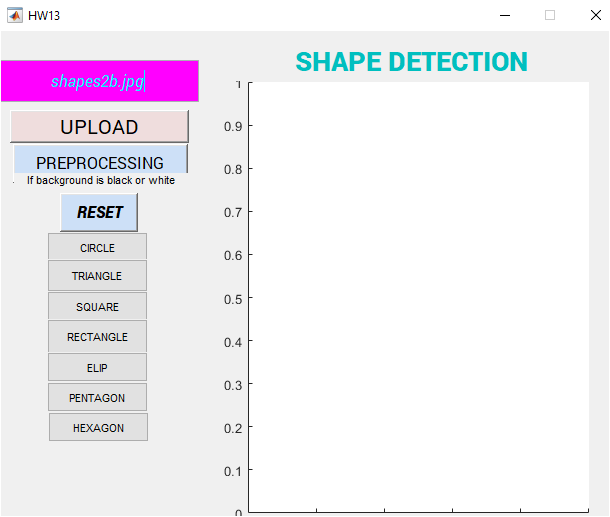
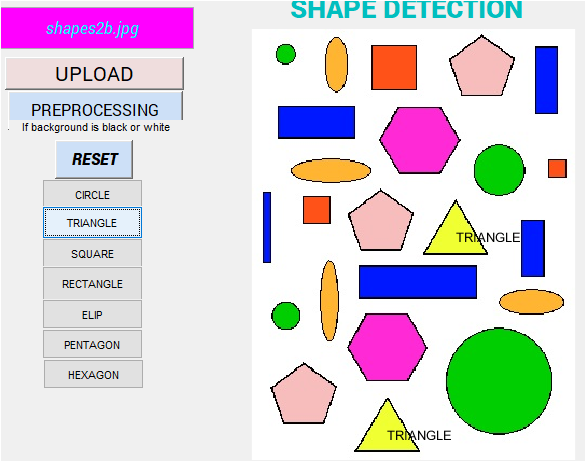
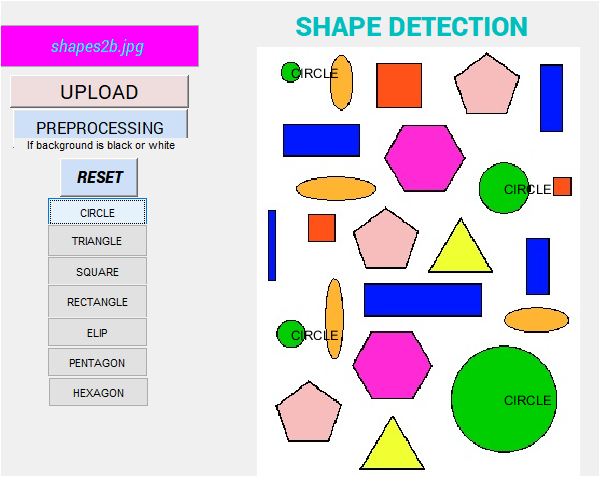
Đinh Hoàng Sáng

