Ministère de l'enseignement supérieur et de la recherche scientifique

Faculté des Sciences de Monastir

Mohamed Chaker Jouini

-LSI1 : TD3 TP6

-Identificateur des objets dans une image par comparaison des valeurs RGB

INTRODUCTION GÉNÉRALE

INTRODUCTION GÉNÉRALE

Ici vous allez écrire une introduction générale sur le domaine choisi.
Ecriture en times new roman , taille 12.
-Le domaine du traitement d'image concerne la manipulation, l'analyse et la compréhension des images. Ce qui inclus un large éventail d'applications dans le monde : de l'amélioration des image ou leur compression jusqu'à la vision par ordinateur (computer vision) ou la réalité augumenté (AR) et meme la médicine diagnostique.
-Divers logiciels peuvent etre utiliser chacune a un objecitf et utilité spécifique, notamment les logiciels du calcule matricielle (comme MATLAB ou Ocatve) qui interagie avec l'image d'un point de vue mathématique qui est explicite dans la notation de RGB ; la représentation des couleurs par un triplet de valeurs entre 0 et 255.
-Octave est un logiciel Open Source paralléle à MATLAB qui est propriétaire. Les deux offre une synatxe conviviale pour effectuer des calculs mathématique complexe, des simulations et d'autre fonctionnalités avancées qu'on peut utiliser pour traiter les images.
-Les scripts d'octave et matlab sont d'extensions (.m) et synatxe assez similaire donc la différence de fonctionnalité ne nous pose aucun probleme puisque aucun « Toolbox » spécifique à matlab était utilisé.

Pour plus d'informations se référer à https://docs.octave.org/latest/

CHAPITRE 1 :

Partie Théorique

CHAPITRE 1:
PARTIE THEORIQUE

Dans cette partie vous allez faire une petite recherche sur le domaine que vous avez choisi.

-Le modèle RGB (Red,Green,Blue) ou RGV est une représentation colormétrique ou les couleurs sont créés par le mélange de trois canaux de couleurs primaire : rouge vert et bleu.

-Chaque pixel dans une image est donc définit par un mélange de ces trois couleurs ou chacun est représenté par un entier allant de 0 à 255 dans un espace de 8bits/canal.

-Cette représentation nous permet de convertir une image à une matrice à 3 dimensions , le troixieme contient le triplet de valeurs et les deux autres dimensions varie de taille par à rapport à l'image.

-On peut donc utiliser des logiciels de calculs mathématique pour manipuler ces matrices (images) : on peut comparer les valeurs RGB de deux pixels adjacents pour déterminer s'ils se ressemblent ou non.

-On peut utiliser un variable « limite » pour controler cette « ressemblance » : limite égale zéro signifie deux pixels de meme couleurs (exactement le meme couleurs), et un limite égale à 255 signifie que les deux pixels « se ressemblent » quelque soit les couleurs, ce qui n'est pas très utile donc on limite ce « limite » à l'intervale [0;100] ou peut etre [0;150].

NB : le variable limite est un entier , ainsi que les valeurs RGB de chaque pixels.

-Donc On parcours les pixels de l'image en comparant les pixels et on les comaprent horizontalement, verticalement ou meme par des ensembles de pixels pour des résultats plus exacte mais à raision de simplicité on va juste comparer les pixels adjacents horizontalements.

-Aprés la comparaison on peut enregistrer les indices des pixels similaires , et pour les visualiser on parcours l'image une autre fois et chnager leurs valeurs RGB à 0 (couleur noir), ce qui n'est pas idéale si l'image contient initialement des pixels noirs.

-Ce qui nous empeche d'utilier des images sans couleur noir OU ceux qui ont un arrière plan d'un couleurs spécifique (blanc par exemple) et on ajoute une condition pour ne pas changer les pixels de ce couleurs.

-Alors On obtient théoriquement tous les objets dans l'image coloré en noir.

-Cela put etre utile dans le domaine de vision par ordinateur ou on doit identifier les objets dans une image.

CHAPITRE 2 :

Partie Pratique

CHAPITRE 2:
PARTIE PRATIQUE

A ce niveau vous allez donner votre proposition accompagnée par des images pour les différents résultats trouvés. Début :

```
img = imread("/***/****/try3.jpg"); %on choisit l’image
[rows, columns, ~] = size(img); %on prend les dimensions de l’image
figure;
imshow(img); %affichage de l’image avant le traitement
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%fonction pour comparer les valeurs RGB
function result= similaire(pixel1, pixel2)
  %variable de configuration du similarite acceptee dans le groupement
    limite = 100;
    %on calcule la differnce des valeurs
    diff = abs(pixel1 - pixel2);
    %et on les compare avec le variable de configuration
    result = all(diff <= limite);
endfunction

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%groupement des indices des pixels a couleurs similaires
%pour le moment la comparaison et seulement faite horizentalement
cluster = [];

%iteration des pixels de l'image
for i = 1:rows
  for j = 1:(columns-1)
    %comparaison des pixels adjacents(meme ligne)
    if similaire(img(i,j,:), img(i,j+1,:))
```

