higher than a given threshold



Rejecting (Outlier) Point Pairs

- Corresponding points with point to point distance higher than a given threshold
- Rejection of pairs that are not consistent with their neighboring pairs [Dorai 98]



Rejecting (Outlier) Point Pairs

- Corresponding points with point to point distance higher than a given threshold
- Rejection of pairs that are not consistent with their neighboring pairs [Dorai 98]
- Sort all correspondences with respect to their error and delete the worst *t*%, Trimmed ICP (TrICP) [Chetverikov et al. 02]
  - *t* is used to estimate the overlap
  - Problem: Knowledge about the overlap is necessary or has to be estimated

Summary: ICP Algorithm

- Potentially sample Points
- Determine corresponding points
- Potentially weight / reject pairs
- Compute rotation R, translation t (e.g. SVD)
- Apply R and t to all points of the set to be registered
- Compute the error $E(R,t)$
- If error decreased and error > threshold
  - Repeat to determine correspondences etc.
  - Stop and output final alignment, otherwise

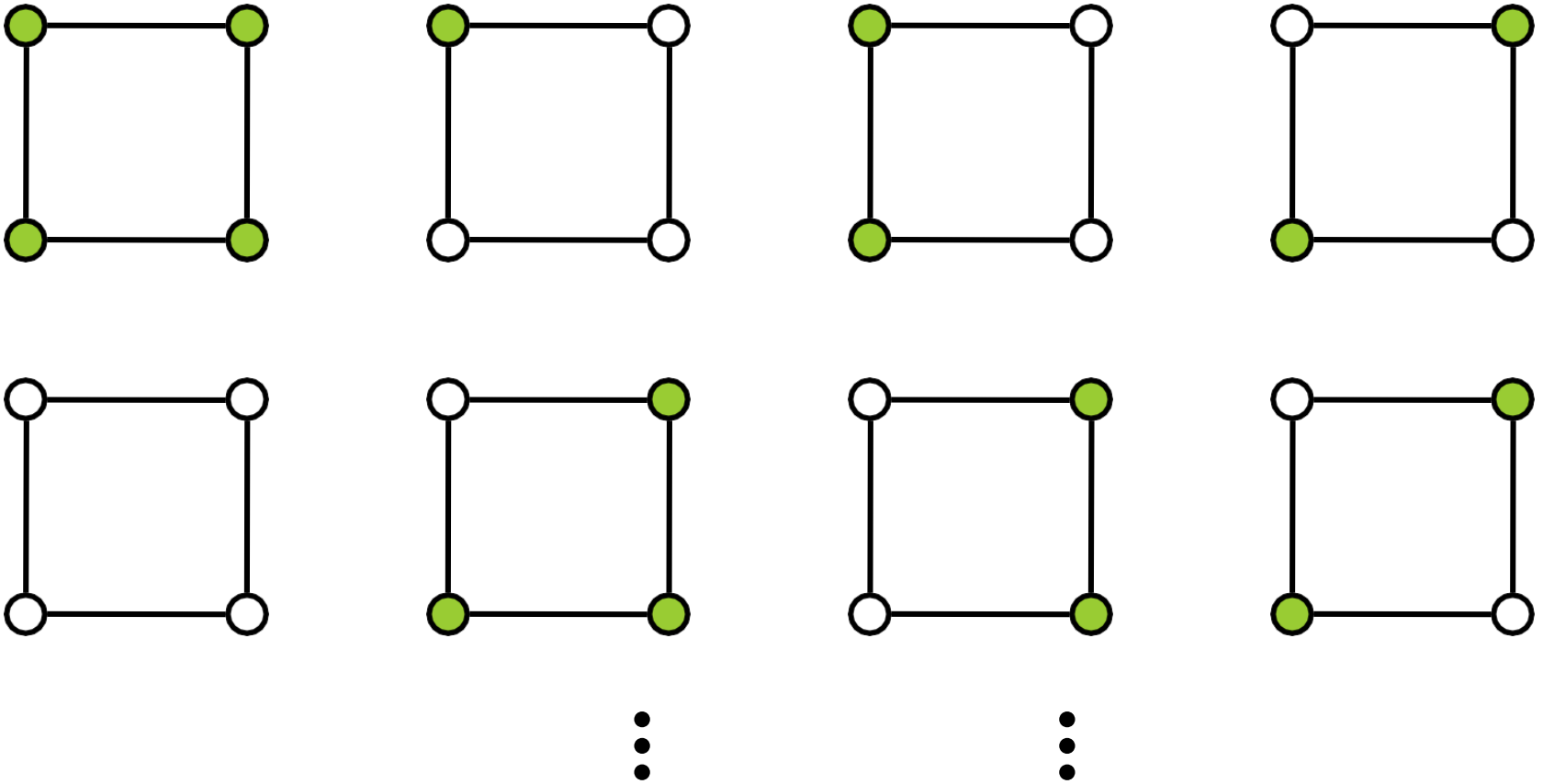
ICP Summary

- ICP is a powerful algorithm for calculating the displacement between scans
- The major problem is to determine the correct data associations
- Convergence speed depends on point matchings
- Given the correct data associations, the transformation can be computed efficiently using SVD

