CS 558: Optional Projects

Stevens ID: 10430965

Name: Yanyan Jiang

***Problem 1: Image Data Association***

***Source code:***

clear all;

bin=10;

disp(['When the number of bins in the histograms is ' num2str(bin) ':']);

trainData=zeros(29,bin\*3);

testData=zeros(29,bin\*3);

trainnum=1; %number of train image

for i=1:29

% train image histogram

train=imread(['ImClasss/' 'train' num2str(i) '.jpg']);

trainR=histogram(train(:,:,1),bin);

trainG=histogram(train(:,:,2),bin);

trainB=histogram(train(:,:,3),bin);

trainData(trainnum,:)=[trainR trainG trainB];

% trainlabel(trainnum)=i;

trainnum=trainnum+1;

end

testnum=1; %number of test image

for i=1:4

% test image histogram

test=imread(['ImClasss/' 'test' num2str(i) '.jpg']);

testR=histogram(test(:,:,1),bin);

testG=histogram(test(:,:,2),bin);

testB=histogram(test(:,:,3),bin);

testData(testnum,:)=[testR testG testB];

% testlabel(testnum)=j;

testnum=testnum+1;

end

% referance:https://blog.csdn.net/hjimce

% mean shift cluster

radius=1500000; %search window radius

threshold=1e-3\*radius; % 1500 % threshold for stop judgment

trainIdx = MS(trainData,radius,threshold);

testIdx = MS(testData,radius,threshold);

%display the result

for i=1:29

disp(['Train image ' num2str(i) ' has been assigned to class ' num2str(trainIdx(i)) '.']);

end

for i=1:29

disp(['Test image ' num2str(i) ' has been assigned to class ' num2str(testIdx(i)) '.']);

end

***Mean-Shift function:***

%reference://blog.csdn.net/hjimce

function Idx = MS(data,radius,threshold)

%mean shift

[m,n] = size(data);

index = 1:m; %cluster category

visit\_record = zeros(m,1); % mark the visited point

count = [];

cluster\_num = 0;

cluster\_center = [];

while ~isempty(index)

%choose a random point that didn't visited as a center

center = data(index(ceil((length(index)-1e-6)\*rand)),:);

visit\_probability = zeros(m,1);%the probability of point visit

while 1

%the inside points of the search window

dis = sum((repmat(center,m,1) - data).^2,2);

innerS = find(dis<(radius\*radius));

visit\_record(innerS) = 1; %record visited points

visit\_probability(innerS) = visit\_probability(innerS) + 1;

%compute the new center position

newcenter = zeros(1,n);

sumweight = 0;

for i = 1:length(innerS)

w = exp(dis(innerS(i))/(radius\*radius));

sumweight = w + sumweight;

newcenter = newcenter + w\*data(innerS(i),:);

end

newcenter = newcenter./sumweight;

% if the moving distance less than threshold, stop

if norm(newcenter - center) <threshold

break;

end

center = newcenter;

end

% determine if the new center need to merge, if not?the number of clusters + 1

merge = 0;

for i = 1:cluster\_num

if norm(center - cluster\_center(i,:)) < radius/2

merge = i;

break;

end

end

if merge == 0 % a new cluster

cluster\_num = cluster\_num + 1;

cluster\_center(cluster\_num,:) = center;

count(:,cluster\_num) = visit\_probability;

else % merge

cluster\_center(merge,:) = 0.5\*(cluster\_center(merge,:) + center);

count(:,merge) = count(:,merge) + visit\_probability;

end

%recount points that have not been visited

index = find(visit\_record == 0);

end

%the result of cluster

for i = 1:m

[~, index] = max(count(i,:));

Idx(i) = index;

end

end

***Histogram function:***

function result=histogram(image,bin)

image=double(image(:));

for i=1:bin

if i==1

result(i)=size(find(image<256/bin\*i),1);

else

result(i)=size(find(image<256/bin\*i),1)-sum(result(1:i-1));

