**Squelettisation des images binaires**

Table des matières

[1. Introduction 1](#_Toc32308984)

[2. Principes de l’algorithme de squelettisation 1](#_Toc32308985)

[3. Connectivité dans les images binaires 2](#_Toc32308986)

[La 8 connectivité 2](#_Toc32308987)

[La 4 connectivité 2](#_Toc32308988)

[4. Configurations des points essentiels 3](#_Toc32308989)

# Introduction

Dans certaines applications de reconnaissance des formes, notamment la reconnaissance de l’écriture manuscrite, la squelettisation des images binaires permet :

* De réduire considérablement la quantité d’information à traiter.
* De faciliter l’accès aux informations par codage des segments d’image (code de Freeman).
* De détecter facilement les points remarquables tels que les ‘bouts’, les éléments de segments et les ‘nœuds’.

# Principes de l’algorithme de squelettisation

* Balayage de l’image point par point.
* Pour chaque point noir une décision est prise de le garder ou de le mettre à blanc, en fonction de ses huit voisins.
* L’algorithme s’arrête lorsqu’un balayage complet de l’image n’a détecté aucun point noir à mettre à blanc.

Les squelettes ainsi obtenus sont composés uniquement de trois types de points :

1. Les bouts, ils ont un seul voisin noir
2. Les points appartenant à un segment, ils ont deux voisins noirs
3. Les nœuds, ont plus de deux voisins

# Connectivité dans les images binaires

Dans les images binaires définies sur un maillage carré, la notion de voisinage d’un point (connectivité) peut être définie de deux façons :

## La 8 connectivité

Pour la 8 connectivité, les 8 points marqués Vx sont voisins du point P.

|  |  |  |
| --- | --- | --- |
| V2 | V3 | V4 |
| V1 | P | V5 |
| V8 | V7 | V6 |

## La 4 connectivité

Pour la 4 connectivité, les 4 points marqués Vx sont voisins du point P.

|  |  |  |
| --- | --- | --- |
|  | V3 |  |
| V1 | P | V5 |
|  | V7 |  |

L’utilisation seule de la 8 connectivité conduit à un paradoxe.

Pour la 8 connectivité, les 4 points noirs sont voisins (connectés) ce qui conduit à dire que cette figure possède un seul ilot de noirs et 5 ilots de blancs. Cela est contradictoire, car selon la 8 connectivité les 5 points blancs sont connectés entrer eux (il existe un chemin les reliant).

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | P |  |
|  |  |  |

Pour résoudre ce paradoxe nous utiliserons la 4 connectivité pour les blancs et la 8 connectivité pour les noirs.

# Configurations des points essentiels

Pour la 8 connectivité chaque point noir possède 8 voisins soit 256 configurations possibles.

|  |  |  |
| --- | --- | --- |
| V2 | V3 | V4 |
| V1 | P | V5 |
| V8 | V7 | V6 |

Pour chacune des 256 configurations nous devons décider si le point noir central est essentiel ou inessentiel.

Nous chercherons les configurations de 8 voisins pour lesquels la suppression du point noir central ne modifie pas la topologie. Autrement dit ne modifie ni le nombre d’ilots de noirs ni le nombre d’ilots de blancs. La connectivité des blancs est évaluée selon la 4 connectivité et la connectivité des noirs selon la 8 connectivité.

Afin de ne pas perdre les « bouts » nous définirons les points noirs ayant les configurations de voisinage suivantes comme des points essentiels :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

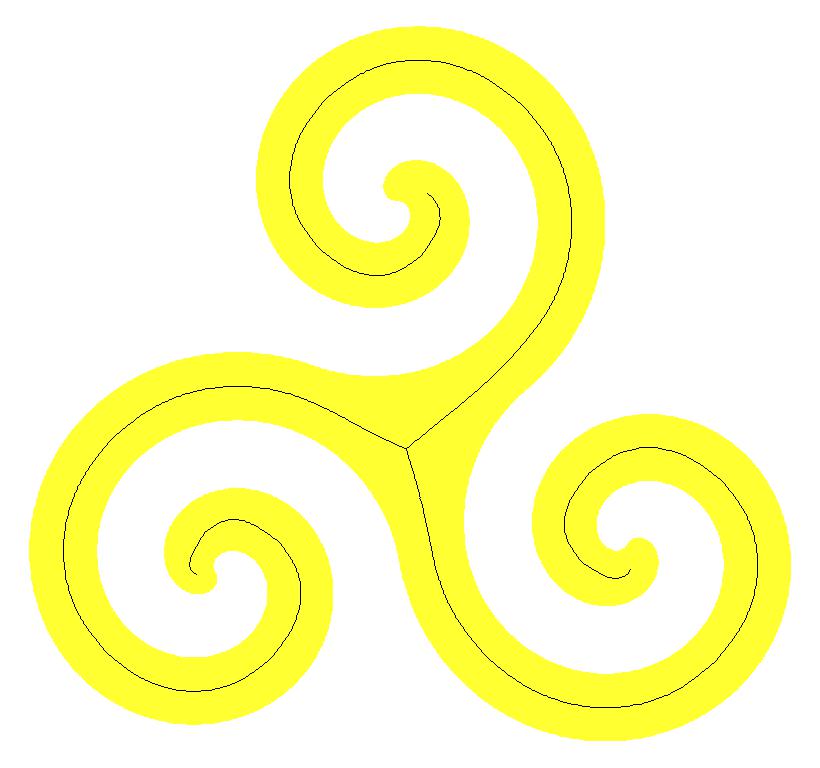
Ces définitions conduisent à identifier 16 configurations pour lesquelles le point central est inessentiel :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **P** |  |  |  |  | **P** |  |  |  |  | **P** |  |  |  |  | **P** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **P** |  |  |  |  | **P** |  |  |  |  | **P** |  |  |  |  | **P** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **P** |  |  |  |  | **P** |  |  |  |  | **P** |  |  |  |  | **P** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **P** |  |  |  |  | **P** |  |  |  |  | **P** |  |  |  |  | **P** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Ces configurations permettent de construire une table (vecteur) de 256 entrées à valeur ‘1’ pour les points essentiels et à valeur ‘0’ pour les points inessentiels.

