HE Ruoqi
HUNG Chia-man

Détection par "boosting"

Plan

- Introduction
- Implémentation
- Premier résultat
- Amélioration (Non proposée par le sujet)
- Conclusion

Introduction

Objectif : Réalisation d'un détecteur de visage

- Phase 1
  Génération des classifieurs faibles
- Phase 2
  Entraînement des classifieurs faibles – Méthode du perceptron
- Phase 3
  Boosting des classifieurs faibles - Adaboost

Implémentation

- Classe *Pic*

- Premier outil : Intégration d'une image

```
Pic* integral(Pic* pic)
```

-> Relation d'itération :

```
res->setPixel(x,y,res->getPixel(x-1,y)+
                  res->getPixel(x,y-1)-
                  res->getPixel(x-1,y-1)+
                  pic->getPixel(x-1,y-1));
```

Implémentation

- Classe *Classifier*

- Génération des classifieurs faibles

```
vector<Classifier>* generateClassifierTable(int width, int height)
```

-> Au total : 168336, vérifié par calcul

```
double Classifier::evaluate(Pic* pInt)
```

-> Normalisation par surface

Implémentation

- Calcul du tableau de caractéristiques d'une image

- Manière séquentielle :

```
double* calculateFeatures(Pic* pInt, vector<Classifier>* table)
```

- Manière parallèle :

```
double* calculateFeaturesMPI(Pic* pInt, vector<Classifier>* table)
```

- Calcul partiel ():

```
double* calculateFeaturesPartial(Pic* pInt, vector<Classifier>* table)
```

Implémentation

- Entraînement des classifieurs faibles – Méthode du perceptron

- Manière séquentielle :

```
void train (vector<Classifier>* table)
```

- Manière parallèle :

```
void trainMPI (vector<Classifier>* table)
```

```
int Classifier::calculateH(Pic* pInt)
```

-> Détermination de visage par un classifieur faible

Implémentation

- Boosting des classifieurs faibles - Adaboost

- Manière séquentielle :

```
ResultClassifier* boost (vector<Classifier>* table, int round,
string resumeFromFile = "")
```

- Manière parallèle :

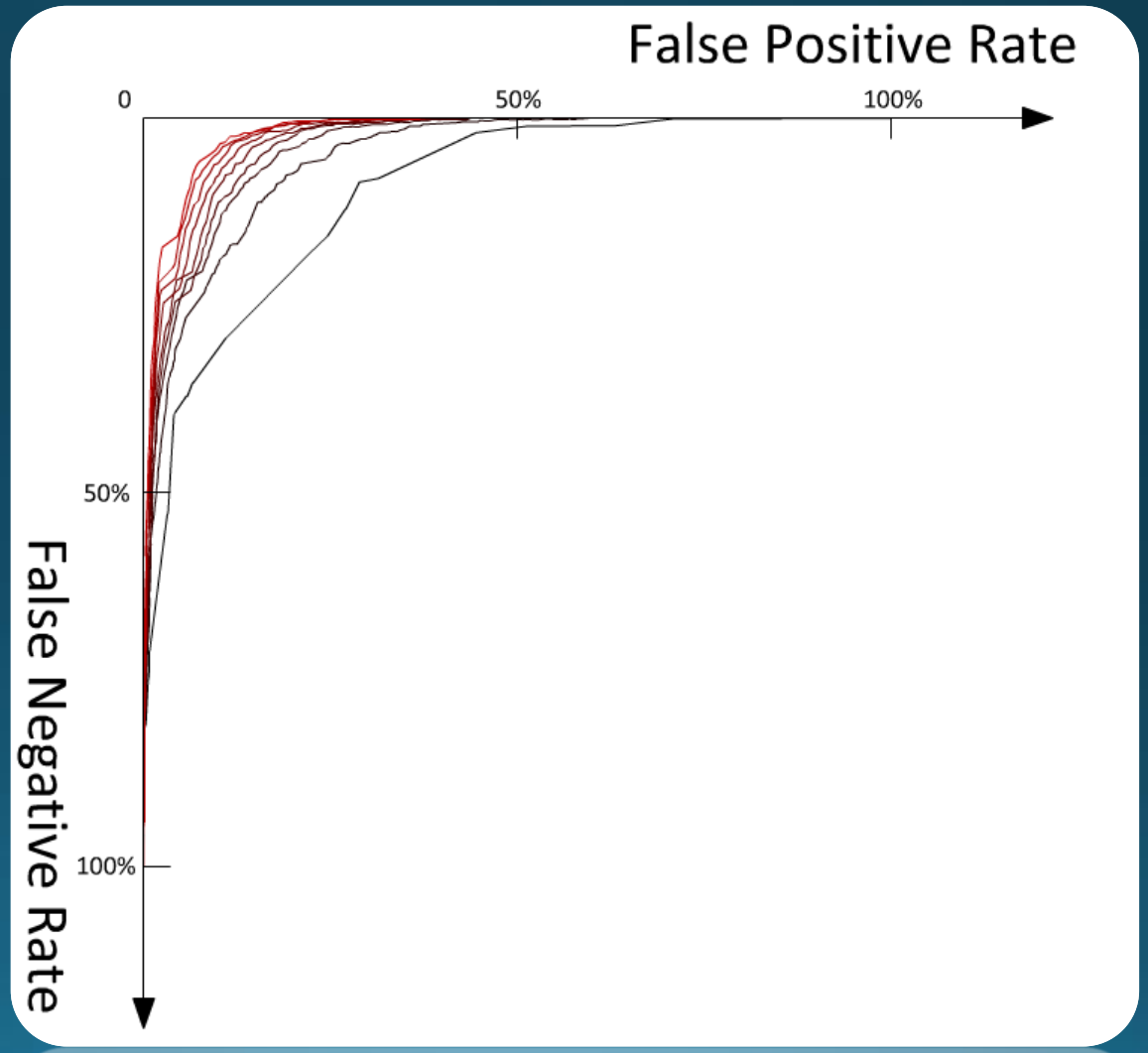
```
ResultClassifier* boostMPI (vector<Classifier>* table, int round,
string resumeFromFile = "")
```

Implémentation

- Classe *ResultClassifier*

```
int ResultClassifier::detect(Pic* pInt, double threshold)
void ResultClassifier::saveToFile(string path)
void ResultClassifier::readFromFile(string path)
```

Premier résultat



Premier résultat

- Classe *BigPic*

```
void BigPic::setWindow(int wx, int wy, int ww, int wh)
```

Amélioration

- Principe :

  Adaboost fort – Entraînement dans chaque tour, on choisit le meilleur classifier faible avec les meilleurs valeurs de w1 et w2.

  Entraînement déterministe – on fixe w1 = ± 1, calcul de w2 qui minimise l'erreur pondérée.

  Même base de données pour le boosting et l'entraînement.

  Classifieur final peut contenir des mêmes classifieurs faibles avec des w1 et w2 différents.

Conclusion

- Encore à développer pour la détection de photo quelconque
- Base de données à améliorer
- Découverte de Machine Learning