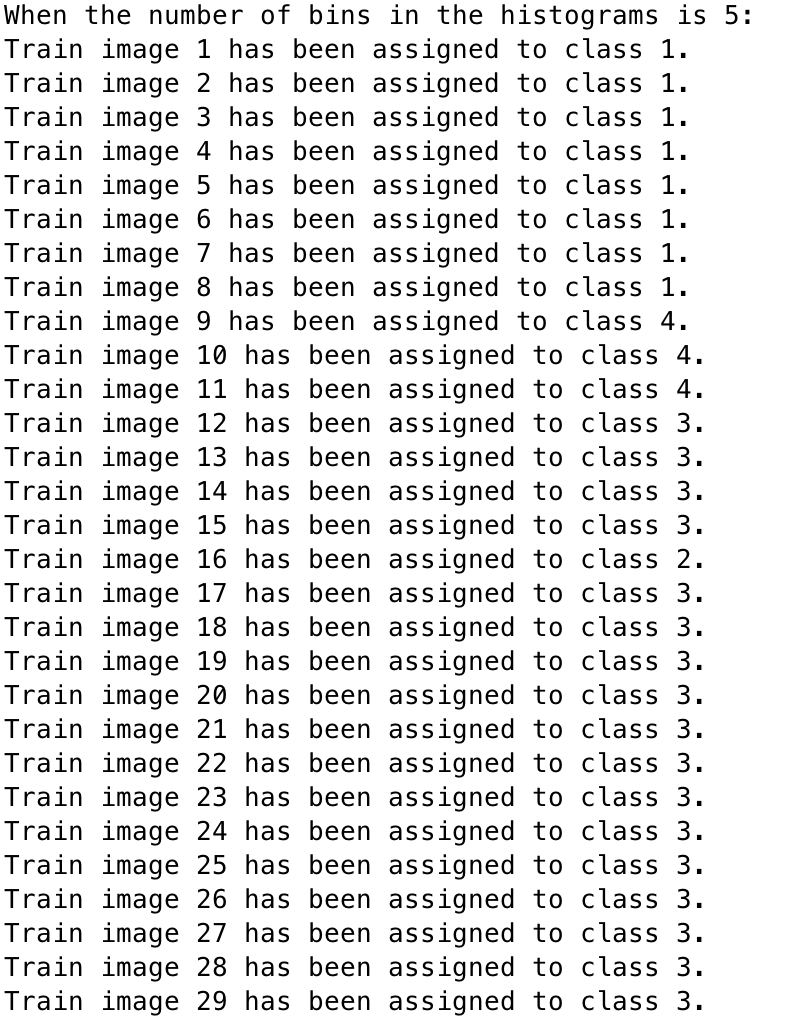
end

end

end

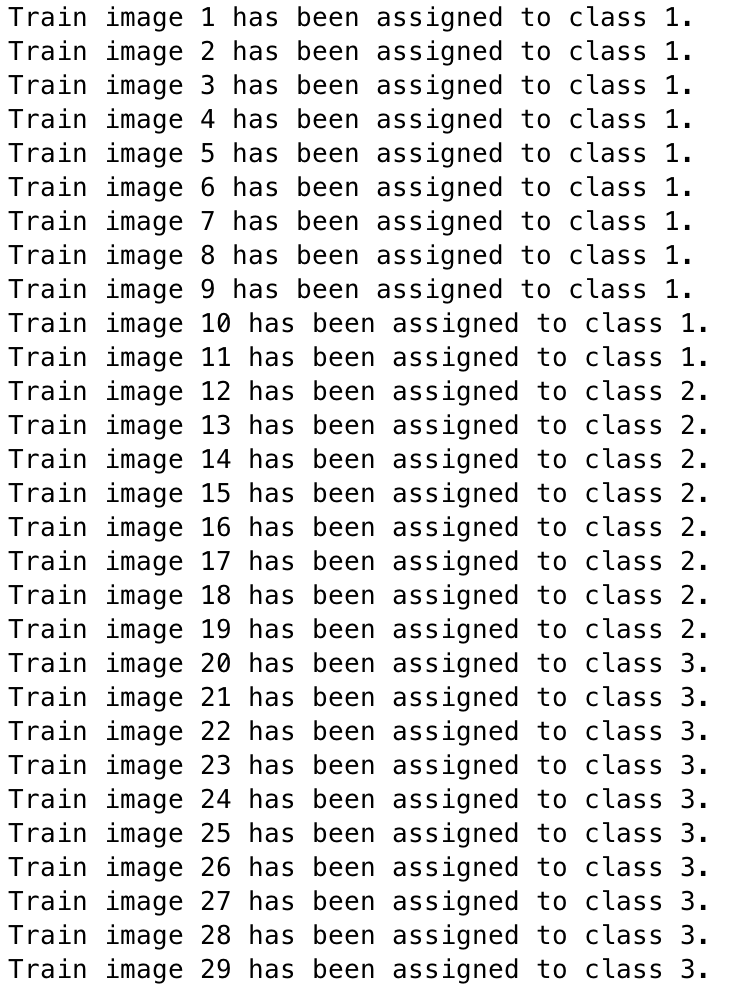
***Result:***

I still choose the histogram as my image descriptor like the HW4. The histogram can clearly and accurately describe