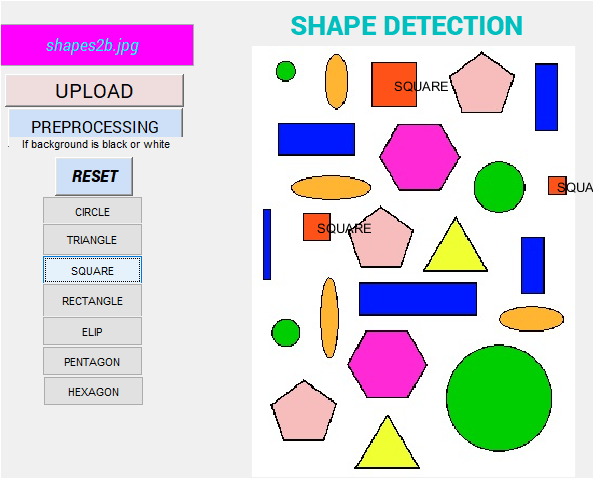
BEBEIU17022

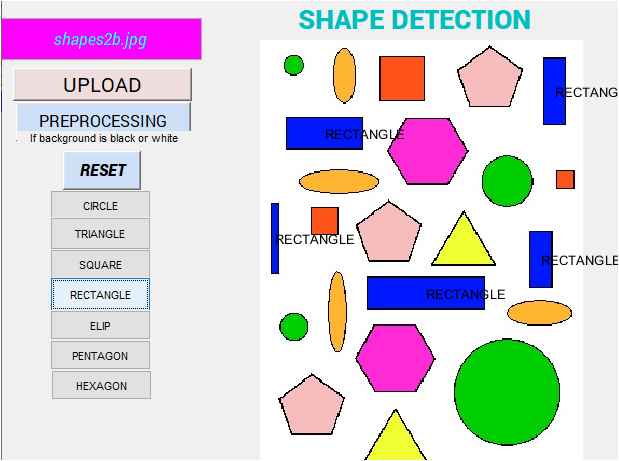
Result:

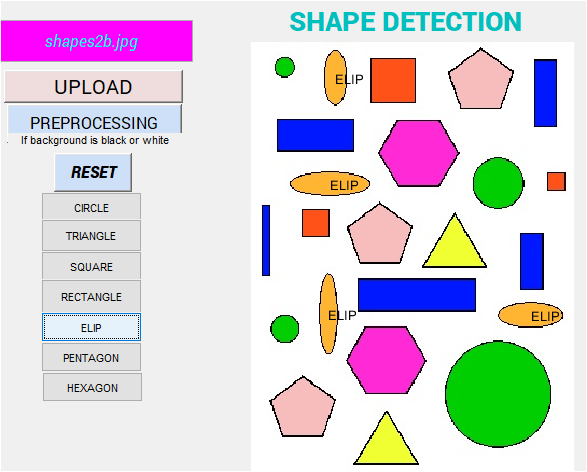


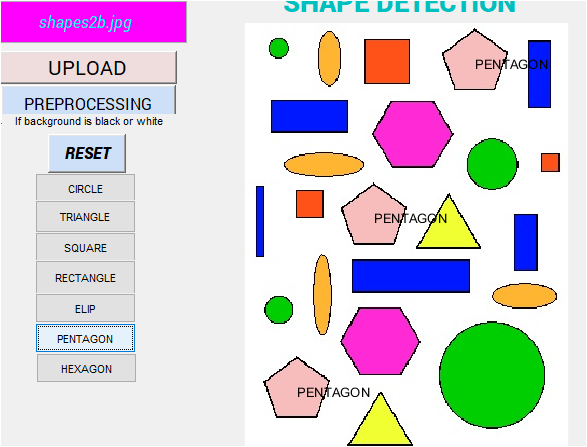


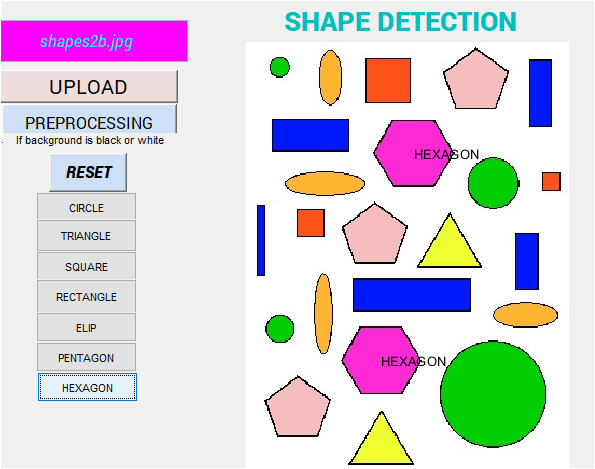












Code: *If you want to test my code, please go PREPROCESSING first before detect. I also send matlab file and .fig file on BB.*

function varargout = GUItest(varargin)

% GUITEST MATLAB code for GUItest.fig

% GUITEST, by itself, creates a new GUITEST or raises the existing

% singleton\*.

%

% H = GUITEST returns the handle to a new GUITEST or the handle to

% the existing singleton\*.

%

% GUITEST('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in GUITEST.M with the given input arguments.

%

% GUITEST('Property','Value',...) creates a new GUITEST or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before GUItest\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to GUItest\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUItest

% Last Modified by GUIDE v2.5 28-May-2021 16:55:05

global txt

global rbg

global out

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @GUItest\_OpeningFcn, ...