Marching Squares

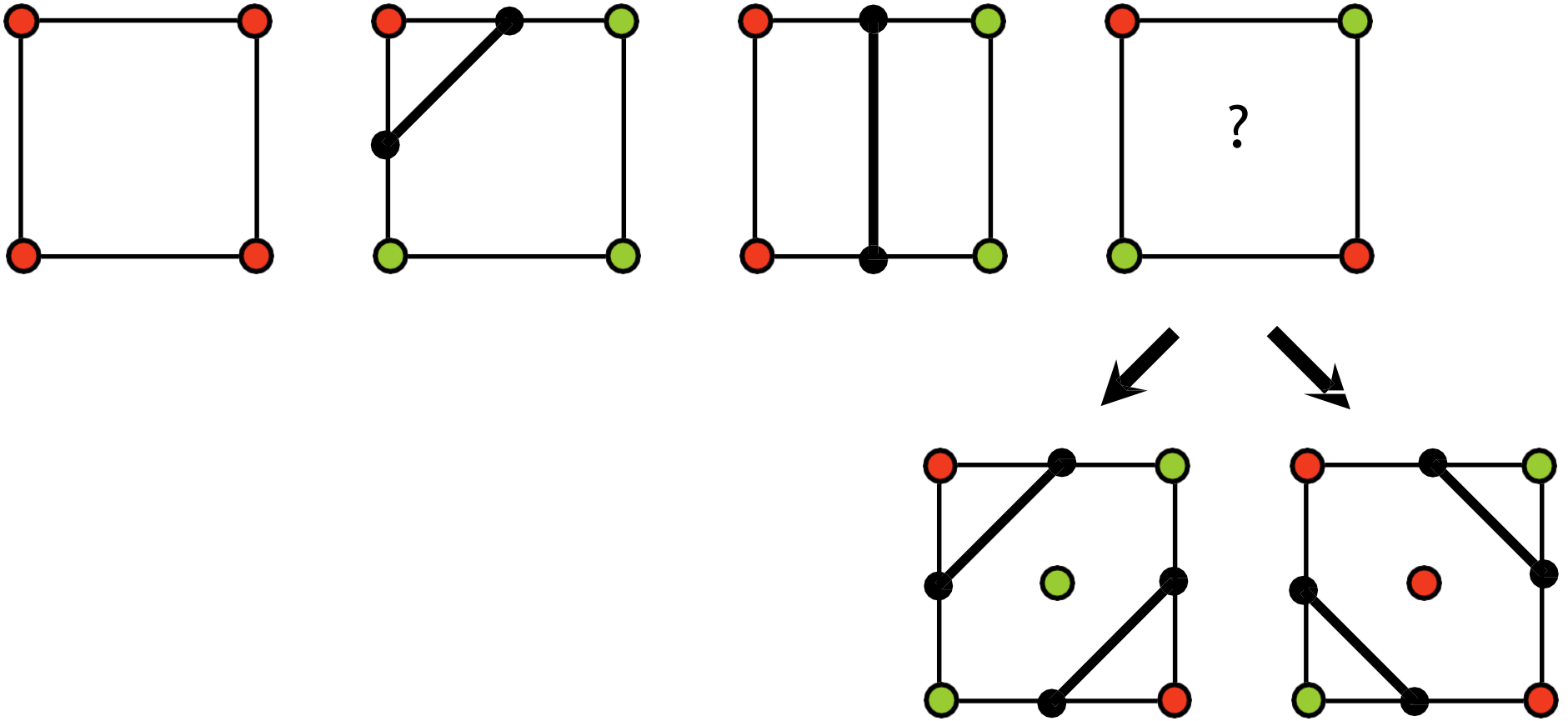
16 configurations différentes configurations en 2D

4 classes (rotation, reflection, negation)



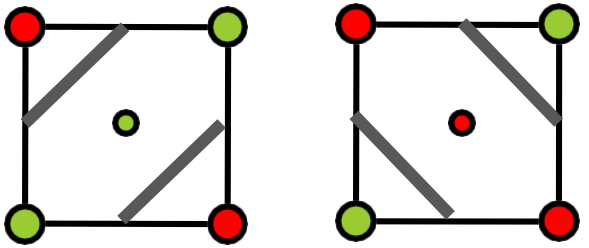
Tessellation in 2D

4 classes (rotation, reflection, negation)

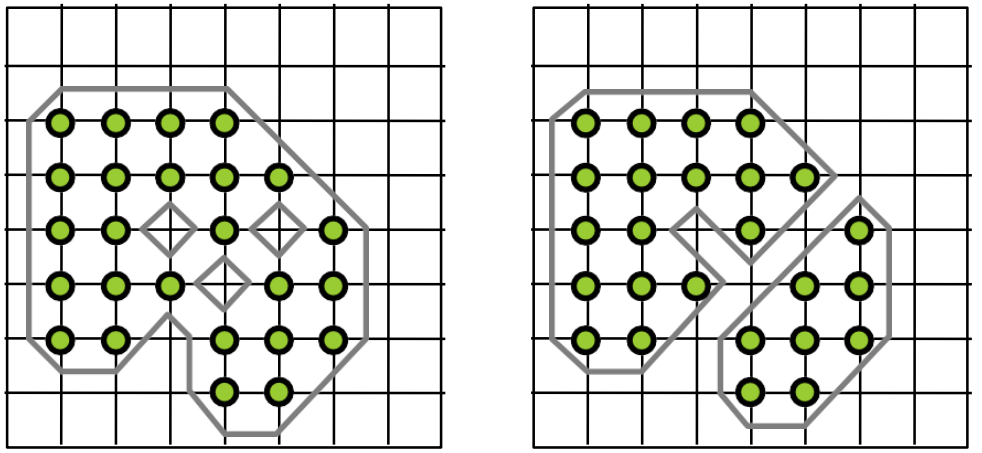


Tessellation in 2D

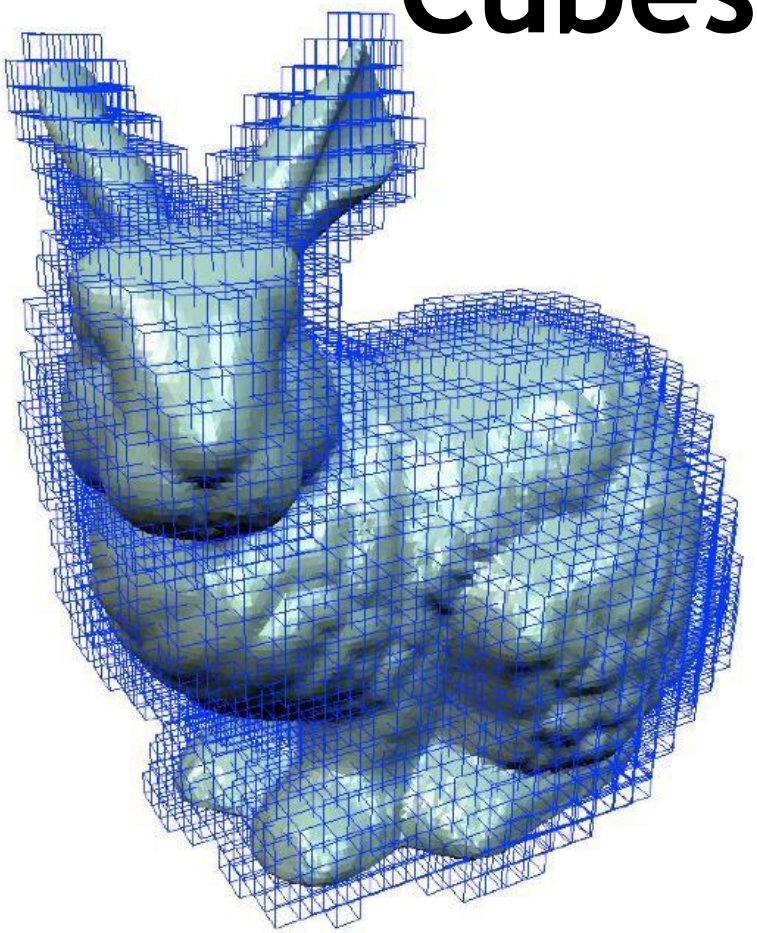
Case 4 is ambiguious:

