DÉTECTION PAR „BOOSTING“
PI N°5 – INF442

Vincent BILLAUT – Marc THIBAULT

Sommaire

I. Principe du détecteur

- Détection de visages dans des images
- Méthode :
  - Apprentissage
  - Validation
  - Test




I. Principe du détecteur

- Détecteur de Viola-Jones
- Retrouver des caractéristiques sur des images

Sommaire

I. Principe du détecteur

II. Premiers essais

III. Implémentation finale

II. Premiers essais

- Importation d'une image en **`Pixel**`**
- Création d'une classe **`Pixel`** : finalement inutile
- Essais de librairies C++

```
(10,9,14)
Printing PixelMap :
  height = 12
  width = 12
(255,255,255)(505,505,505)(760,760,760)(1012,1012,1012)(1266,1266,1266)(15
(315,315,315)(573,573,573)(828,828,828)(1080,1080,1080)(1339,1339,1339)(15
(389,389,389)(647,647,647)(902,902,902)(1156,1156,1156)(1415,1415,1415)(16
(455,455,455)(713,713,713)(1040,1040,1040)(1430,1430,1430)(1829,1829,1829)
(529,529,529)(788,788,788)(1247,1247,1247)(1887,1887,1887)(2541,2541,2541)
(592,592,592)(852,852,852)(1446,1446,1446)(2341,2341,2341)(3248,3248,3248)
(661,661,661)(921,921,921)(1604,1604,1604)(2747,2747,2747)(3909,3909,3909)
(800,800,800)(1060,1060,1060)(1743,1743,1743)(3074,3074,3074)(4486,4486,44
(1019,1019,1019)(1279,1279,1279)(1964,1964,1964)(3296,3296,3296)(4776,4776
(1274,1274,1274)(1682,1682,1682)(2367,2367,2367)(3704,3704,3704)(5184,5184
(1526,1526,1526)(2189,2189,2189)(3095,3095,3095)(4530,4530,4530)(6049,6049
(1780,1780,1780)(2698,2698,2698)(3859,3859,3859)(5549,5549,5549)(7319,7319
```

II. Premiers essais

- Conversion d'une image en fichier texte
- Travail effectué en Python : imgToTxt.py

```
12
12
255 60 74 66 74 63 69 139 219 255 252 252
250 8 0 0 1 1 0 0 0 148 255 255
255 0 0 72 132 135 89 0 2 0 221 221
252 0 2 136 250 255 248 188 1 5 98 98
254 5 0 140 255 253 255 250 68 0 39 39
255 1 0 135 255 253 255 255 136 1 0 0
254 0 1 137 253 252 254 252 137 0 0 0
254 0 3 136 255 255 253 255 102 2 32 32
255 4 0 138 255 253 255 237 16 1 102 102
255 0 2 103 185 189 146 56 0 0 186 186
254 2 0 2 0 1 7 0 0 118 255 255
255 0 0 0 4 0 0 53 170 255 253 253
```

II. Premiers essais

- Calcul parallèle des caractéristiques
- Exportation dans le même .txt par les processi
- Importation des caractéristiques du .txt

```
28112  2 2 7 5 1 : -8
28113  1 2 14 1 7 : -42
28114  2 2 7 8 1 : -49
28115  1 2 14 4 7 : -10
28116  2 3 9 7 9 : -46
28117  2 3 9 10 9 : -168
28118  2 2 14 0 0 : 0
28119  2 3 9 2 10 : 0
28120  2 2 7 11 1 : -161
28121  2 2 7 2 2 : 0
28122  2 2 14 3 0 : 0
28123  2 2 14 6 0 : -25
```

Sommaire

I. Principe du détecteur

II. Premiers essais

III. Implémentation finale

III. Implémentation finale

- Implémentation généraliste des caractéristiques :