Squelette du TRISKEL





# Programme Qt C++

QImage Squelette(QImage image)

{

bool fin = false;

int r,g,b,v,vb; //v contiendra l'indexe dans la table des configuration de voisins

char v0,v1,v2,v3,v4,v5,v6,v7;

char vb0,vb1,vb2,vb3,vb4,vb5,vb6,vb7;

int nb = 0;

int table[256] = {

0,0,0,1,0,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,1,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,

0,0,1,1,1,1,1,0,0,0,1,1,1,1,0,1,0,1,0,1,0,1,0,0,0,0,0,1,0,1,1,1,0,1,1,0,1,0,1,1,0,1,1,0,1,0,1,1,0,1,

0,1,0,1,0,0,0,0,0,1,0,1,1,1,0,1,1,0,1,0,1,1,0,1,1,0,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,1,

0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1,0,1,1,1,0,1,0,1,0,1,

0,0,0,0,0,1,0,1,1,1,0,1,1,0,1,0,1,1,0,1,1,0,1,0,1,1,0,1,0,1,0,1,0,0,0,0,0,1,0,1,1,1,0,1,1,0,1,0,1,1,

0,1,1,0,1,0};

cout << "H = " << image.height() << " L = " << image.width() << endl;

while (fin == false){

fin = true;

nb++;

for (int y = 1; y < image.height()-1; ++y){ //On décalle de 1 pour permettre l'analyse des 8 voisins

for (int x = 1; x < image.width()-1; ++x) {

// cout << "x = " << x << " y = " << y;

v = 0; //Mise à 0 de l'indexe dans la table des configuration de voisins

vb = 0;

QColor pxl(image.pixel(x,y));

r = pxl.red();

g = pxl.green();

b = pxl.blue();

if (r == 0){ //Si le point est noir

// cout << "x = " << x << " y = " << y << endl;

image.setPixel(x, y,qRgb(0,0,0)); //alors r=v=b=noir

QColor pxl1(image.pixel(x,y-1));

if (pxl1.red() == 0 ) {v=1; v0 ='X';} else v0 ='-'; //if V0 black

if (pxl1.blue() == 0) {vb=1; vb0 = 'X';} else vb0 = '-';

QColor pxl2(image.pixel(x+1,y-1));

if (pxl2.red() == 0) {v=v+2; v1 ='X';} else v1 ='-'; //if V1 black

if (pxl2.blue() == 0) {vb=vb+2; vb1 = 'X';} else vb1 = '-';

QColor pxl3(image.pixel(x+1,y));

if (pxl3.red() == 0) {v=v+4; v2 ='X';} else v2 ='-'; //if V2 black

if (pxl3.blue() == 0) {vb=vb+4; vb2 = 'X';} else vb2 = '-';

QColor pxl4(image.pixel(x+1,y+1));

if (pxl4.red() == 0) {v=v+8; v3 ='X';} else v3 ='-'; //if V3 black

if (pxl4.blue() == 0) {vb=vb+8; vb3 = 'X';} else vb3 = '-';

QColor pxl5(image.pixel(x,y+1));

if (pxl5.red() == 0) {v=v+16; v4 ='X';} else v4 ='-'; //if V4 black

if (pxl5.blue() == 0) {vb=vb+16; vb4 = 'X';} else vb4 = '-';

QColor pxl6(image.pixel(x-1,y+1));

if (pxl6.red() == 0) {v=v+32; v5 ='X';} else v5 ='-'; //if V5 black

if (pxl6.blue() == 0) {vb=vb+32; vb5 = 'X';} else vb5 = '-';

QColor pxl7(image.pixel(x-1,y));

if (pxl7.red() == 0) {v=v+64; v6 ='X';} else v6 ='-'; //if V6 black

if (pxl7.blue() == 0) {vb=vb+64; vb6 = 'X';} else vb6 = '-';