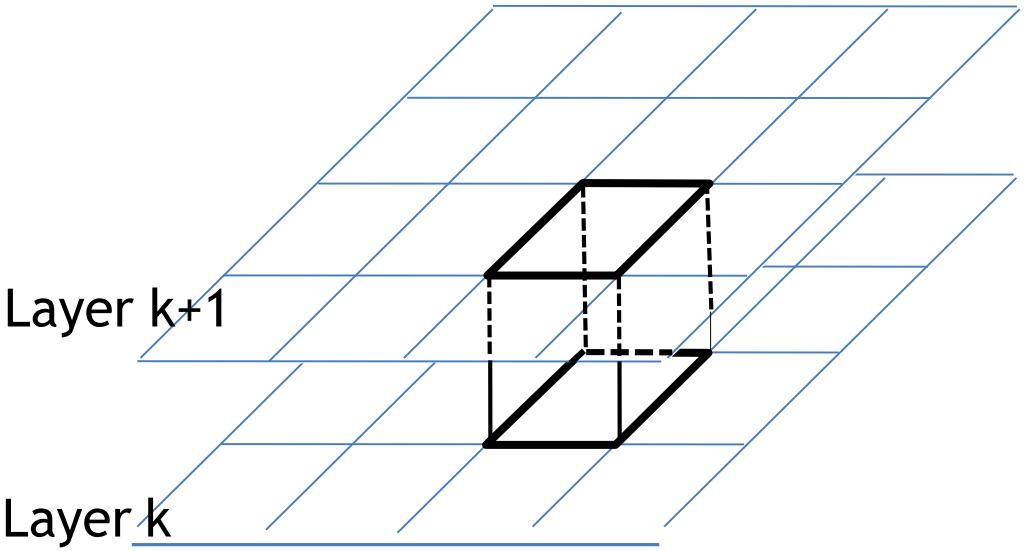


Always pick consistently



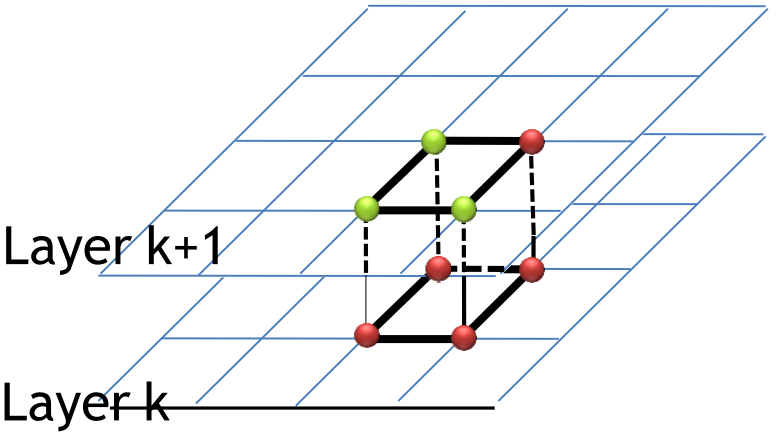
3D: Marching Cubes





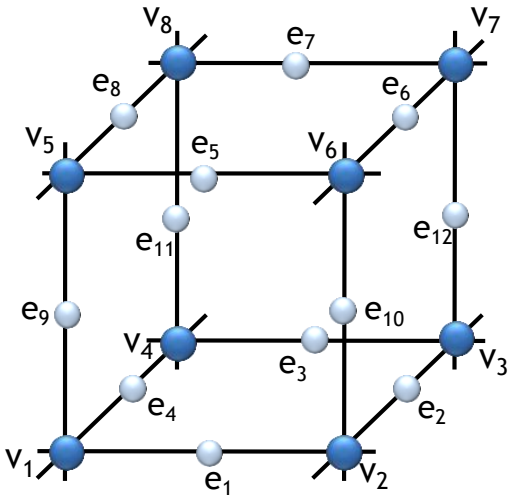
Marching Cubes

- Marching Cubes (Lorensen and Cline 1987)
  1. Load 4 layers of the grid into memory
  2. Create a cube whose vertices lie on the two middle layers
  3. Classify the vertices of the cube according to the implicit function (inside, outside or on the surface)

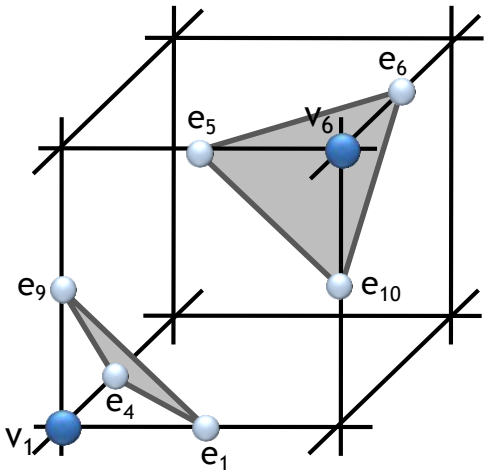


Marching Cubes

4. Compute case index. We have $2^8 = 256$ cases (0/1 for each of the eight vertices) – can store as 8 bit (1 byte) index.