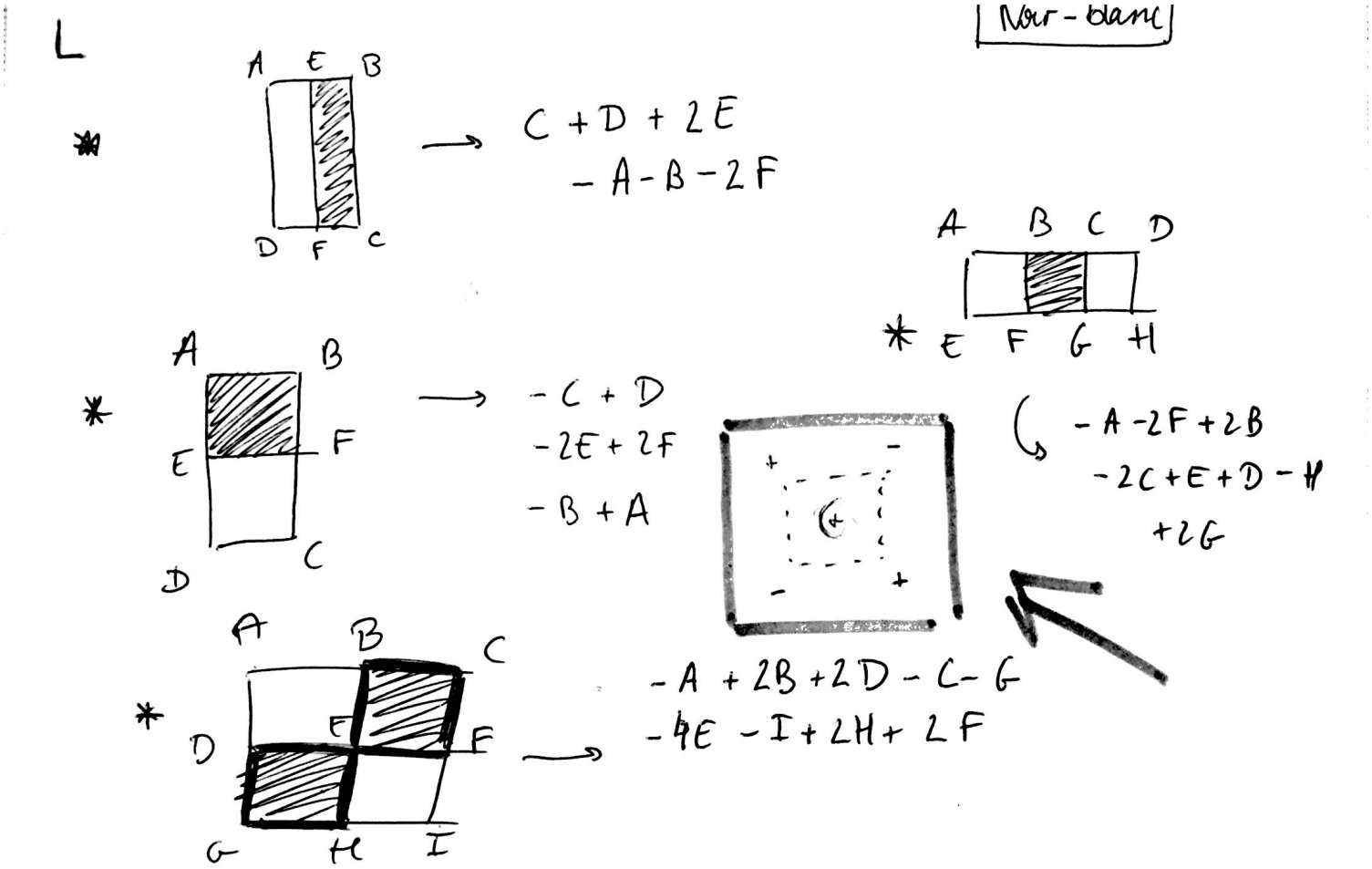
  classe **`Caracteristique`**

- Objet IntegralImage qui contient l'information d'une image :

