Reconnaissance et suivi
de modèles déformables (structurés)

Master IVI,
module VisA

généralités

- Comment suivre un objet déformable dans une séquence vidéo?
- On dispose de points caractéristiques
- le plus souvent: extraire un squelette pour structurer
(meme si pas toujours: ex, contours actifs)

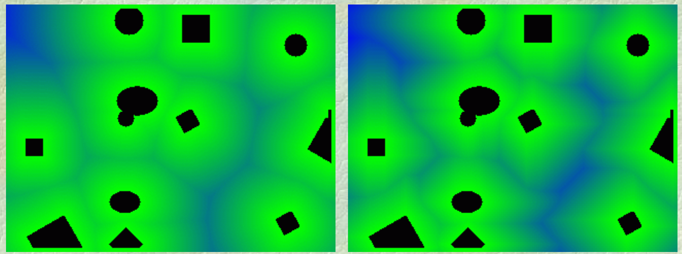
-> extraction et suivi de squelette

Cartes de distances

- Cartes de distances continues: intéressantes en théorie (Voronoi/Delaunay), mais...
- Cartes de distances discretes: on perd en précision, mais inévitable en pratique

Cartes de distances

- Cartes de distances discretes: approches «naives»:
D(p) = min { dist(p,q), q O}



Distance euclidienne Distance de Manhattan

Tres rapide à coder, mais tres lent

Cartes de distances

• Cartes de distances discretes: distance de Chanfrein
On associe une valeur entière à chaque déplacement elémentaire dans un voisinage -> masque

	1	
1	0	1
	1	

d_4

1	1	1
1	0	1
1	1	1

d_8

4	3	4
3	0	3
4	3	4

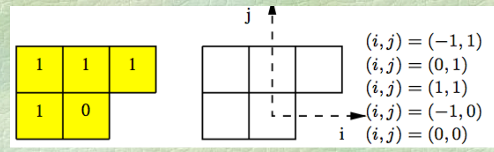
$d_{3,4}$

	11		11	
11	7	5	7	11
	5	0	5	
11	7	5	7	11
	11		11	

$d_{5,7,11}$

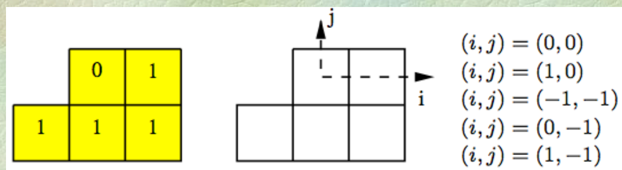
Cartes de distances

• Cartes de distances discretes: demi-masques
propagation de l'information
- initialisation 0 pour les points de référence, +infini pour les autres
- premier balayage de l'image de gauche à droite, et de haut en bas, avec le demi-masque de chanfrein avant (partie du masque décrivant la distance des points voisins se trouvant avant le centre dans le sens du balayage)
calcul: $DM(x,y)=Min\ (I(x+i,\ y+j)+M_{i,j})$ pour tous les (i,j) du masque



Cartes de distances

• Cartes de distances discretes: demi-masques
Second balayage: de droite à gauche et de bas en haut, calcul avec le demi-masque arrière



Cartes de distances

• Cartes de distances discretes: Rosenfeld-Pfaltz
Exemple

Balayage avant

1	1	1
1	0	

∞	∞	∞	∞	∞		1	2
∞	∞	∞	∞	1	1	1	2
∞	∞	∞	2	2	2	2	2
∞		1	2	3	3	3	3
1	1	1	2	3	4	4	4
2	2	2	2	3	4	5	5

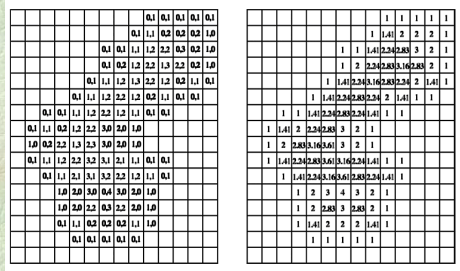
Balayage arrière

	0	1
1	1	1

3	3	3	2	1		1	2
2	2	2	2	1	1	1	2
1	1	1	2	2	2	2	2
1		1	2	3	3	3	3
1	1	1	2	3	4	4	4
2	2	2	2	3	4	5	5