'gui\_OutputFcn', @GUItest\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before GUItest is made visible.

function GUItest\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to GUItest (see VARARGIN)

% Choose default command line output for GUItest

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes GUItest wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = GUItest\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

function edit1\_Callback(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text

% str2double(get(hObject,'String')) returns contents of edit1 as a double

global txt

txt = get(hObject, 'String')

% --- Executes during object creation, after setting all properties.

function edit1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global txt

global rgb

name = char(txt);

rgb=imread(name);

imshow(rgb)

% --- Executes on button press in CIRCLE.

function CIRCLE\_Callback(hObject, eventdata, handles)

% hObject handle to CIRCLE (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global out

%edge detection with Canny

bw=out;

%dien tich do matlab

%lay centroid

s = regionprops(bw, 'centroid');

%lay area

dt = regionprops(bw, 'area');

%lay perimeter

cv = regionprops(bw, 'perimeter');

dim = size(s);

%lay boundaries X,Y

boundaries = bwboundaries(bw);

for k=1:dim(1)

b= boundaries{k};

dim = size(b);

%Calculate the khoang cach tu centroid to each point at the boundaries

for i=1:dim(1)

khoangcach{k}(1,i) = sqrt ((b(i,2) - s(k).Centroid(1))^2 + (b(i,1) - s(k).Centroid(2))^2 );

end

%detemine the max and min khoang cach

a=max(khoangcach{k});

b=min(khoangcach{k});

%get the area from the prop command

%this is the area based on the number of pixels in the shape

c=dt(k).Area;

d=cv(k).Perimeter;

%dien tich

%pi\*R^2

tron = round(pi\*a^2);

if ((abs(c-tron)/c)<0.1)

text(s(k).Centroid(1),s(k).Centroid(2),'CIRCLE')

end

end

% --- Executes on button press in TRIANGLE.

function TRIANGLE\_Callback(hObject, eventdata, handles)

% hObject handle to CIRCLE (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global out;

%edge detection with Canny

%get rid of noise of less than 15 pixel; %loc noise trong khoang cach 30 pixel

%fill any hole to complete a shape

bw = out;

%dien tich do matlab

%lay centroid

s = regionprops(bw, 'centroid');

%lay area

dt = regionprops(bw, 'area');

%lay perimeter

cv = regionprops(bw, 'perimeter');

dim = size(s);

%lay boundaries X,Y

boundaries = bwboundaries(bw);

for k=1:dim(1)

b= boundaries{k};

dim = size(b);

%Calculate the khoang cach tu centroid to each point at the boundaries

for i=1:dim(1)

khoangcach{k}(1,i) = sqrt ((b(i,2) - s(k).Centroid(1))^2 + (b(i,1) - s(k).Centroid(2))^2 );

end

%detemine the max and min khoang cach

a=max(khoangcach{k});

b=min(khoangcach{k});

%get the area from the prop command

%this is the area based on the number of pixels in the shape

c=dt(k).Area;

d=cv(k).Perimeter;

%dien tich

canh=round(d/3);

S= round(canh\*(a+b)/2);

%other condition

pytago = sqrt(a\*a-b\*b)

canh2 = canh/2;

if ((abs(S-c)/c <0.1) && (abs(pytago-canh2)/canh2 < 0.1))

text(s(k).Centroid(1),s(k).Centroid(2),'TRIANGLE')

end

end

% hObject handle to TRIANGLE (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in SQUARE.

function SQUARE\_Callback(hObject, eventdata, handles)

% hObject handle to CIRCLE (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global out

bw=out

%dien tich do matlab

%lay centroid

s = regionprops(bw, 'centroid');

%lay area

dt = regionprops(bw, 'area');

%lay perimeter

cv = regionprops(bw, 'perimeter');

dim = size(s);

%lay boundaries X,Y

boundaries = bwboundaries(bw);

for k=1:dim(1)

b= boundaries{k};

dim = size(b);

%Calculate the khoang cach tu centroid to each point at the boundaries

for i=1:dim(1)

khoangcach{k}(1,i) = sqrt ((b(i,2) - s(k).Centroid(1))^2 + (b(i,1) - s(k).Centroid(2))^2 );

end

%detemine the max and min khoang cach

a=max(khoangcach{k});

b=min(khoangcach{k});