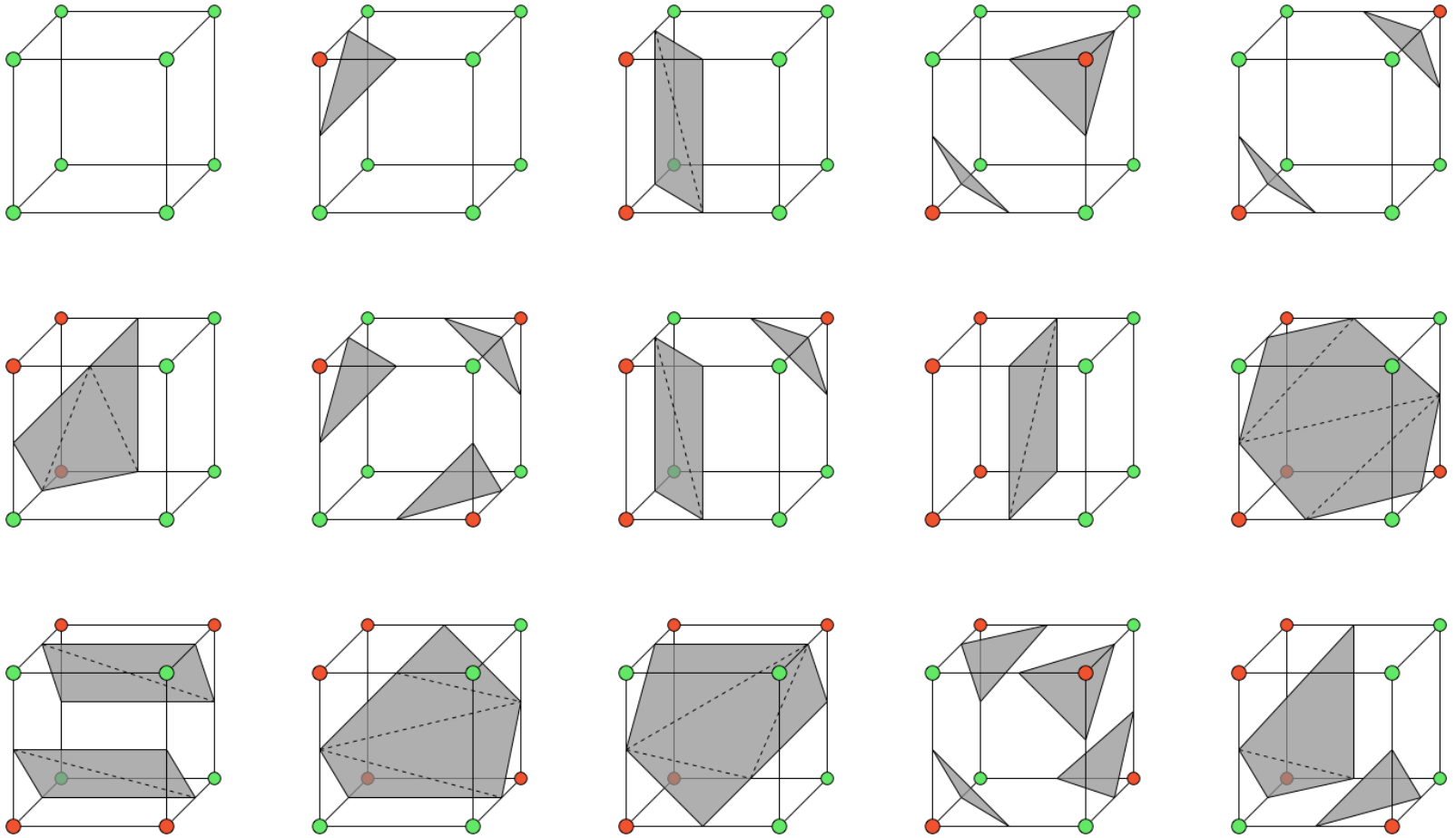
index = | v_1 | v_2 | v_3 | v_4 | v_5 | v_6 | v_7 | v_8 |



index = | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | = 33

Marching Cubes

- Unique cases (by rotation, reflection and negation)

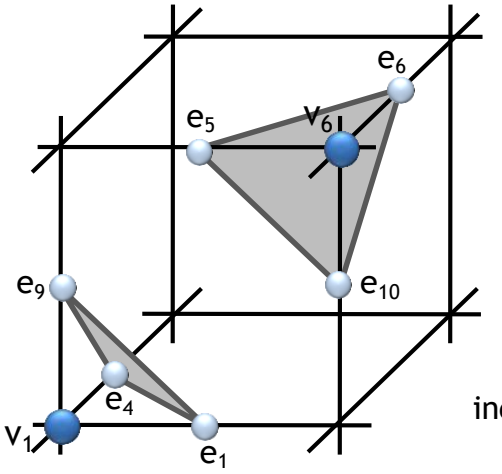


Tessellation

3D – Marching Cubes

5. Using the case index, retrieve the connectivity in the look-up table

- Example: the entry for index 33 in the look-up table indicates that the cut edges are e_1; e_4; e_5; e_6; e_9 and e_{10} ; the output triangles are (e_1; e_9; e_4) and (e_5; e_{10}; e_6).



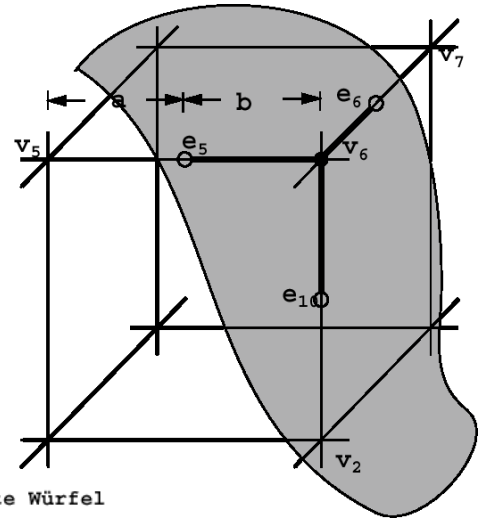
index = 0 0 1 0 0 0 0 1 = 33

Marching Cubes

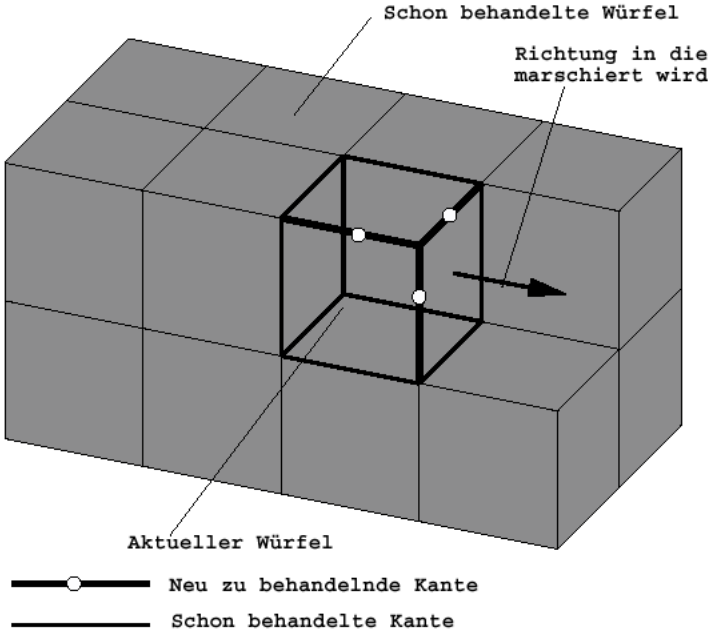
6. Calculer la position des sommets par interpolation