Cartes de distances

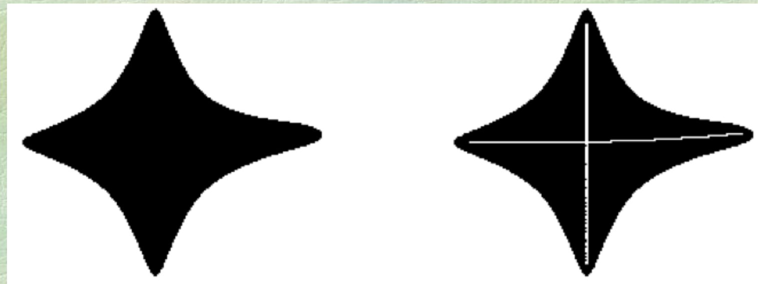
- Cartes de distances discretes: distance de danielsson



$$Distance=\sqrt{((x1-x2)*sp[0])^2+((y1-y2)*sp[1])^2}$$

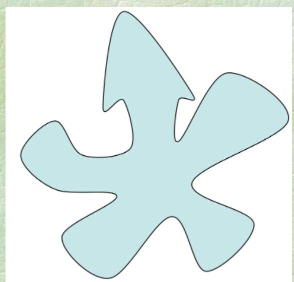
Extraction de squelette

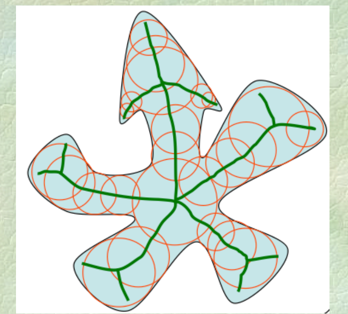
- Principe: codage de forme, représentation simplifiée pour la reconnaissance de forme (préservation topologique, taille)



Extraction de squelette

- Principe: codage de forme, représentation simplifiée pour la reconnaissance de forme (préservation topologique, taille)



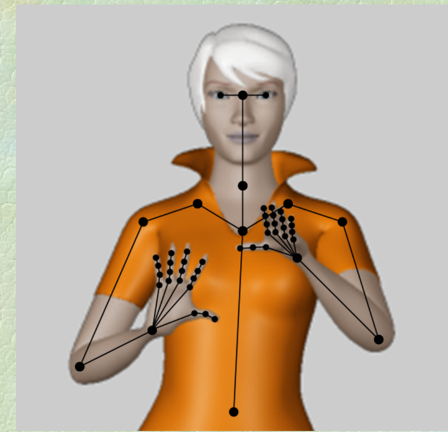


Simplification de squelettes

- Classiquement nécessaire
- Passage discret ->version vectorisée

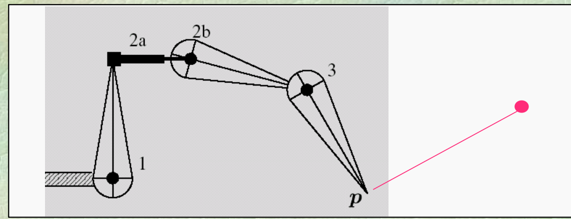
Tracking de squelettes

- Suivi de squelette: problème dual de celui de l'animation de squelettes
  - -> outils de robotique, humain virtuel



Tracking de squelettes: cinématique inverse

- Objet articulé
  - Liens, articulations
  - Objectif à atteindre



Tracking de squelettes: cinématique inverse

- Donnée : position à atteindre (M)
- Sortie : valeurs des paramètres des articulations
- Θ=vecteur des paramètres du modèle
  - $\Theta = (\theta_1, \theta_2, \theta_3, \ldots, t_1, t_2, \ldots)$
- Trouver $\Theta = g(\mathrm{M})$
- Deux rotations: calcul direct
- Trois rotations : calcul direct
- N articulations : ?

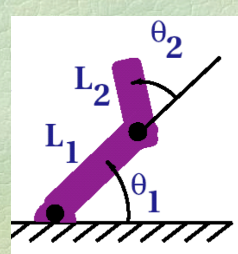
Tracking de squelettes: cinématique inverse

Deux rotations

- Solution directe :

$$d = \sqrt{x^2 + y^2}$$
$$d^2 = (L_1 + L_2 \cos\theta_2)^2 + (L_2 \sin\theta_2)^2$$
$$d^2 = L_1^2 + L_2^2 + 2L_1L_2\cos\theta_2$$
$$\cos\theta_2 = \frac{d^2 - L_1^2 - L_2^2}{2L_1L_2}$$

