```
// Structure pour représenter une région (ou un nœud) dans le quadtree
typedef struct QuadtreeNode {
   int x, y;          // Coordonnées du coin supérieur gauche de la région
   int width, height;  // Dimensions de la région
   int isLeaf;        // Booléen indiquant si la région est homogène (feuille)
   struct QuadtreeNode* children[4]; // Pointeurs vers les 4 sous-régions
} QuadtreeNode;

// Fonction pour créer un nouveau nœud du quadtree
QuadtreeNode* create_node(int x, int y, int width, int height, int isLeaf) {
   QuadtreeNode* node = (QuadtreeNode*)malloc(sizeof(QuadtreeNode));
   node->x = x;    node->y = y;    node->width = width;    node->height = height;
   node->isLeaf = isLeaf;
   printf("-Creation node %d,%d,%d,%d \n",x,y,width,height);
   for (int i = 0; i < 4; i++)
         node->children[i] = NULL;
   return node;
}

// Critère d'homogénéité : si la différence de couleur dépasse un seuil, la région n'est pas homogène
int is_homogeneous(int x, int y, int width, int height) {
   if ( width > 25 || height > 25)       return 0;
   else        return 1;
}

// Fonction récursive pour construire le quadtree
QuadtreeNode* build_quadtree(int x, int y, int width, int height) {
   QuadtreeNode* node = create_node(x, y, width, height, 1);
   if (is_homogeneous(x, y, width, height) == 0) {
      node->isLeaf = 0;
      node->children[0] = build_quadtree( x, y, width / 2, height / 2);                    // Nord-ouest
      node->children[1] = build_quadtree( x + width / 2, y, width / 2, height / 2);        // Nord-est
      node->children[2] = build_quadtree( x, y + height / 2, width / 2, height / 2);       // Sud-ouest
      node->children[3] = build_quadtree( x + width / 2, y + height / 2, width / 2, height / 2); // Sud-est
   }
   return node;
}
```

```
// Fonction pour libérer la mémoire du quadtree
void free_quadtree(QuadtreeNode* node) {
    if (node==NULL) return;
    for (int i = 0; i < 4; i++) {
        free_quadtree(node->children[i]);
    }
    free(node);
}


int main() {
    // Charger l'image avec OpenCV

    // Construire le quadtree pour l'image complète
    QuadtreeNode* root = build_quadtree(0, 0, 200, 200);

    // Libérer la mémoire du quadtree
    free_quadtree(root);

    return 0;
}
```