$$\mathbf{v}_s = t\mathbf{v}_a + (1-t)\mathbf{v}_b$$

$$t = \frac{F(\mathbf{v}_b)}{F(\mathbf{v}_b) - F(\mathbf{v}_a)}$$

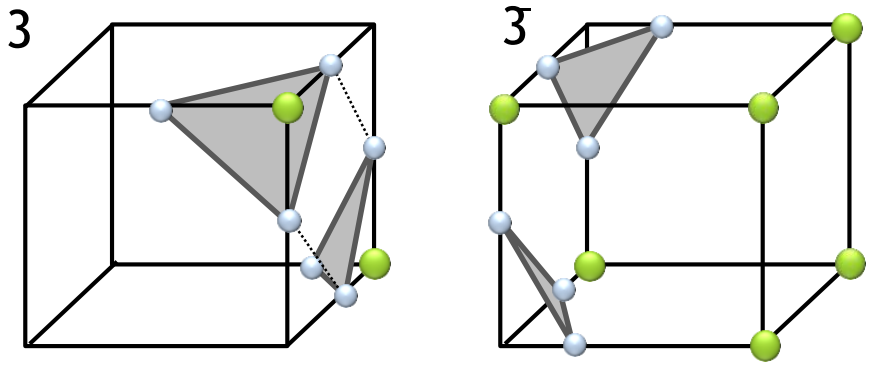


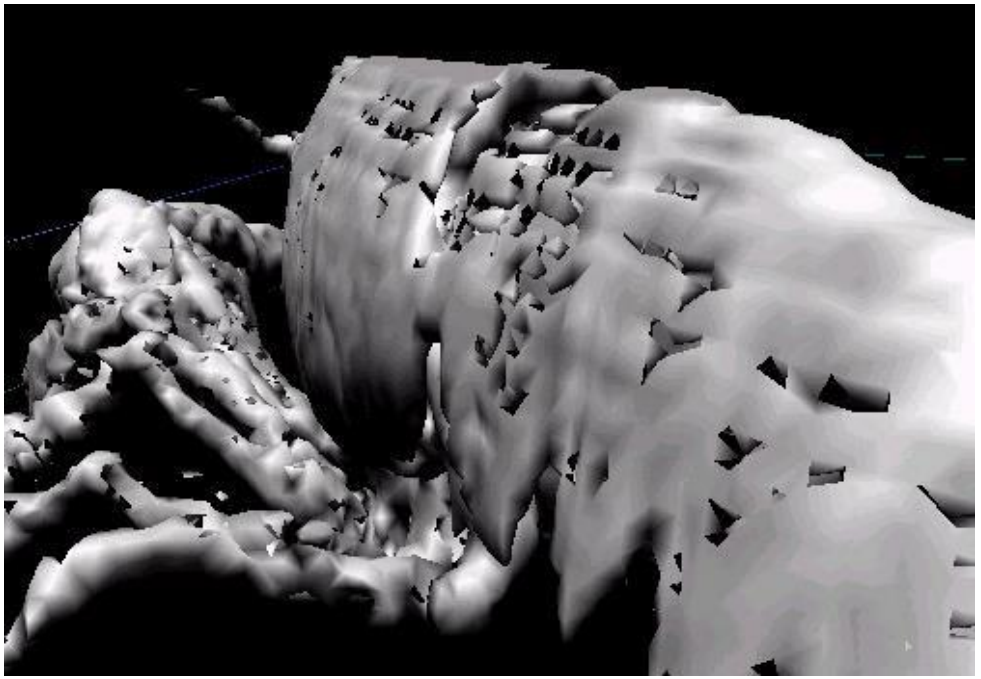
7. Passer au cube suivant



Marching Cubes – Problems

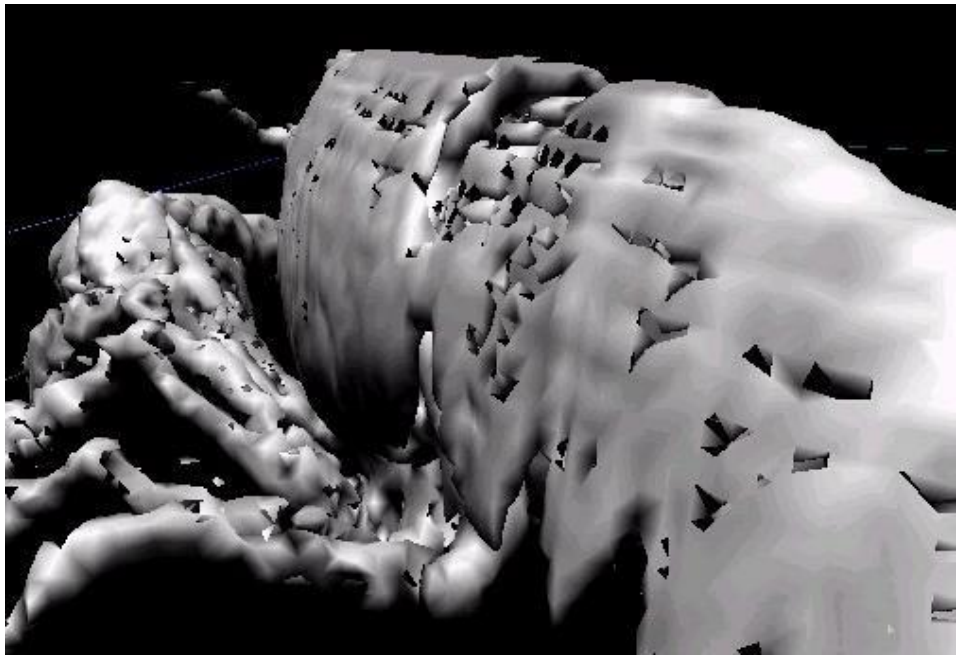
- Have to make consistent choices for neighboring cubes – otherwise get holes