the color information of the image. I tried several times to change the number of bins in the histograms like 5, 10, 20. Then I find the 10 can show the best result.

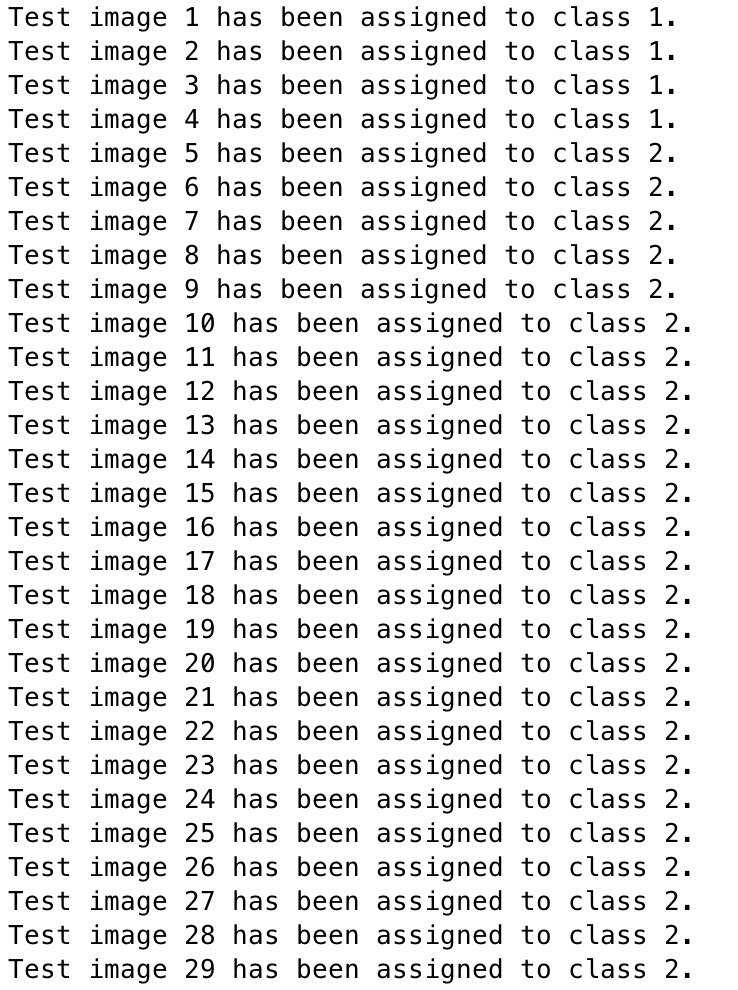
(I just show 5 here.) 