  `classe`

  `ii`

III. Implémentation finale

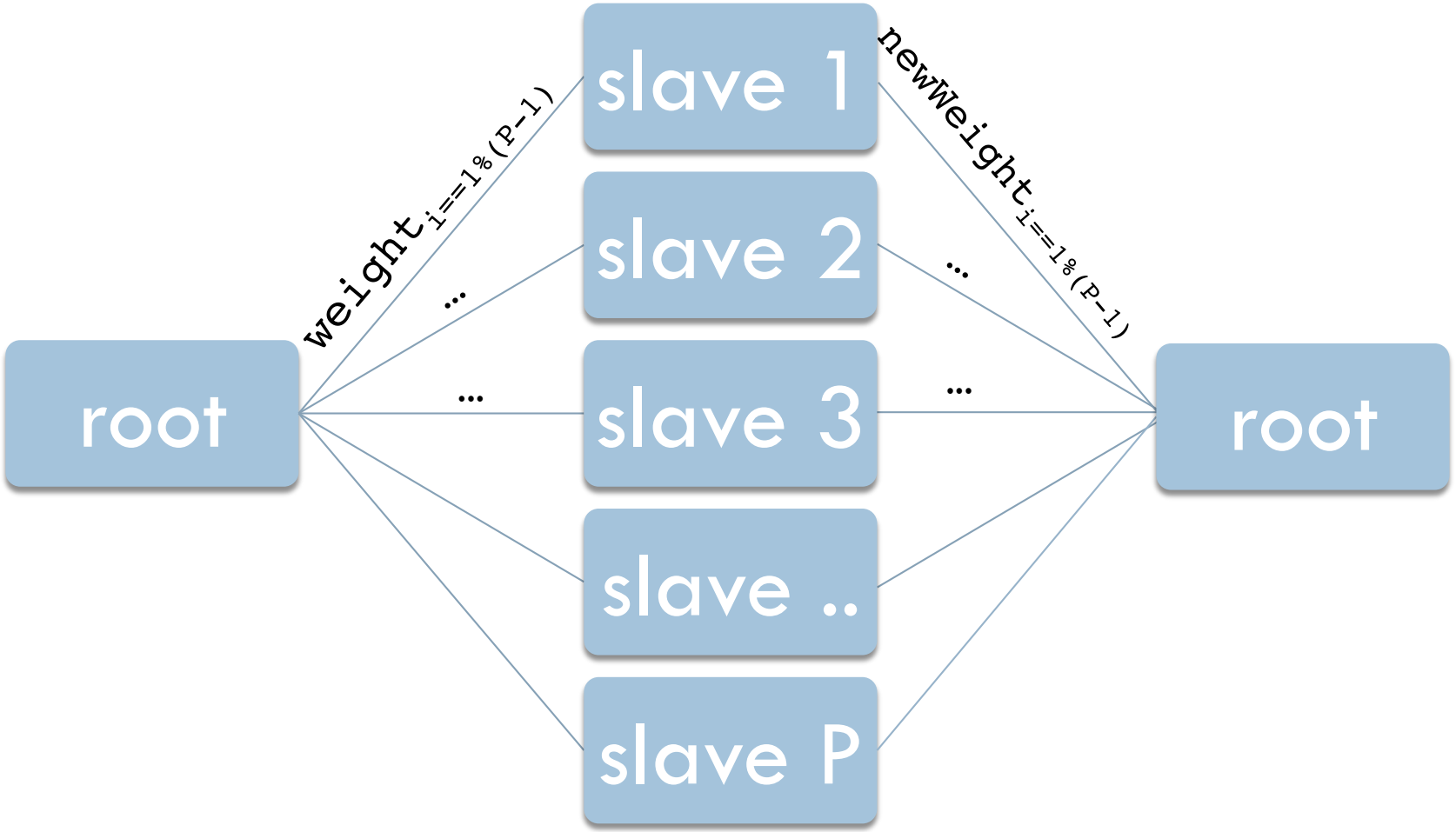


III. Implémentation finale

- Objet **`Classifier`**
  - contient les **`weights`**
  - permet la gestion de leurs indices