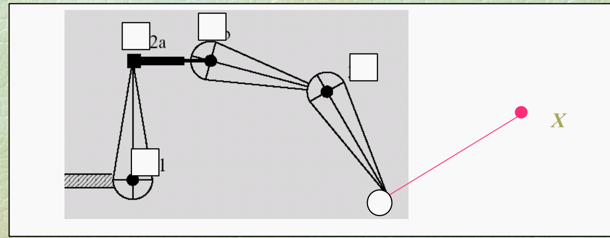
$$\tan(\alpha) = \frac{L_2 \sin\theta_2}{L_1 + L_2\cos\theta_2}$$
$$\tan(\theta_1 + \alpha) = \frac{y}{x}$$
$$\theta_1 = \arctan\left(\frac{y}{x}\right) - \arctan\left(\frac{L_2\sin\theta_2}{L_1 + L_2\cos\theta_2}\right)$$



Trois rotations

- Encore une solution directe
  - trigonométrie
- Paramètre supplémentaire
  - Choix de la solution
- Limites des articulations

Cas général : n articulations



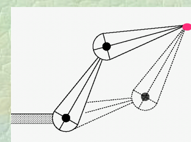
- On veut que $f(\Theta)$=M
- Θ=vecteur des paramètres du modèle
  - $\Theta = (\theta_1, \theta_2, \theta_3, \ldots, t_1, t_2, \ldots)$
- $f(\Theta)$ position de l'extrémité du modèle (coord. 2D)
- M position de la cible (coordonnées 2D)
- Connaissant M, trouver Θ

Pourquoi c'est difficile ?

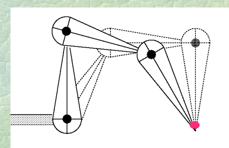
- Problème non-linéaire
- Plusieurs solutions
- Pas toujours bien conditionné
- Limites des articulations

Non-unicité

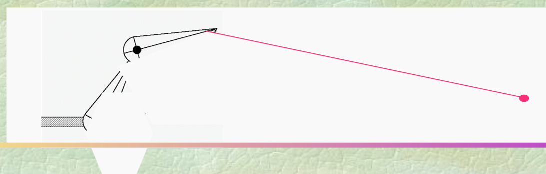
- Deux solutions :



- Intervalle de solutions :

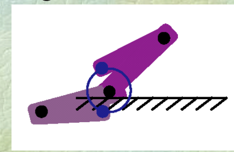


- Pas de solutions :

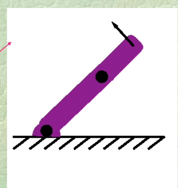


Pas toujours bien conditionné

- Petite différence sur M, grande différence sur Θ :



- Changer Θ ne rapproche pas de M :



Au fait, que vaut f ?

- Matrice de transformation
- Concaténation des matrices
- $f(\Theta) = \mathbf{R}_1(\theta_1)\mathbf{T}_1\mathbf{R}_2(\theta_2)\,\mathbf{T}_2\mathbf{R}_3(\theta_3)\,\mathbf{T}_3 \ldots M_0$
  - M_0 position extrémité du bras avant rotations
- Non-linéaire à cause des rotations
- Calcul de f : cinématique directe

Racines d'une fonction non-linéaire

- On veut trouver Θ tel que : $f(\Theta)$-M = 0
- Linéarisation du problème :

$$f(\Theta + h) = f(\Theta) + f'(\Theta)h + f''(\Theta)h^2 + \ldots$$

- On part d'une valeur de Θ et de $f(\Theta)$
- On ne connaît pas Λ tel que $f(\Theta+\Lambda)=M$
- On peut trouver Λ' qui s'en rapproche
- $\Theta \leftarrow \Theta+\Lambda'$ et on itère

Linéarisation

- Séries de Taylor :

$$f(\Theta + h) = f(\Theta) + f'(\Theta)h + f''(\Theta)h^2 + \ldots$$

- Cas des fonctions à plusieurs variables :

$$f(\Theta + h) = f(\Theta) + \mathbf{J}(\Theta)h + {}^t h\mathbf{H}(\Theta)h + \ldots$$

- $\mathbf{J}$ Jacobien de f, forme linéaire
- $\mathbf{H}$ Hessien de f, forme quadratique

Jacobien

- Matrices des dérivées d'une fonction à plusieurs variables :

$$J_{ij} = \frac{\partial f_i}{\partial x_j}$$

$$\mathbf{J}(\Theta) = \begin{bmatrix} \frac{\partial f_x}{\partial x_0}(\Theta) & \frac{\partial f_x}{\partial x_1}(\Theta) & \dots \\ \frac{\partial f_y}{\partial x_0}(\Theta) & \frac{\partial f_y}{\partial x_1}(\Theta) & \dots \\ \frac{\partial f_z}{\partial x_0}(\Theta) & \frac{\partial f_z}{\partial x_1}(\Theta) & \dots \end{bmatrix}$$