I also tried different threshold and the radius of mean-shift search window. Finally, I choose 1500000 as the radius and the threshold is about between 1000 and 1500. I also find a calculation formula from internet that show the relation of the best threshold choose and the radius.

I input the image sort as Fountain-P11, Herz-Jesu-P8, entry-P10. You can see the result of the cluster is same as the fact.



Then perform the same clustering procedure of test image using the same parameters. (The sort of test image is Castle-P19, entry-P10)



I got the result shows the first 4 images is a cluster and others belong to another one. But because there is scene overlap among these datasets, the correct rate of clustering procedure is reduced a lot.

***Problem 2: Segmentation***

***Source code:***

clear;close all;clc;

% for i=1:19

% RGB = imread(['images/' 'castle' num2str(i) '.jpg']);

RGB = imread(['images/' 'castle' num2str(6) '.jpg']);

%convert frame from RGB to YCBCR colorspace

YCBCR = rgb2ycbcr(RGB);

%filter YCBCR image between values and store filtered image to threshold

%the green value in ycbcr

Y\_MIN = 30; Y\_MAX = 160;

Cb\_MIN = 53; Cb\_MAX = 128;

Cr\_MIN = 34; Cr\_MAX = 128;

threshold=roicolor(YCBCR(:,:,1),Y\_MIN,Y\_MAX)&roicolor(YCBCR(:,:,2),Cb\_MIN,Cb\_MAX)&roicolor(YCBCR(:,:,3),Cr\_MIN,Cr\_MAX);

%reference:https://blog.csdn.net/whuhan2013/article/details/53956606

%perform morphological operations on thresholded image to eliminate noise

%and emphasize the filtered object

erodeElement = strel('square', 3) ;

dilateElement= strel('square', 12) ;

threshold = imerode(threshold,erodeElement);

threshold = imerode(threshold,erodeElement);

threshold=imdilate(threshold, dilateElement);

threshold=imdilate(threshold, dilateElement);

threshold=imfill(threshold,'holes');

%get the of basic property:'Area', 'Centroid', and 'BoundingBox'

stats = regionprops(threshold, 'basic');

[C,area\_index]=max([stats.Area]);

face\_locate=[stats(area\_index).Centroid(1),stats(area\_index).Centroid(2)];

I = imcrop(RGB,[face\_locate(1)-600,face\_locate(2)-400,1000,800]);% Clipping target box

I1 = rgb2hsv(I); % transform RGB to HSV

h = I1(:, :, 2); % S

bw = im2bw(h, graythresh(h)); % threshing

bw=imfill(bw,'holes');

erodeElement = strel('square', 3) ;

dilateElement=strel('square', 5) ;

bw = imerode(bw,erodeElement);

bw = imerode(bw,erodeElement);

bw=imdilate(bw, dilateElement);

bw=imdilate(bw, dilateElement);

bw1=imfill(bw,'holes');

bw1 = bwareaopen(bw1, 8000); % Filter collection with more than 8000 pixels

bw2 = cat(3, bw1, bw1, bw1); % model

I1 = I .\* uint8(bw2);

I2=RGB;

for i= 1:size(I2,1)

for j=1:size(I2,2)

if ~(j > face\_locate(1)-600 && j< face\_locate(1)-600+1000 && i>face\_locate(2)-400 &&i<face\_locate(2)-400+800)

I2(i,j,1) = 0;

I2(i,j,2) = 0;