%get the area from the prop command

%this is the area based on the number of pixels in the shape

c=dt(k).Area;

d=cv(k).Perimeter;

%dien tich

%pi\*R^2

canh=round(d/4);

square=round(canh\*canh);

othercanh= 2\*sqrt(a\*a-b\*b);

if ((abs(square-c)/c < 0.15) && (abs(othercanh-canh)/othercanh <0.1))

text(s(k).Centroid(1),s(k).Centroid(2),'SQUARE')

end

end

% hObject handle to SQUARE (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in RECTANGLE.

function RECTANGLE\_Callback(hObject, eventdata, handles)

% hObject handle to RECTANGLE (see GCBO)

global out

bw=out

%dien tich do matlab

%lay centroid

s = regionprops(bw, 'centroid');

%lay area

dt = regionprops(bw, 'area');

%lay perimeter

cv = regionprops(bw, 'perimeter');

dim = size(s);

%lay boundaries X,Y

boundaries = bwboundaries(bw);

for k=1:dim(1)

b= boundaries{k};

dim = size(b);

%Calculate the khoang cach tu centroid to each point at the boundaries

for i=1:dim(1)

khoangcach{k}(1,i) = sqrt ((b(i,2) - s(k).Centroid(1))^2 + (b(i,1) - s(k).Centroid(2))^2 );

end

%detemine the max and min khoang cach

a=max(khoangcach{k});

b=min(khoangcach{k});

%get the area from the prop command

%this is the area based on the number of pixels in the shape

c=dt(k).Area;

d=cv(k).Perimeter;

%dien tich

%pi\*R^2

canhdai=round(2\*sqrt(a\*a-b\*b))

canhngan=round((d-2\*canhdai)/2);

canhngan2=round(2\*b);

canhdai2=round((d-2\*canhngan2)/2);

dientich=round(canhdai\*canhngan);

dientich2=round(canhdai\*canhngan2);

if ((abs(dientich-c)/c < 0.39) && (canhdai/canhngan>1) && (abs(dientich2-c)/c < 0.2) && (canhdai/canhngan2>1) && (canhdai2/canhngan2>1) && (canhdai2/canhngan2>1) && (abs(canhdai2-canhdai)/canhdai2<0.1)&&(abs(canhngan2-canhngan)/canhngan2<0.3))

text(s(k).Centroid(1),s(k).Centroid(2),'RECTANGLE')

end

end

% --- Executes on button press in PREPROCESSING.

function PREPROCESSING\_Callback(hObject, eventdata, handles)

global txt

global rgb

global out

name = char(txt);

rgb=imread(name);

a=rgb2gray(rgb);

[r c] = size(a);

for i=0:255

count(i+1,1)=i;

count(i+1,2)=0;

end

for i=1:r

for j=1:c

for k=1:256

if (a(i,j) == count(k,1))

count(k,2) = count(k,2)+1;

end

end

end

end

maxindex = max(count(:,2))

for i=1:256

if (count(i,2) == maxindex)

fmaxpixel = count(i,1);

break

end

end

threshold = 20;

for i=1:r

for j=1:c

if (fmaxpixel >= 150)

if (a(i,j) >= fmaxpixel-threshold)

a(i,j) = fmaxpixel;

end

else

if (a(i,j) <= fmaxpixel+threshold)

a(i,j) = fmaxpixel;

end

end

end

end

for i=1:r

for j=1:c

if (a(i,j) == fmaxpixel)

out(i,j)=0;

else

out(i,j)=255;

end

end

end

out = bwareaopen(out,20);

out = imfill(out,'holes');

% --- Executes on button press in ELIP.

function ELIP\_Callback(hObject, eventdata, handles)

% hObject handle to ELIP (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% hObject handle to RECTANGLE (see GCBO)

global out

bw=out

%dien tich do matlab

%lay centroid

s = regionprops(bw, 'centroid');

%lay area

dt = regionprops(bw, 'area');

%lay perimeter

cv = regionprops(bw, 'perimeter');

dim = size(s);

%lay boundaries X,Y

boundaries = bwboundaries(bw);

for k=1:dim(1)

b= boundaries{k};

dim = size(b);