Jacobien

- Variation de $f(\Theta)$ au premier ordre
- Approximation linéaire de f
- Matrice 3*n (ou 2*n en 2D)
- Calcul de **J**:
  - $f(\Theta) = \mathbf{R}_1(\theta_1)\mathbf{T}_1\mathbf{R}_2(\theta_2)\mathbf{T}_2\mathbf{R}_3(\theta_3)\mathbf{T}_3 \dots M_0$

$$\frac{\partial f}{\partial \theta_1} = \frac{\partial R_1}{\partial \theta_1} T_1 R_2 T_2 R_3 T_3 \dots M_0$$

Linéarisation du problème

- $f(\Theta)$-M = E, erreur actuelle
- Trouver Λ, tel que $\mathbf{J}(\Theta)\Lambda$ = E
- Λ : résolution système linéaire
- Approximation linéaire : petits déplacements
  - Petits déplacements dans la direction de Λ
- Série de petits déplacements
- Recalculer **J** et E à chaque étape

Résolution itérative

- Petits pas par petits pas
- À chaque étape :
  - Choix entre plusieurs solutions
  - Prendre solution proche position actuelle
- Taille des pas :
  - Chercher taille optimale ?
  - Rotations : pour x < 2 degrés:
    - sin(x) ≈ x
    - cos(x) ≈ 1
  - Garder pas < 2 degrés

Algorithme

```
inverseKinematics()
 {
   start with previous Θ;
        E = target - computeEndPoint();
   for(k=0; k<k_max && |E| > eps; k++){
     J = computeJacobian();
     solve J Λ= E;
     if (max(Λ)>2) Λ= 2Λ/max(Λ);
     Θ = Θ+ Λ;
          E = target - computeEndPoint();
   }
 }
```

Tracking de squelettes: cinématique inverse

La bonne question

- **solve J Λ = E;**
- **J** n'est pas inversible
  - En fait, **J** n'est même pas carrée
  - **J** matrice 2*n :

$$\begin{bmatrix} \frac{\partial f_x}{\partial x_0}(\Theta) & \frac{\partial f_x}{\partial x_1}(\Theta)\dots \\ \frac{\partial f_y}{\partial x_1}(\Theta) & \frac{\partial f_y}{\partial x_1}(\Theta)\dots \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} E_x \\ E_y \end{bmatrix}$$

Tracking de squelettes: cinématique inverse

Pseudo-Inverse

- $\mathbf{J}^T\mathbf{J}$ est carrée (n*n). Donc :
  - $\mathbf{J}\,\Lambda = \mathrm{E}$
  - $\mathbf{J}^T\mathbf{J}\,\Lambda = \mathbf{J}^T\mathrm{E}$
  - $\Lambda = (\mathbf{J}^T\mathbf{J})^{-1}\,\mathbf{J}^T\mathrm{E}$
  - $\Lambda = \mathbf{J}^+\mathrm{E}$
- $\mathbf{J}^+ = (\mathbf{J}^T\mathbf{J})^{-1}\,\mathbf{J}^T$ *pseudo-inverse* de **J**
  - Pareil que l'inverse si **J** est carrée et inversible
  - Propriétés : $\mathbf{JJ}^+\mathbf{J} = \mathbf{J}$, $\mathbf{J}^+\mathbf{JJ}^+ = \mathbf{J}^+$
  - **J** est $m*n \Rightarrow \mathbf{J}^+$ est $n*m$
- Comment calculer $\mathbf{J}^+$?
  - Surtout si $\mathbf{J}^T\mathbf{J}$ n'est pas inversible

Tracking de squelettes: cinématique inverse

Singular Values Decomposition

- Toute matrice m*n peut s'exprimer par SVD:
  - $\mathbf{A}=\mathbf{USV}^T$
    - **U, V** : matrices rectangulaires, colonnes orthogonales
    - S matrice diagonale, *singular values*

$$\mathbf{A} = \mathbf{U}\begin{pmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_n \end{pmatrix}\mathbf{V}^T$$

Singular Values Decomposition

- **S** unique à l'ordre et au signe des valeurs près
  - Ordre canonique : s_i positifs, ordre croissant
  - Rang de **A** : nombre de valeurs non-nulles
  - Déterminant : produit des valeurs
- **U, V** : colonnes orthogonales

$$\mathbf{U} = \begin{pmatrix} \vec{h}_1 & \vec{h}_2 & \cdots & \vec{h}_n \end{pmatrix}$$

