

Projet EDTS

Suivi de visage/objet automatiquement dans une vidéo RGB

Junzhe Hao, Yi YU, ASI 5
15/12/2015

Plan

- ❑ **Introduction**
- ❑ **Bibliographie**
- ❑ **Méthode**
- ❑ **Réalisation**
- ❑ **Démonstration**
- ❑ **Amélioration**
- ❑ **Conclusion**



Introduction

L'objectif : suivi d'objet/visage dans une vidéo avec un filtre particulière.

Language utilisé : python (cv, cv2, numpy, scipy)



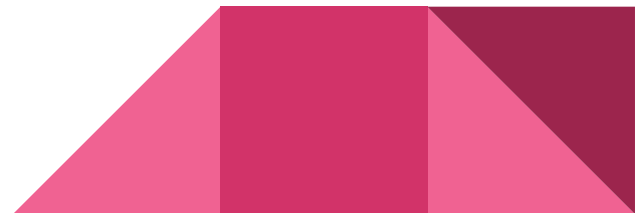
Bibliographie

- ❏ An Adaptive Color-Based Particle Filter

Fabian Kaelin, McGill University

- ❏ Object Tracking with an Adaptive Color-Based Particle Filter

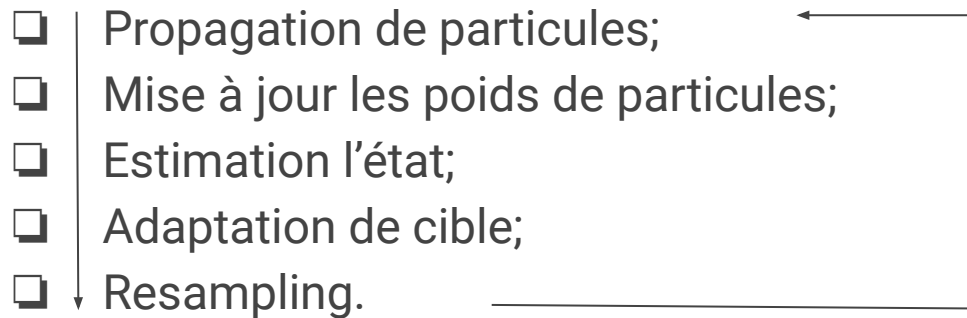
Katja Nummiaro, Esther Koller-Meier, Luc Van Gool



Méthode choisie

Filtre particulaire basé sur couleur :

L'initialisation : choix de cible, génération des particules (nb de particules : 100)

- 
- ```
graph TD; A[L'initialisation : choix de cible, génération des particules (nb de particules : 100)] --> B[Propagation de particules;]; B --> C[Mise à jour les poids de particules;]; C --> D[Estimation l'état;]; D --> E[Adaptation de cible;]; E --> F[Resampling.]; F --> B;
```
- ❑ Propagation de particules;
  - ❑ Mise à jour les poids de particules;
  - ❑ Estimation l'état;
  - ❑ Adaptation de cible;
  - ❑ Resampling.

# Propagation des particules

❑ Modèle dynamique :

$$s = \{\underbrace{x, y}_{\text{position}}, \underbrace{\hat{x}, \hat{y}}_{\text{vitesse}}, \underbrace{H_x, H_y}_{\text{taille}}, \underbrace{\hat{a}}_{\text{scaling change}}\}$$

L'objectif : prédire l'état d'une cible potentielle.

$$s_{t+1} = A * s_t + w_t$$

Les particules sont propagées en utilisant ce modèle.



# Mise à jour les poids de particules

## ❏ Modèle likelihood

L'objectif : calculer le poids de chaque particule.

La méthode utilisée : comparaison de la distribution de couleur

8 bins :

|   | 0-31 | 32-63 | 64-95 | 96-127 | 128-159 | 160-191 | 192-223 | 224-255 |
|---|------|-------|-------|--------|---------|---------|---------|---------|
| u | 1    | 2     | 3     | 4      | 5       | 6       | 7       | 8       |

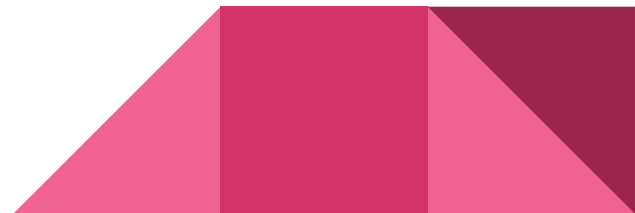
# Calculution de poids d'une particule

1. Weighting function :  $k(r_i) = \begin{cases} 1 - r_i^2 & r_i \leq 1 \\ 0 & otherwise \end{cases}$
2. Distribution de couleur :  $p_s^{(u)} = f \sum_{i=1}^I k(r_i) \delta[h(v_i) - u]$