- Objet **`MPI_Manager`**
  - en charge de la mise à jour du **`Classifier`**
  - en charge de la parallélisation

III. Implémentation finale



III. Implémentation finale

```
for (int c = 0; c < numCaracs; c++) {
  // std::cout << "proc "<<taskid<<" ; i = "<<i << " ; j = "<<j << " ; c = "<<c << std::endl;

  multiplicateur_y = 0;
  while (caracs[c].getHauteur(multiplicateur_y) <= dimy) {
    multiplicateur_x = 0;
    while (caracs[c].getLargeur(multiplicateur_x) <= dimx) {
      // attention au boulot de root
      if (taskid == root) {
        if(!classifieur)
        {std::cout << "Classifieur vide : "<<taskid << std::endl;}

        weights[0] = *(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,1));
        weights[1] = *(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,2));
        // if(weights[0] != 1 || weights[1] != 0)
        //  std::cout << "WTF : "<< weights[0] << " , "<< weights[1] << std::endl;
        MPI_Isend(weights,2, MPI_DOUBLE, (compteur % (numtasks-1)) +1, 0,MPI_COMM_WORLD,&request);
      }
      /*********************/
      /****ENVOI DU SLAVE****/
      /*********************/

      else if ((compteur % (numtasks-1)) +1 == taskid) {
        MPI_Recv(weights, 2, MPI_DOUBLE, root, 0,MPI_COMM_WORLD, &status);
        valeur = img->traiterUnite(x, y, multiplicateur_x,multiplicateur_y, &(caracs[c]));
        /*cout << c << " " << i << " " << j
              << " " << multiplicateur_x
              << " " << multiplicateur_y << " : "
              << valeur << endl;*/
        newWeights(weights, valeur);
        MPI_Send(weights, 1, MPI_DOUBLE, root, 0,MPI_COMM_WORLD);
        MPI_Send(weights+1, 1, MPI_DOUBLE, root, 1,MPI_COMM_WORLD);
      }
      if(taskid == root){
        // std::cout << "Modif ? "<< *(classifieur->getWeights(c, x, y,
                                // multiplicateur_x, multiplicateur_y,1));
        MPI_Irecv(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,1),1, MPI_DOUBLE,(compteur % (numtasks-1)) +1, 0, MPI_COMM_WORLD,&request);
        // std::cout << " --> "<< *(classifieur->getWeights(c, x, y,
                                // multiplicateur_x, multiplicateur_y,1))<< std::endl;
        MPI_Irecv(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,2),1, MPI_DOUBLE,(compteur % (numtasks-1)) +1, 1, MPI_COMM_WORLD,&request);
      }
      compteur++;
      multiplicateur_x++;
```

III. Implémentation finale

```
multiplicateur_y = 0;
while (caracs[c].getHauteur(multiplicateur_y) <= dimy) {
  multiplicateur_x = 0;
  while (caracs[c].getLargeur(multiplicateur_x) <= dimx) {
    // attention au boulot de root
    if (taskid == root) {
      if(!classifieur)
      {std::cout << "Classifieur vide : "<<taskid << std::endl;}

      weights[0] = *(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,1));
      weights[1] = *(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,2));
      // if(weights[0] != 1 || weights[1] != 0)
      //  std::cout << "WTF : "<< weights[0] << " , "<< weights[1] << std::endl;
      MPI_Isend(weights,2, MPI_DOUBLE, (compteur % (numtasks-1)) +1, 0,MPI_COMM_WORLD,&request);
    }
    /**********************/
```

III. Implémentation finale

```
        /**********************/

        else if ((compteur % (numtasks-1)) +1 == taskid) {
          MPI_Recv(weights, 2, MPI_DOUBLE, root, 0,MPI_COMM_WORLD, &status);
          valeur = img->traiterUnite(x, y, multiplicateur_x,multiplicateur_y, &(caracs[c]));
          /*cout << c << " " << i << " " << j
                << " " << multiplicateur_x
                << " " << multiplicateur_y << " : "
                << valeur << endl;*/
          newWeights(weights, valeur);
          MPI_Send(weights, 1, MPI_DOUBLE, root, 0,MPI_COMM_WORLD);
          MPI_Send(weights+1, 1, MPI_DOUBLE, root, 1,MPI_COMM_WORLD);
      }
```