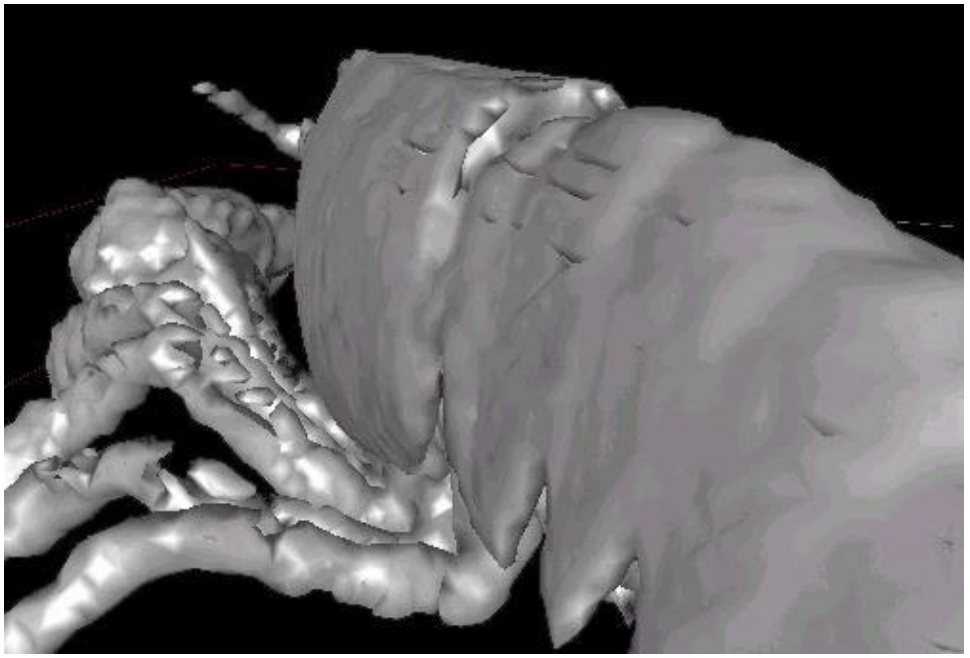


Marching Cubes – Problèmes

- Résolution des ambiguïtés



Ambiguity



No Ambiguity

Alternating Direction Method of Multipliers (ADMM)

Problème:

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

Lagrangien augmenté :

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2$$

Étapes:

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x L_\rho(x, z^k, y^k) \\ z^{k+1} &:= \operatorname*{argmin}_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

Requiert une leçon, mais vous pouvez me croire...

ADMM formulation de Sparse ICP

$$(R,t)=argmin \sum_i \|R.p_i+t-q_i\|^p$$ est reformulé :

$$(R,t)=argmin \sum_i \|z_i\|^p \quad s.t. \quad z_i=R.p_i+t-q_i$$

Lagrangien augmenté : $$L=\sum_i \|z_i\|^p+y_i^T\cdot(R.p_i+t-q_i-z_i)+\rho/2\|R.p_i+t-q_i-z_i\|^2$$

ADMM formulation de Sparse ICP

$$(R,t)=argmin \sum_i \|R.p_i+t-q_i\|^p$$ est reformulé :

$$(R,t)=argmin \sum_i \|z_i\|^p \quad s.t. \quad z_i=R.p_i+t-q_i$$

Lagrangien augmenté : $$L=\sum_i \|z_i\|^p+y_i^T\cdot\underbrace{(R.p_i+t-q_i-z_i)}_{:=b}+\rho/2\|R.p_i+t-q_i-z_i\|^2$$

Itérer:

a) z = argmin L. Vous pouvez vérifier que : $$L=\sum_i \|z_i\|^p+\rho/2\,\|z_i-(b+y_i/\rho)\|^2+const$$
→ Solution analytique indépendante (« shrink operateur », voir papier)

ADMM formulation de Sparse ICP

$$(R,t)=argmin \sum_i \|R.p_i+t-q_i\|^p$$ est reformulé :

$$(R,t)=argmin \sum_i \|z_i\|^p \quad s.t. \quad z_i=R.p_i+t-q_i$$

Lagrangien augmenté : $$L=\sum_i \|z_i\|^p+y_i^T\cdot\underbrace{(R.p_i+t-q_i-z_i)}_{:=b}+\rho/2\|R.p_i+t-q_i-z_i\|^2$$

Itérer:

a) z = argmin L. Vous pouvez vérifier que : $L=\sum_i \|z_i\|^p+\rho/2\,\|z_i-(b+y_i/\rho)\|^2+const$
→ Solution analytique indépendante (« shrink operateur », voir papier)

b) (R,t)= argmin L. N'implique que des termes quadratiques → Procustes classique

ADMM formulation de Sparse ICP

$$(R,t)=argmin \sum_i \|R.p_i+t-q_i\|^p$$ est reformulé :

$$(R,t)=argmin \sum_i \|z_i\|^p \quad s.t. \quad z_i=R.p_i+t-q_i$$

Lagrangien augmenté : $$L=\sum_i \|z_i\|^p+y_i^T\cdot\underbrace{(R.p_i+t-q_i-z_i)}_{:=b}+\rho/2\|R.p_i+t-q_i-z_i\|^2$$