%Calculate the khoang cach tu centroid to each point at the boundaries

for i=1:dim(1)

khoangcach{k}(1,i) = sqrt ((b(i,2) - s(k).Centroid(1))^2 + (b(i,1) - s(k).Centroid(2))^2 );

end

%detemine the max and min khoang cach

a=max(khoangcach{k});

b=min(khoangcach{k});

%get the area from the prop command

%this is the area based on the number of pixels in the shape

c=dt(k).Area;

d=cv(k).Perimeter;

%dien tich

%pi\*R^2

canhdai=a;

canhngan=b;

landa = (a-b)/(a+b);

dientich=a\*b\*pi;

const=1+(3\*(landa^2))/(10+sqrt(4-3\*(landa^2)));

chuvi= pi\*(a+b)\*const;

tron = round(pi\*a^2);

if ((abs(dientich-c)/c <0.1) && (abs(chuvi-d)/d<0.013))

if (abs(tron-c)/c<0.1)

else

text(s(k).Centroid(1),s(k).Centroid(2),'ELIP')

end

end

end

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in RESET.

function RESET\_Callback(hObject, eventdata, handles)

% hObject handle to RESET (see GCBO)

global txt

global rgb

name=char(txt)

rgb=imread(name)

imshow(rgb);

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in PENTAGON.

function PENTAGON\_Callback(hObject, eventdata, handles)

global out

bw=out

%dien tich do matlab

%lay centroid

s = regionprops(bw, 'centroid');

%lay area

dt = regionprops(bw, 'area');

%lay perimeter

cv = regionprops(bw, 'perimeter');

dim = size(s);

%lay boundaries X,Y

boundaries = bwboundaries(bw);

for k=1:dim(1)

b= boundaries{k};

dim = size(b);

%Calculate the khoang cach tu centroid to each point at the boundaries

for i=1:dim(1)

khoangcach{k}(1,i) = sqrt ((b(i,2) - s(k).Centroid(1))^2 + (b(i,1) - s(k).Centroid(2))^2 );

end

%detemine the max and min khoang cach

a=max(khoangcach{k});

b=min(khoangcach{k});

%get the area from the prop command

%this is the area based on the number of pixels in the shape

c=dt(k).Area;

d=cv(k).Perimeter;

canh=round(d/5);

canh2=round(2\*sqrt(a^2-b^2));

chuvi=canh\*5;

chuvi2=canh2\*5;

dientich=5\*canh\*b/2;

dientich2=5\*canh2\*b/2;

if (abs(dientich-c)/c<0.1 && abs(chuvi2-d)/d<0.1 && abs(chuvi-d)/d<0.05 && abs(dientich2-c)/c<0.05)

text(s(k).Centroid(1),s(k).Centroid(2),'PENTAGON')

end

end

% --- Executes on button press in HEXAGON.

function HEXAGON\_Callback(hObject, eventdata, handles)

global out

bw=out

%dien tich do matlab

%lay centroid

s = regionprops(bw, 'centroid');

%lay area

dt = regionprops(bw, 'area');

%lay perimeter

cv = regionprops(bw, 'perimeter');

dim = size(s);

%lay boundaries X,Y

boundaries = bwboundaries(bw);

for k=1:dim(1)

b= boundaries{k};

dim = size(b);

%Calculate the khoang cach tu centroid to each point at the boundaries

for i=1:dim(1)

khoangcach{k}(1,i) = sqrt ((b(i,2) - s(k).Centroid(1))^2 + (b(i,1) - s(k).Centroid(2))^2 );

end

%detemine the max and min khoang cach

a=max(khoangcach{k});

b=min(khoangcach{k});

%get the area from the prop command

%this is the area based on the number of pixels in the shape

c=dt(k).Area;

d=cv(k).Perimeter;

canh=round(d/6);

canh2=round(2\*sqrt(a^2-b^2));

chuvi=canh\*6;

chuvi2=canh2\*6;

dientich=6\*canh\*b/2;

dientich2=6\*canh2\*b/2;

if (abs(dientich-c)/c<0.1 && abs(dientich2-c)/c<0.2 && abs(chuvi-d)/d<0.1)

text(s(k).Centroid(1),s(k).Centroid(2),'HEXAGON')

end

end