I2(i,j,3) = 0;

else

I2(i,j,:)=I1(i-floor(face\_locate(2)-400),j-floor(face\_locate(1)-600),:);

end

end

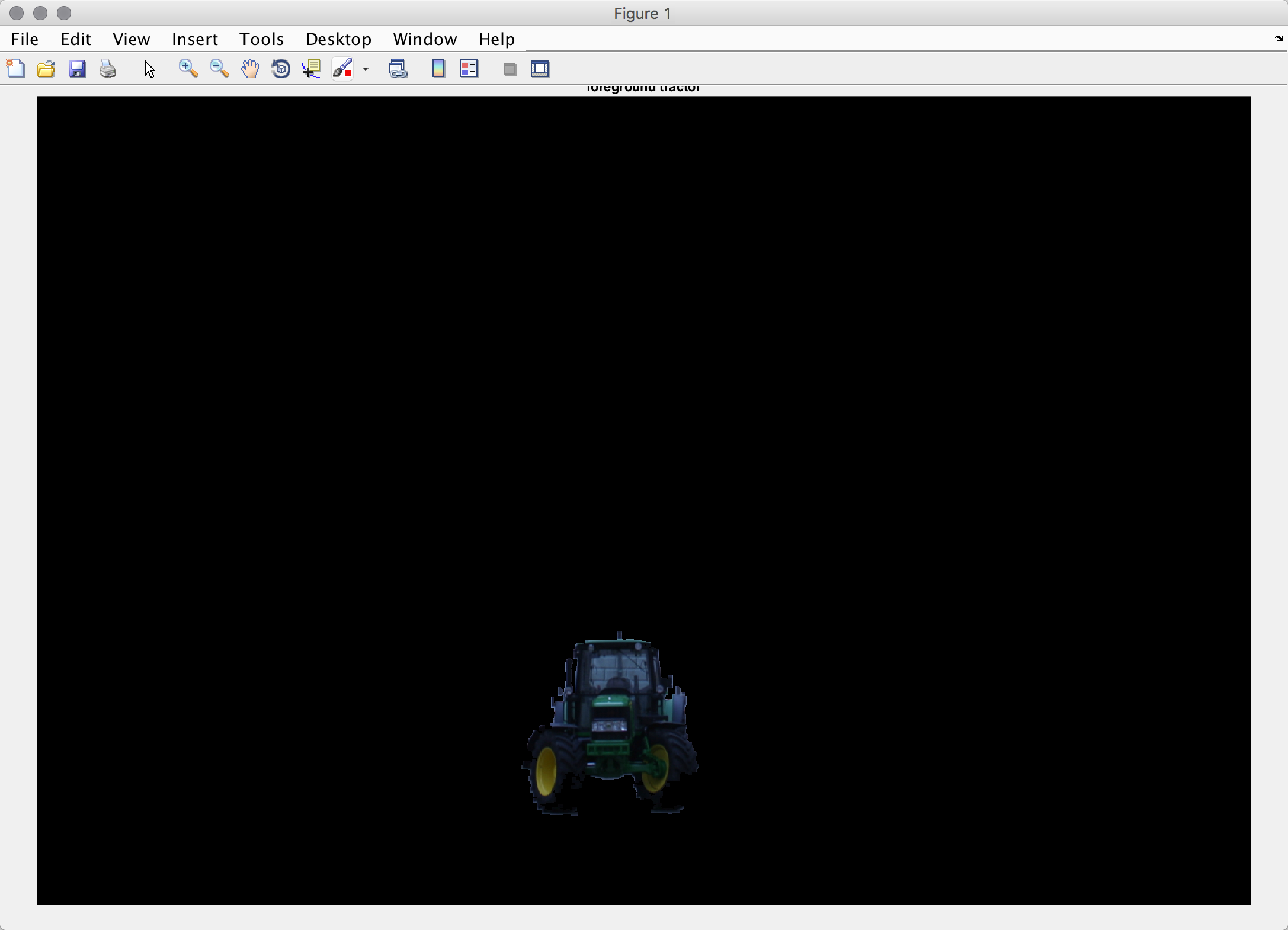
end

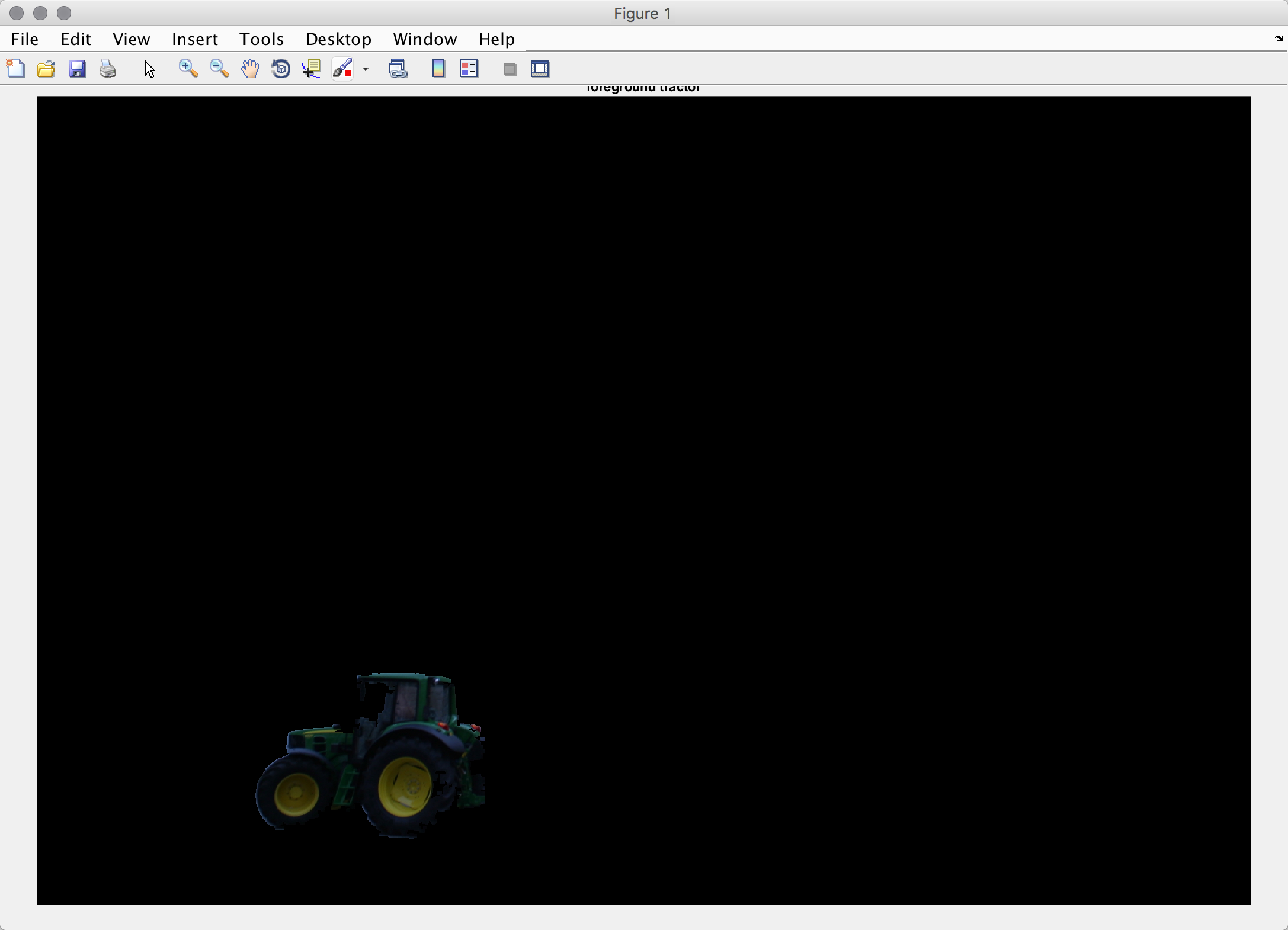
figure;