f : facteur de normalisation :  $f = \sum_{i=1}^I \frac{1}{k(r_i)}$

3. Distance de Bhattacharyya :  $d = \sqrt{1 - \rho[p, q]}$

avec :  $\rho[p, q] = \sum_{u=1}^8 \sqrt{p(u)q(u)}$





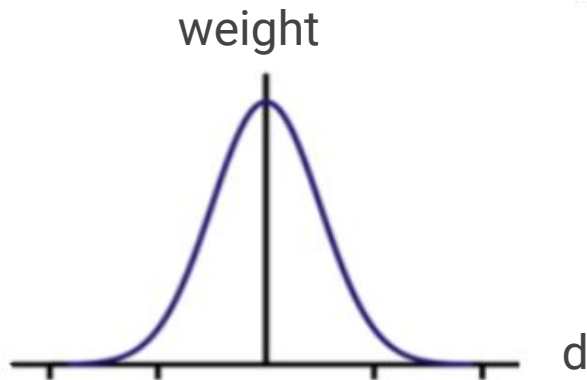
# Calculution de poids d'une particule

4. Le poids :

$$\pi^{(n)} = \frac{1}{(\sqrt{2\pi}\sigma)} e^{-\frac{d^2}{2\sigma^2}} = \frac{1}{(\sqrt{2\pi}\sigma)} e^{-\frac{(1-\rho[p_s^{(n)}, q])}{2\sigma^2}}$$

5. Normalisation de poids :

$$\pi^{(n)} = \frac{\pi^{(n)}}{\sum_{i=1}^N \pi^{(i)}}$$



# Estimation d'état

La moyenne d'état de toutes les particules :

$$E(S) = \sum_{n=1}^N \pi^{(n)} s^{(n)}$$

La meilleur approximation de l'état de la cible



# Adaptation de la cible (Target update)


L'apparence de la cible va changer au cours du temps (Les conditions de l'illumination, l'angle, la rotation, la taille ... )

L'objectif de l'adaptation : surmonter le changement de l'apparence de la cible.

Pour chaque bin  $u$  :

$$q_{t+1}^{(u)} = (1 - \alpha)q_t^{(u)} + \alpha p_{E(S_{t+1})}^{(n)}$$

$\alpha$  mesure le poids de contribution de mean state histogramme au modèle de la cible.



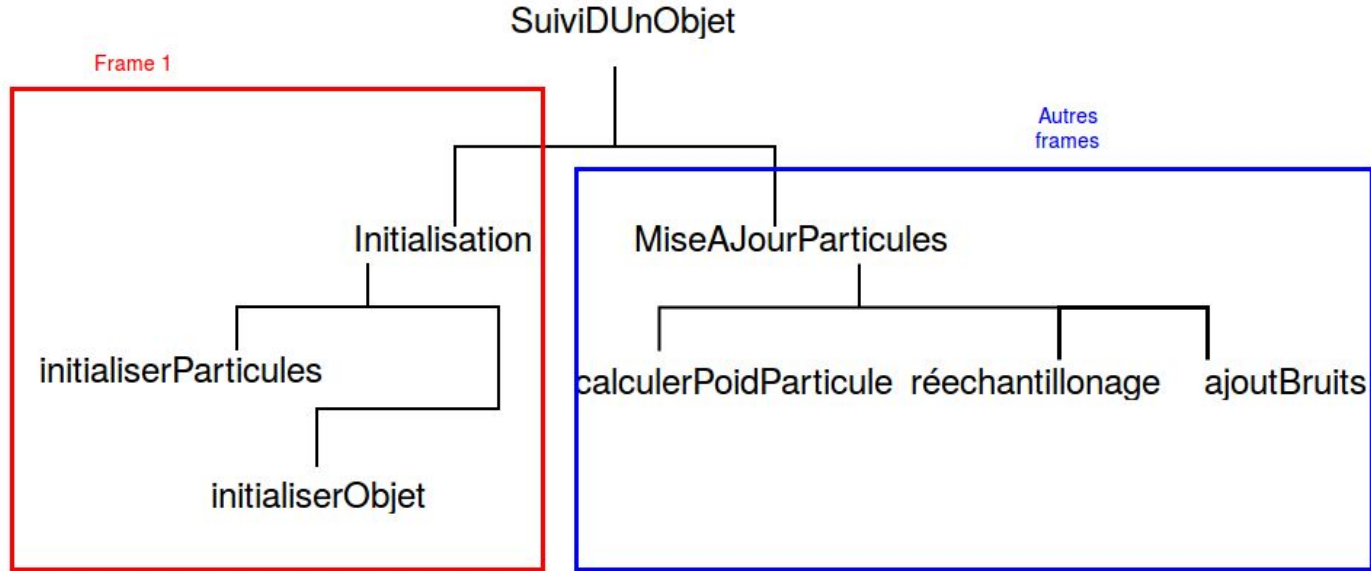
# Resampling

Particle degeneracy : Il existe des particules bougent dans les directions mauvaises alors donc leurs poids deviennent petits après plusieurs itérations. Au fur et à mesure, ces particules ne possèdent plus les informations de la cible.

Resampling : éviter l'effet particle degeneracy. On copie les particules avec les poids élevés, les particules avec poids faibles sont éliminées.