III. Implémentation finale

```
if(taskid == root){
  // std::cout << "Modif ? "<< *(classifieur->getWeights(c, x, y,
                              // multiplicateur_x, multiplicateur_y,1));
  MPI_Irecv(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,1),1, MPI_DOUBLE,(compte
  // std::cout << " --> "<< *(classifieur->getWeights(c, x, y,
                              // multiplicateur_x, multiplicateur_y,1))<< std::endl;
  MPI_Irecv(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,2),1, MPI_DOUBLE,(compte
}
compteur++;
multiplicateur_x++;
```

III. Implémentation finale

```
for (int c = 0; c < numCaracs; c++) {
  // std::cout << "proc "<<taskid<<" ; i = "<<i << " ; j = "<<j << " ; c = "<<c << std::endl;

  multiplicateur_y = 0;
  while (caracs[c].getHauteur(multiplicateur_y) <= dimy) {
    multiplicateur_x = 0;
    while (caracs[c].getLargeur(multiplicateur_x) <= dimx) {
      // attention au boulot de root
      if (taskid == root) {
        if(!classifieur)
        {std::cout << "Classifieur vide : "<<taskid << std::endl;}

        weights[0] = *(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,1));
        weights[1] = *(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,2));
        // if(weights[0] != 1 || weights[1] != 0)
        //  std::cout << "WTF : "<< weights[0] << " , "<< weights[1] << std::endl;
        MPI_Isend(weights,2, MPI_DOUBLE, (compteur % (numtasks-1)) +1, 0,MPI_COMM_WORLD,&request);
      }
      /*********************/
      /****ENVOI DU SLAVE****/
      /*********************/

      else if ((compteur % (numtasks-1)) +1 == taskid) {
        MPI_Recv(weights, 2, MPI_DOUBLE, root, 0,MPI_COMM_WORLD, &status);
        valeur = img->traiterUnite(x, y, multiplicateur_x,multiplicateur_y, &(caracs[c]));
        /*cout << c << " " << i << " " << j
              << " " << multiplicateur_x
              << " " << multiplicateur_y << " : "
              << valeur << endl;*/
        newWeights(weights, valeur);
        MPI_Send(weights, 1, MPI_DOUBLE, root, 0,MPI_COMM_WORLD);
        MPI_Send(weights+1, 1, MPI_DOUBLE, root, 1,MPI_COMM_WORLD);
      }
      if(taskid == root){
        // std::cout << "Modif ? "<< *(classifieur->getWeights(c, x, y,
                                // multiplicateur_x, multiplicateur_y,1));
        MPI_Irecv(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,1),1, MPI_DOUBLE,(compteur % (numtasks-1)) +1, 0, MPI_COMM_WORLD,&request);
        // std::cout << " --> "<< *(classifieur->getWeights(c, x, y,
                                // multiplicateur_x, multiplicateur_y,1))<< std::endl;
        MPI_Irecv(classifieur->getWeights(c, i, j,multiplicateur_x, multiplicateur_y,2),1, MPI_DOUBLE,(compteur % (numtasks-1)) +1, 1, MPI_COMM_WORLD,&request);
      }
      compteur++;
      multiplicateur_x++;
```

III. Implémentation finale

- Implémentation de Boosting :
  - implémentation dans une classe `Classifier_Final`
  - contient les $\lambda_{k,j}$
  - update à chaque image de validation

  - parallélisation : comme MPI_Train

III. Implémentation finale

- Implémentation du Test
  - essais pour plusieurs seuils θ
  - on maximise le F-Score

MERCI DE VOTRE ATTENTION

Vincent BILLAUT – Marc THIBAULT