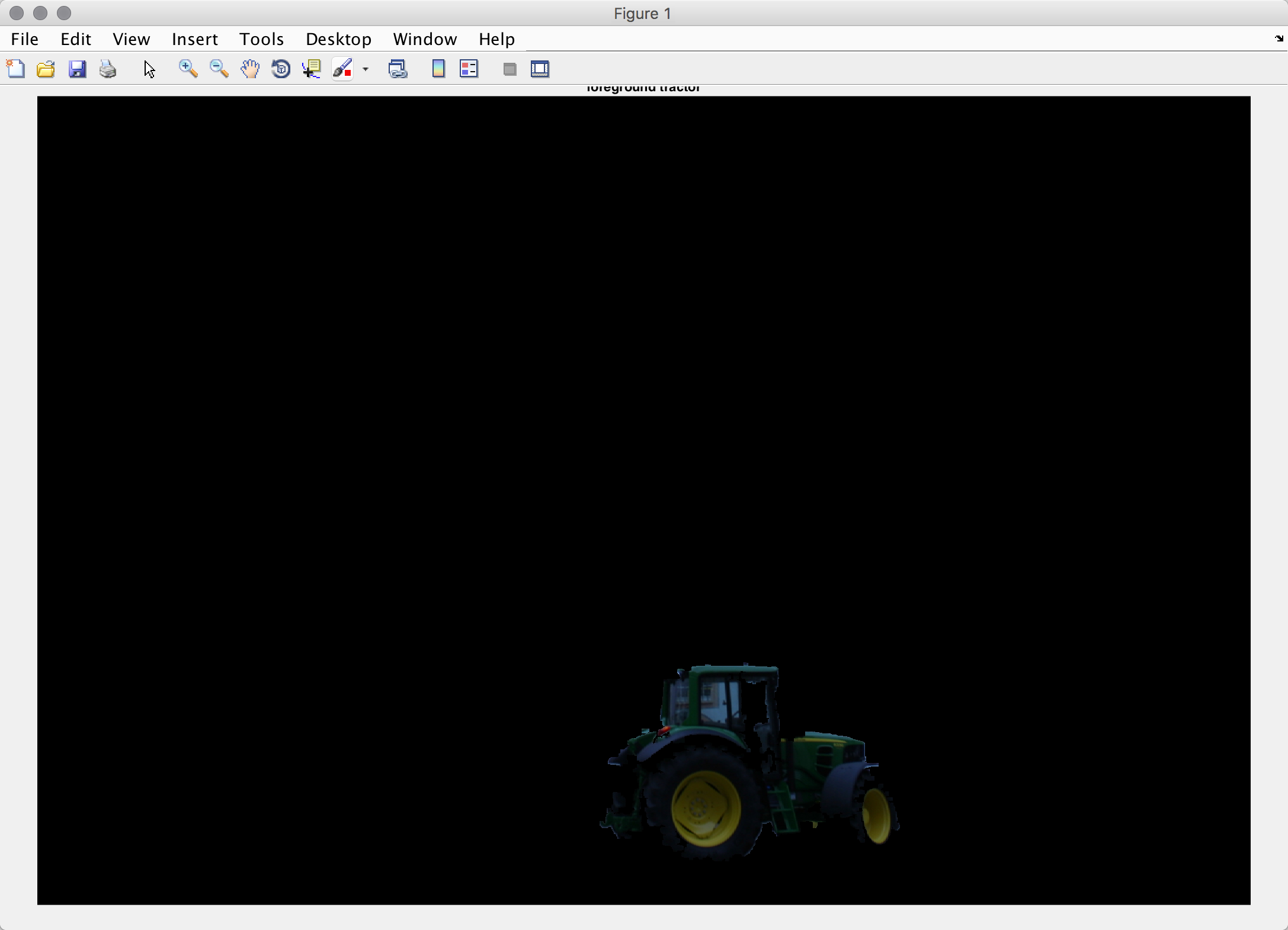
imshow(I2); title('foreground tractor');

% end

***Result:***







There are several better results of my work. I first artificially determined that the main color of the target object is green, and then perform the separation operation.