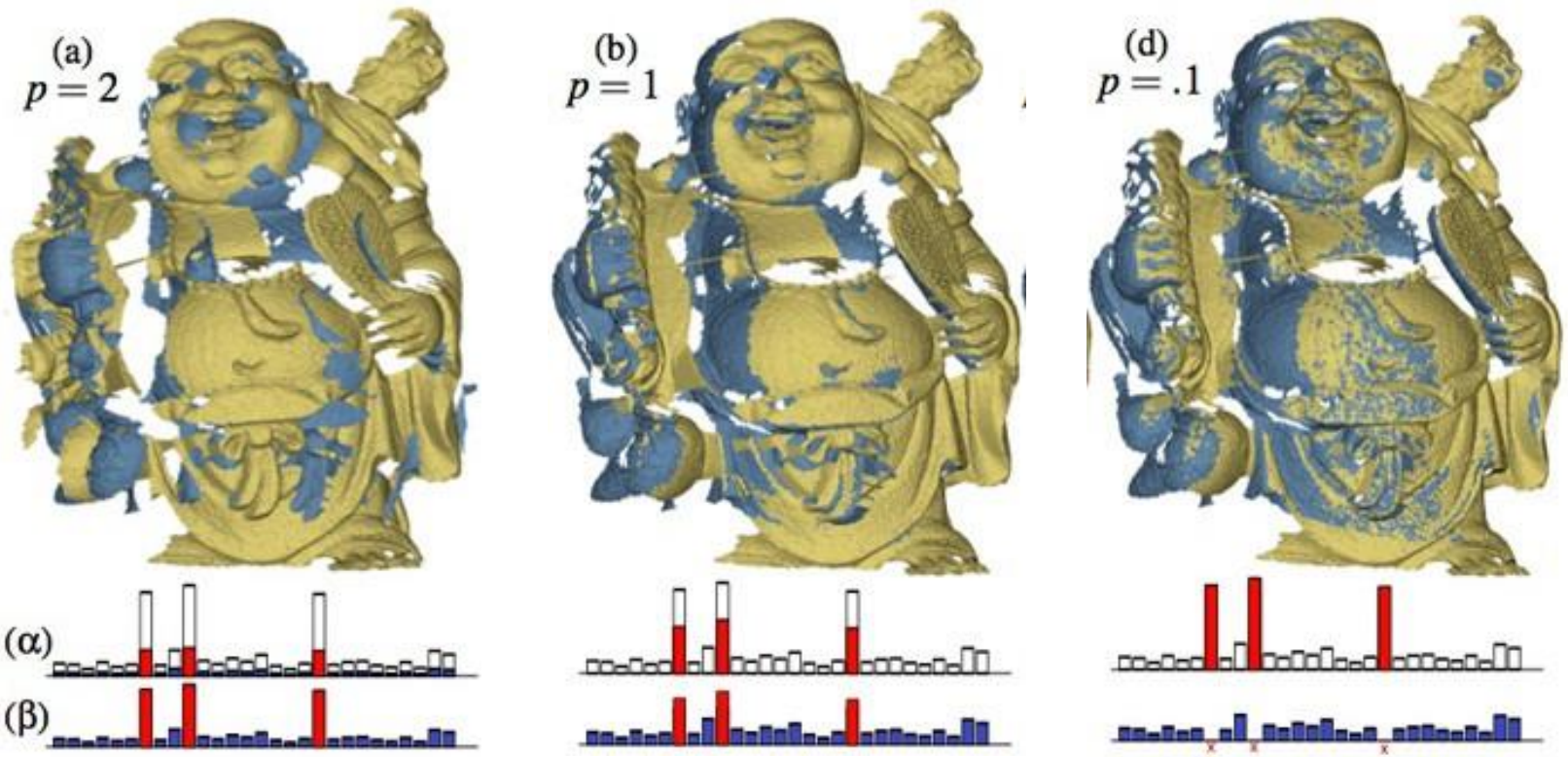
Itérer:

a) z = argmin L. Vous pouvez vérifier que : $L=\sum_i \|z_i\|^p+\rho/2\,\|z_i-(b+y_i/\rho)\|^2+const$
→ Solution analytique indépendante (« shrink operateur », voir papier)