QColor pxl8(image.pixel(x-1,y-1));

if (pxl8.red() == 0) {v=v+128; v7 ='X';} else v7 ='-'; //if V7 black

if (pxl8.blue() == 0) {vb=vb+128; vb7 = 'X';} else vb7 = '-';

if (table[v] > 0 and //Detecton des bouts

(vb==2 or vb==8 or vb==32 or vb==128 or vb==1 or vb==4 or vb==16 or vb==64))

{image.setPixel(x, y,qRgb(r,255,50)); fin = false;}

if (table[v] > 0 and table[vb] > 0) { //Le point est inessentiel

image.setPixel(x, y,qRgb(r,255,50));

fin = false;

// cout << v7 << v0 << v1 << " " << vb7 << vb0 << vb1 << endl;

// cout << v6 << 'X' << v2 << " " << vb6 << 'X' << vb2 << endl;

// cout << v5 << v4 << v3 << " " << vb5 << vb4 << vb3 << endl << endl;

}

}

} //Fin de ligne

} //Fin de la passe

//Copie du resultat de la passe dans l'image originale

cout << "Fin de la passe: " << nb << "\n" << flush;

if (nb == 500) fin = true;

for (int y = 1; y < image.height()-1; ++y){ //On décalle de 1 pour permettre l'analyse des 8 voisins

for (int x = 1; x < image.width()-1; ++x){

QColor pxl(image.pixel(x,y));

r = pxl.red();

g = pxl.green();

b = pxl.blue();

if (g == 255) image.setPixel(x, y,qRgb(255,255,b));

}

}

QString fichier = "C:/Users/patri/Documents/PATRICK/ETUDES/Squelettisation/TRISKEL\_SQUELETTE"+QString::number(nb)+".jpg";

image.save(fichier);

// cout << "Enregistrement Nr " << nb << endl;

}

//Balayage du squelette pour repérer les bouts et le noeuds et les enregistrer

ofstream points\_remarquables ("C:/Users/patri/Documents/PATRICK/ETUDES/Squelettisation/points\_remarquables.txt");

for (int y = 1; y < image.height()-1; ++y){ //On décalle de 1 poureviter les bords

for (int x = 1; x < image.width()-1; ++x){

v = 0;

vb = 0;

QColor pxl(image.pixel(x,y));

r = pxl.red();

g = pxl.green();

b = pxl.blue();

if (r == 0){

image.setPixel(x, y,qRgb(0,0,0)); //alors r=v=b=noir

QColor pxl1(image.pixel(x,y-1));

if (pxl1.red() == 0 ) {v=1; v0 ='X';} else v0 ='-'; //if V0 black

if (pxl1.blue() == 0) {vb=1; vb0 = 'X';} else vb0 = '-';

QColor pxl2(image.pixel(x+1,y-1));

if (pxl2.red() == 0) {v=v+2; v1 ='X';} else v1 ='-'; //if V1 black

if (pxl2.blue() == 0) {vb=vb+2; vb1 = 'X';} else vb1 = '-';

QColor pxl3(image.pixel(x+1,y));

if (pxl3.red() == 0) {v=v+4; v2 ='X';} else v2 ='-'; //if V2 black

if (pxl3.blue() == 0) {vb=vb+4; vb2 = 'X';} else vb2 = '-';

QColor pxl4(image.pixel(x+1,y+1));

if (pxl4.red() == 0) {v=v+8; v3 ='X';} else v3 ='-'; //if V3 black

if (pxl4.blue() == 0) {vb=vb+8; vb3 = 'X';} else vb3 = '-';

QColor pxl5(image.pixel(x,y+1));

if (pxl5.red() == 0) {v=v+16; v4 ='X';} else v4 ='-'; //if V4 black

if (pxl5.blue() == 0) {vb=vb+16; vb4 = 'X';} else vb4 = '-';

QColor pxl6(image.pixel(x-1,y+1));

if (pxl6.red() == 0) {v=v+32; v5 ='X';} else v5 ='-'; //if V5 black

if (pxl6.blue() == 0) {vb=vb+32; vb5 = 'X';} else vb5 = '-';

QColor pxl7(image.pixel(x-1,y));

if (pxl7.red() == 0) {v=v+64; v6 ='X';} else v6 ='-'; //if V6 black

if (pxl7.blue() == 0) {vb=vb+64; vb6 = 'X';} else vb6 = '-';

QColor pxl8(image.pixel(x-1,y-1));

if (pxl8.red() == 0) {v=v+128; v7 ='X';} else v7 ='-'; //if V7 black

if (pxl8.blue() == 0) {vb=vb+128; vb7 = 'X';} else vb7 = '-';

if (v==2 or v==8 or v==32 or v==128 or v==1 or v==4 or v==16 or v==64){

cout << "Point v = " << v << " x = " << x << " y = " << y << endl;

points\_remarquables << "v = " << v << " x = " << x << " y = " << y << endl;

}

}

} //Fin de ligne

} //Fin d'image

points\_remarquables.close();

return image;

}