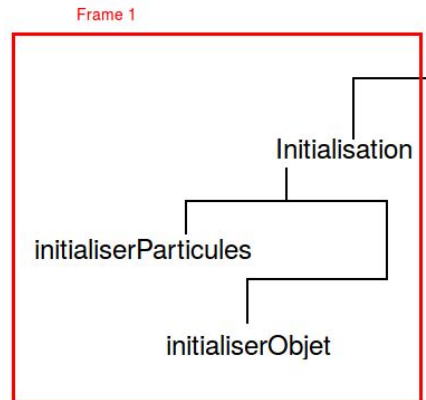


# Réalisation -- Analyse descendante



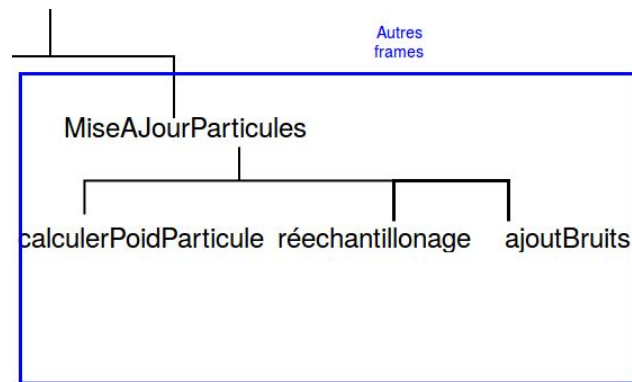
# Réalisation -- Frame 1

- Initialisation du objet cible
  - Segmenter l'objet manuellement utilisant un rectangle
    - Taille du rectangle  $HR*WR$
    - Position initiale de l'objet
  - Calculer la distribution de couleur pour le rectangle
    - Choix de l'espace de couleur : RGB, YUV, etc
    - Pénalisation pour chaque composant de couleur
- Initialisation de particules
  - Définir le modèle de particule
  - Positionner les particules
    - Nb de particules  $N$  à initialiser



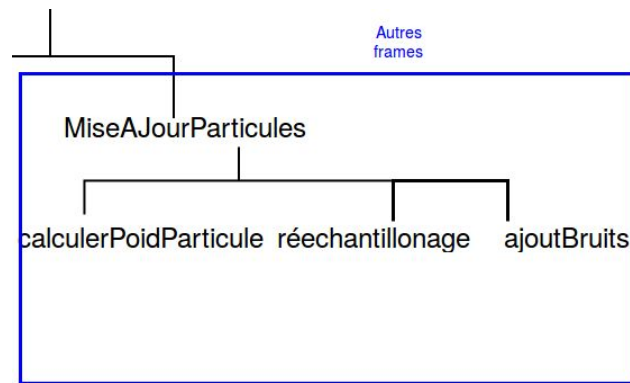
# Réalisation--Autres frames

- Définir le poid de chaque particule  $p_i$ 
  - Définir un rectangle de  $HR*WR$  avec  $p_i$  au centre
    - Éliminer les particules près de la frontière
  - Calculer la distribution de couleur
  - Calculer la distance entre les deux distribution
  - Définir le poids à base de la distance
    - Normaliser les poids



# Réalisation--Autres frames

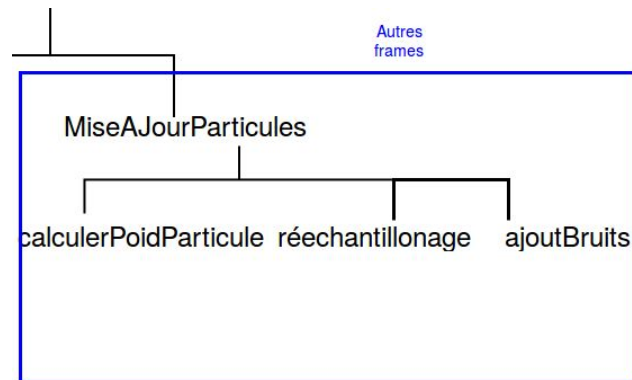
- Ré-échantillonnage des particules
  - Trier les particules selon son poids
  - Répéter X fois pour les particules importants
    - Valeur X dépend du poids du particule
    - Arrête la répétition si nb particules = N
  - Obtenir un ensemble de nouveaux particules





# Réalisation--Autres frames

- Bouger les particules
  - Déplacer les particules utilisant le modèle dynamique
    - Même déplacement pour l'ensemble de particule
  - Ajouter des bruits
    - Permet de déplacer les particules de façon aléatoire



# Réalisation--Structure du code

```
cap = cv2.VideoCapture('vtest.avi')
ret, frame1 = cap.read()
```

Initialisation()

```
while(cap.isOpened()):
```

```
 ret, frame = cap.read()
```

MisaAJour()

```
 cv2.imshow('frame',gray)
```

```
 if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
 break
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```



# Démonstration

- Performance



# Améliorations

- Mise à jour aussi la distribution de couleur de l'objet
  - Grand changement de la distribution de couleur
  - Déformation de l'objet cible
- Définir un seuil pour la distance entre les distributions de couleur
  - Ré-initialiser les particules en cas besoin
- Avantage du python
  - Les types générale : set, diction, etc



# Conclusion

- Réalisation d'une application du filtre particulaire
- Un projet intéressant à améliorer



# Question