Pseudo-Inverse avec SVD

- Calculer SVD: $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$
- Pseudo-inverse : $\mathbf{A}^+ = \mathbf{V}\mathbf{S}^{-1}\mathbf{U}^T$

$$\begin{pmatrix} s_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & s_n \end{pmatrix}^{-1} = \begin{pmatrix} \frac{1}{s_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{1}{s_n} \end{pmatrix}$$

- Singulière : $s_i = 0$
- Mal conditionnée $s_i \ll s_0$
  - Prendre 0 au lieu de $1/s_i$ pour ces valeurs
  - Test : $s_i < \varepsilon\, s_0$

Résoudre **A***X*=**B** avec SVD

- $\mathbf{A}^+$ tronquée
- $\mathbf{A}^+\mathbf{B}$ donne la solution des *moindres carrés* :
  - Pas de solutions : X qui minimise $\|\mathbf{A}X-\mathbf{B}\|^2$
  - Plusieurs solutions : $\|X\|^2$ minimal tel que $\mathbf{A}X=\mathbf{B}$
  - Stable numériquement, même pour matrices mal-conditionnées
- SVD : marteau-pilon
  - $O(n^3)$ (lent)
  - Marche toujours, et assez rapide pour nous.
  - Difficile à implémenter, cf. *Numerical Recipes in C*
- Autres méthodes pour IK: CCD, $\mathbf{J}^T$,…

Et la cinématique inverse ?

- On veut résoudre $X=f(\Theta)+\mathbf{J}(\Theta)\Lambda$
  - $f(\Theta)$ position de l'extrémité du bras
  - i^e colonne de **J** vient de l'articulation i
  - $f(\Theta)= \mathbf{R}_1(\theta_1)\mathbf{T}_1\mathbf{R}_2(\theta_2)\,\mathbf{T}_2\mathbf{R}_3(\theta_3)\,\mathbf{T}_3\ldots M_0$

$$\frac{\partial f}{\partial \theta_1} = \frac{\partial R_1}{\partial \theta_1} T_1 R_2 T_2 R_3 T_3 \ldots M_0$$

$$\frac{\partial f}{\partial \theta_1} = R_1(\theta_1 + \tfrac{\pi}{2}) T_1 R_2 T_2 R_3 T_3 \ldots M_0$$

Calcul du Jacobien

- Jacobien d'une rotation :

$$\vec{r} = f(\Theta) - \text{pivot}_i = \begin{pmatrix} r_x \\ r_y \end{pmatrix}$$

$$\frac{\partial f}{\partial \theta_i}(\Theta) = \begin{pmatrix} -r_y \\ r_x \end{pmatrix}$$

- Jacobien d'une translation
  - Vecteur dans la direction de translation
- Remarques :
  - Calcul en coordonnées du monde, pas du modèle
  - Degrés/radians !!! (dérivée *=π/180 ?)
  - Un degré de liberté par articulation

Algorithme

```
inverseKinematics()
  {
    Vector Θ = getLinkParameters();
    Vector E = target - computeEndPoint();
          for(k=0; k<k_max && E.norm() > eps; k++){
      Matrix J = computeJacobian();
      Matrix J+ = pseudoInverse(J);
      Vector Λ = J+E ;
      if (max(Λ)>2) Λ *= 2/max(Λ);
      Θ = Θ+ Λ;
            putLinkParameters();
      E = target - computeEndPoint();
    }
  }
```


Inverse Kinematics, niveau II

- Limites aux articulations

Limites aux articulations

- Chaque articulation a un intervalle limité
  - Par ex. le coude : varie sur [0,π]
  - Élément important du réalisme
- Pour forcer les limites :
  - Tester si dépassement
  - Annuler paramètre *i*
  - Recalculer sans *i*
  - Vérifier les autres paramètres

Tracking de squelettes: cinématique inverse

Limites aux articulations

- Algorithme modifié :
  - Après avoir calculé Λ, test pour ch. articulation:

$$\theta_i^{\min} < \theta_i + \lambda_i < \theta_i^{\max}$$

  - Si ça sort de l'intervalle :
    - Annuler colonne i de $\mathbf{J}$
    - Revient à annuler paramètre i
  - Recalculer $\mathbf{J}^+$
    - Moindres carrés : $\lambda_i \approx 0$
    - Pour plus de robustesse, forcer $\lambda_i = 0$
  - Trouver Λ, itérer

Reconnaissance d'objets déformables: graphes de Reeb

Et pour finir...

- Un outil intéressant : graphes de Reeb
  - on envisage un objet sous la forme de coupes horizontales
  - les sommets du graphe sont les points de changements topologiques
  - les arcs du graphe sont ce qu'il reste de l'objet, en contractant la surface jusqu'aux points sommets
  - utilisable pour la catégorisation d'objets