b) (R,t)= argmin L. N'implique que des termes quadratiques → Procustes classique

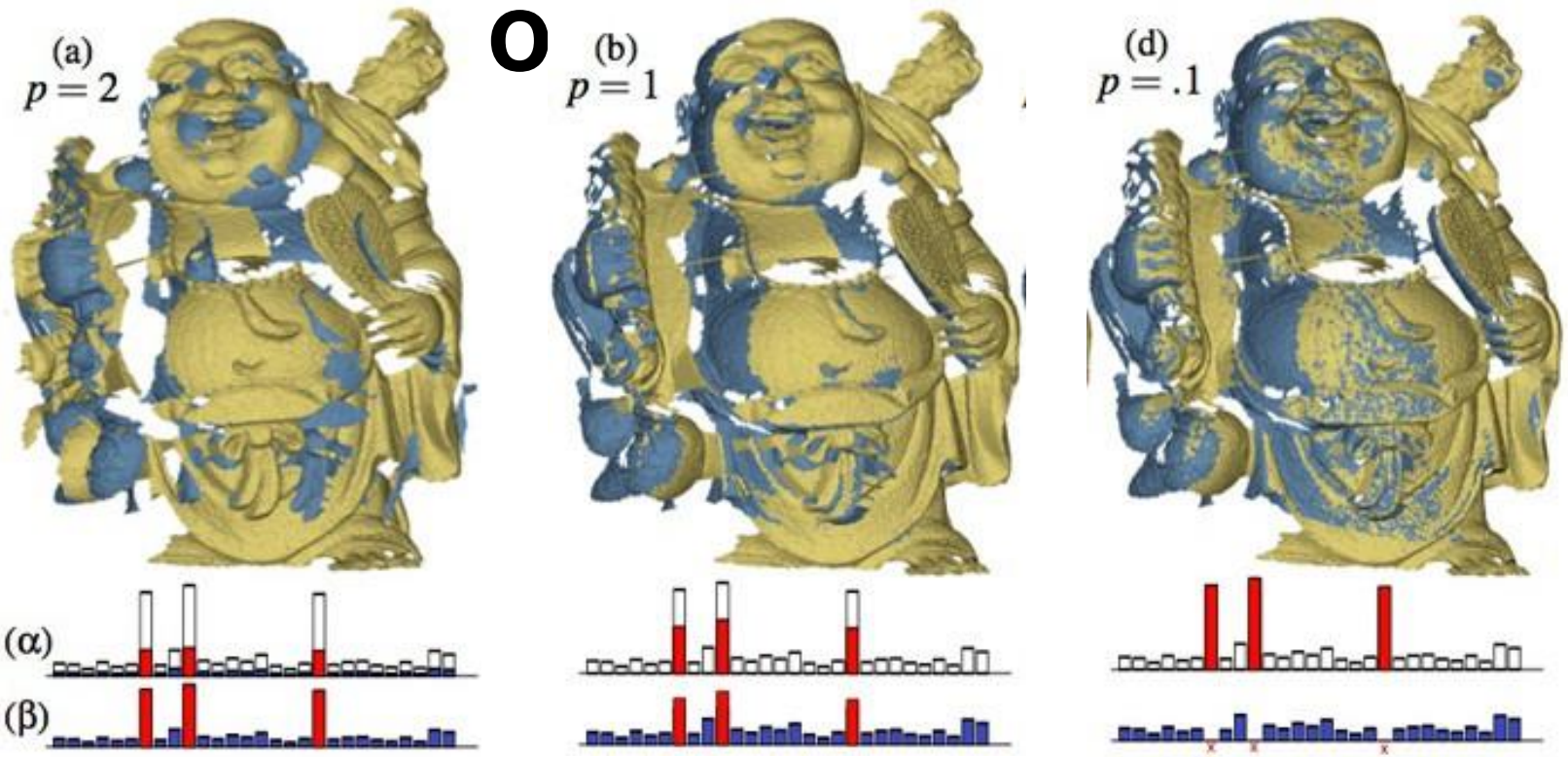
c) $y_i:=y_i+\rho(R.p_i+t-q_i-z_i)$

Application : « Sparse » ICP



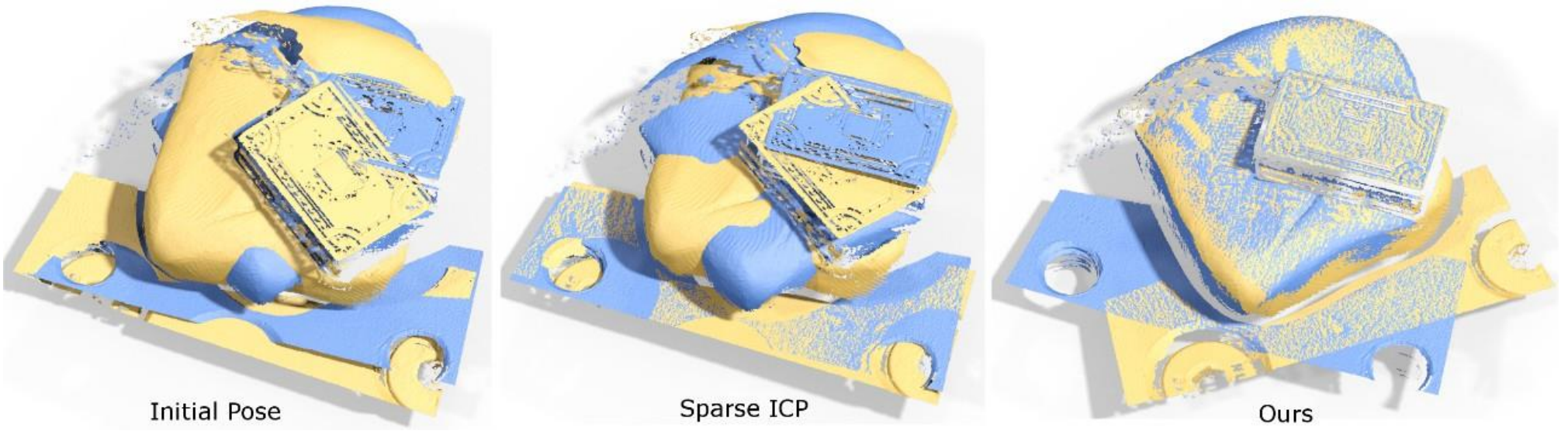
[Bouaziz et al. 2013] : Sparse Iterative Closest Point

Vidéo démo



https://www.youtube.com/watch?v=ii2vHBwlmo8

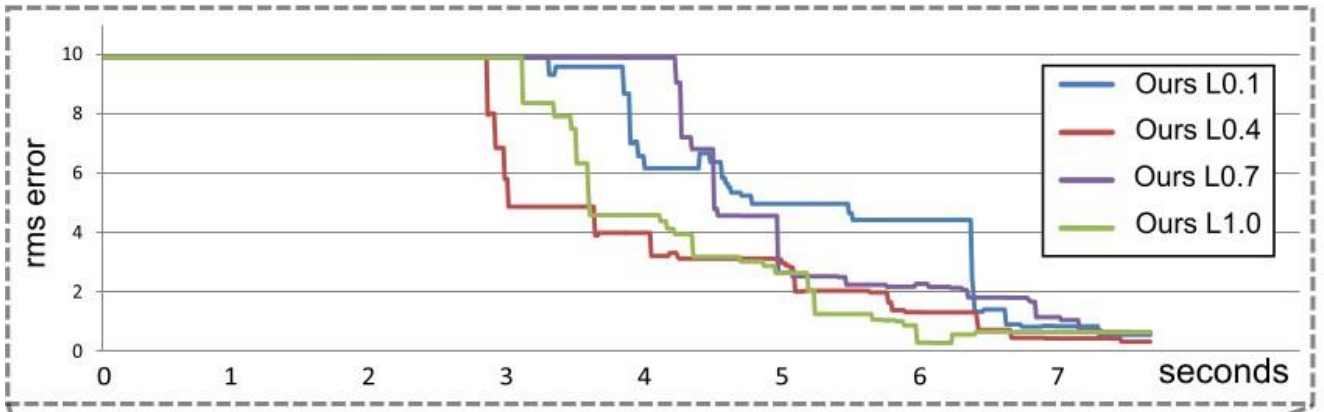
Sparse ICP : optimisation



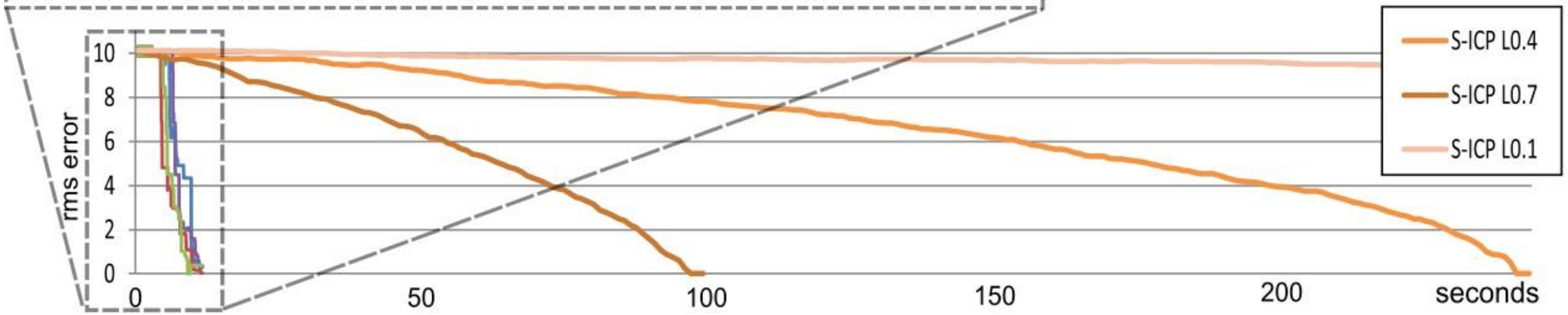
- ADMM est un couteau suisse, MAIS :
- Minima locaux
- Il y a souvent mieux !
- Comme la: Combiner ADMM avec un simple recuit simulé

[Mavridis et al. 2015] : Efficient Sparse ICP

Efficient sparse ICP



Norm	S-ICP	Ours	Speedup
$\ell_{1.0}$	37.6	5.2	7.2×
$\ell_{0.7}$	99.9	5.6	17.8×
$\ell_{0.4}$	246.8	6.6	37.4×
$\ell_{0.1}$	2646.3	7.5	352.8×



[Mavridis et al. 2015] : Efficient Sparse ICP