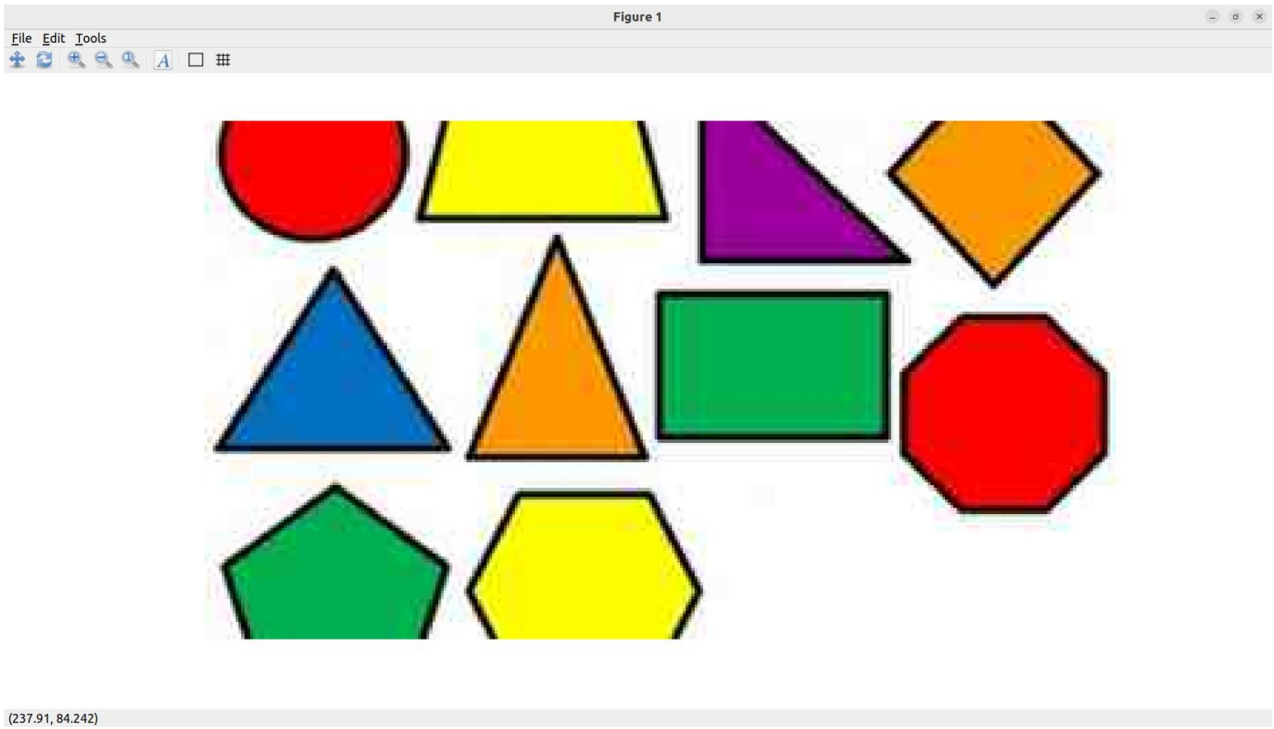
```
      %on utilise if pour que le j+1 ne soit pas "out of range"
      if j+1 <= columns
        if similaire(img(i, j, :), img(i, j+1, :))
          %on ajoute les indices des pixels similaires
          cluster = [cluster; i, j; i, j+1];
        endif
      endif
    endif
  endfor
endfor

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%identifier les groupements par les donner des valeurs RGB nuls
for k = 1:2:size(cluster, 1)
 %on ne change pas les pixels blancs (valeurs rgb=255)
  if img(cluster(k, 1), cluster(k, 2), :) ~= 255 && img(cluster(k+1, 1), cluster(k+1, 2), :) ~= 255
   %on donne les pixels similaires un couleur noire
    img(cluster(k, 1), cluster(k, 2), :) = 0;
    img(cluster(k+1, 1), cluster(k+1, 2), :) = 0;
  endif
endfor
figure;
imshow(img); %affichage apres le traitement
```

Les Résultats :

IMAGE ORIGINALE :



LIMITE = 5 :

LIMITE=10:

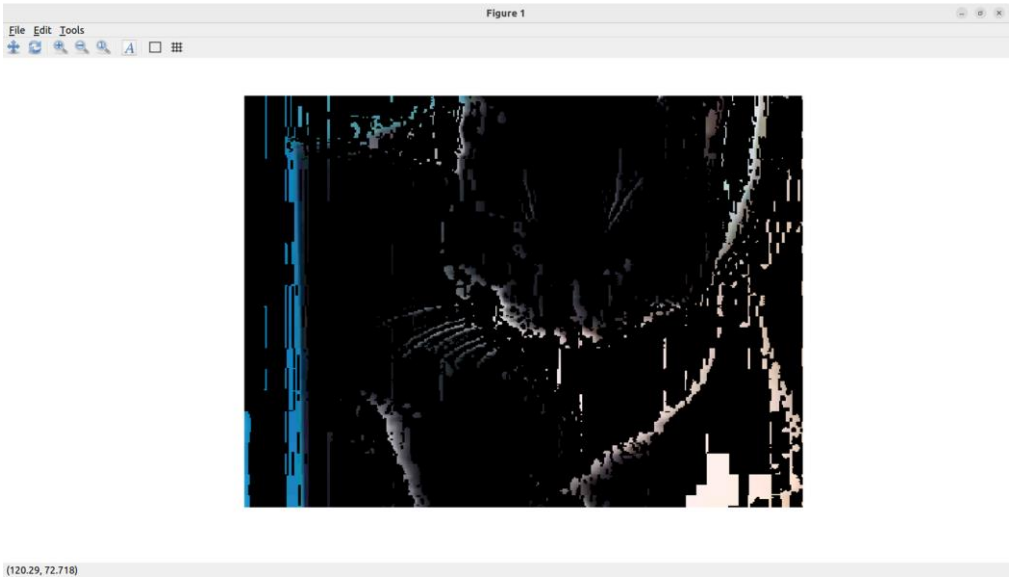
LIMITE= 50:

LIMITE=150:

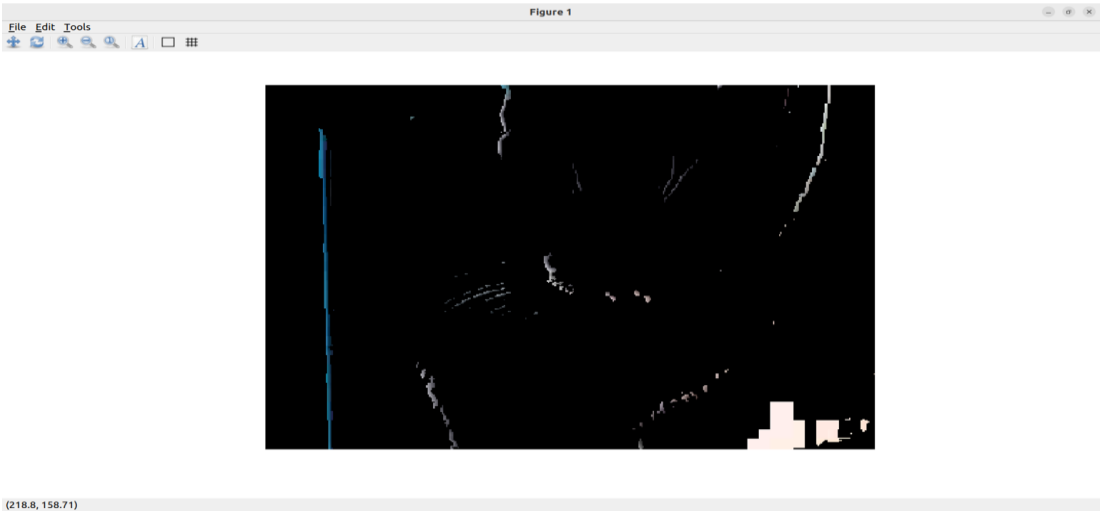
LIMITE=255 (valeur maximale)

AUTRE IMAGE:

LIMITE=1:



LIMITE=10:



CONCLUSION GÉNÉRALE

CONCLUSION GÉNÉRALE

Pour la conclusion vous aller développer une conclusion sur le travail fait.

-Cette solution pour identifier les objets n'est pas générale et n'est pas effectif que dans des cas bien définis: Arriére plan de meme couleur qui sépare les objets , les paramétres de ce couleur doivent etre configurer dans la condition d'exclusion à la fin du programme.

-L'image du chat était presque totalement convertit en noir puisque il y a des bandes continue de couleurs ; ce qui peut etre "reparer" par convertir à des différentes couleurs (on a converti tous les pixels semblables en noir) et pour distinguer les couleur changer on désigne un tableau de chnagement : par exemples si il y a des pixels rouges on les convertis en bleu , et les bleu en violet ... Puis on fait compare les pixels originales par les pixels modifier (visuellement).

-Ce n'est pas vraiment nécessaire pour le programme lui meme puisqu'on a déja enregistrer tous les indices des couples de pixels similaires , pour distinguer les différentes objets on peut stocker dans un tableau les indices d'un couleur bien donné et qui ajoute les indics des pixels de meme couleurs et qui sont "proches", exemple : tableau 1 contient indice (133,465) de couleur orangé, le programme trouve un autre pixels orangé(différente variation mais accepte par le variable limite) dans (134,468) et un autre dans (200,105), on ajoute seulement le premier puisqu'il est proche donc on peut dire c'est le meme objet.Bien sur c'est mieux d'avoir un repere qui respect les dimensions des images pour bien définir le mot "proche".