During the seek a balance between segmentation quality with the level of supervision process. The high level supervision is good for segmentation but it spent too much manpower. My framework is based on semi-supervised learning, needs to start the target selection color input and the target box threshold selection. Because of the limited time, my results are not perfect now, I hope I can improve it later.

***Problem 4:SNIC***

***Source code:***

im = imread('wt\_slic.png');

im = double(im);

[m,n,~]=size(im);

R = im(:,:,1);

G = im(:,:,2);

B = im(:,:,3);

% sz = m\*n;

S = 50;

K = round((m\*n) / (S\*S));

Center = zeros(K,5);

label = -ones(m,n);

distance = inf(m,n);

% queue = -ones{m\*n};

k=0;

head=1;

tail=1;

for i = 1:round(m/S)

for j = 1:round(n/S)

k = k+1;

Center(k,:)=[(i-1)\*S + S/2,(j-1)\*S + S/2,0,0,0];

queue(1,head)= {[Center(k,1),Center(k,2),k,0]};

head=head+1;

end

end

numk = K;

kR = zeros(1,numk);

kG = zeros(1,numk);

kB = zeros(1,numk);

kx = zeros(1,numk);

ky = zeros(1,numk);

ksize = zeros(1,numk);

CONNECTIVITY = 4; %values can be 4 or 8

M = 10;%compactness; %10.0;

invwt = (M\*M\*numk)/(m\*n);

% qlength = head;

pixelcount = 0;

while(tail~=head)

x = queue{1,tail}(1);

y = queue{1,tail}(2);

k = queue{1,tail}(3);

% [x,y,k,d] = queue(1,tail);

tail = tail+1;

if label(x,y) < 0

label(x,y) = k;

pixelcount = pixelcount +1;

kR(1,k) = kR(1,k) + R(x,y);

kG(1,k) = kG(1,k) + G(x,y);

kB(1,k) = kB(1,k) + B(x,y);

kx(1,k) = kx(1,k) + x;

ky(1,k) = ky(1,k) + y;

ksize(1,k) = ksize(1,k) + 1.0;

for ii = -1:1 %connectivity = 8

for jj = -1:1

xx = x + ii;

yy = y + jj;

if ii~=0 || jj~=0

if xx >= 1 && xx < m && yy >= 1 && yy < n

% ii = yy \* w + xx ;

if label(xx,yy) < 0 %//create new nodes

ldiff = kR(1,k) - R(xx,yy)\*ksize(1,k);

adiff = kG(1,k) - G(xx,yy)\*ksize(1,k);

bdiff = kB(1,k) - B(xx,yy)\*ksize(1,k);

xdiff = kx(1,k) - xx\*ksize(1,k);

ydiff = ky(1,k) - yy\*ksize(1,k);

colordist = ldiff\*ldiff + adiff\*adiff + bdiff\*bdiff;

xydist = xdiff\*xdiff + ydiff\*ydiff;

slicdist = (colordist + xydist\*invwt)/(ksize(1,k)\*ksize(1,k));

% //late normalization by ksize(k), to have only one division operation

queue(1,head)={[xx,yy,k,slicdist]};

head = head+1;

end

end

end

end

end

end

end

% if label(1,1) < 0

% label(1,1) = 0;

% end

% for y = 2:n

% for x = 2:m

% i = y\*m+x;

% if label(x,y) < 0

% if label(x-1,y) >= 0

% label(x,y) = label(x-1,y);

% else label(x,y-1) >= 0

% label(x,y) = label(x,y-1);

% end

% end

% end

% end

for k=1:K

for i=2:m-1

for j=1:n-1

if label(i,j) ~= label(i,j+1) || label(i,j) ~= label(i+1,j)

im(i,j,1)= 0;

im(i,j,2)= 0;

im(i,j,3)= 0;

end

end

end

end

***Result:***

I’m sorry I didn’t finish this job. I hope you can see my code. I think the steps is correct.