Aluno: Eduardo Mafra Pereira
Email: emafra0@gmail.com

Trabalho 5

Visão computacional

Neste relatório foi preciso traçar realizar a detecção de 3 classes de objetos fornecidos (estrelas, círculos e quadrados), a partir disto foi necessário estabelecer linhas que conectar os objetos de mesma classe. Em uma segunda sequência ao desafio proposto foi necessário desviar possíveis interferências que o objeto estrala pudesse causar na linha caso bloqueasse seu caminho. Sendo assim foi desenvolvido um algoritmo que detecta os objetos, relaciona os objetos de mesma classe através de linhas e desvia as linhas que são bloqueadas pela estrela. O algoritmo portanto é apresentado da seguinte maneira:

```
video = videoinput('winvideo',2);
flag=0;
for i=1:40
    foto = getsnapshot(video); %Captura a imagem do vídeo.
    imshow(foto); % Mostra a imagem capturada.
    im_gray = rgb2gray(foto);
    %imshow(im_gray);
    im_bw = im2bw(im_gray);
    %imshow(im_bw);
    im_bin = not(im_bw);
    %imshow(im_bin);
    im_label = bwlabel(im_bin,8);
    %imtool(im_label);
    N = max(max(im_label)); % Numero de objetos detectados.
    prop = regionprops(im_label,'all'); % Função para obter as propriedades
    % de cada objeto.
     hold on
    % Lista de objetos:
    quadrado=[];
    circulos=[];
    strars=[];
    % Índices das Listas:
    q=0;
    q1=0;
    q2=0;
```

```
    % Laço para percorrer todos os objetos:
for i = 1:N
    [a,b] = size(prop(i).PixelList) % Captura através da variável "a" o
    % Numero de pixels de cada objeto
    if a>300 && a<3000 % Condição para eliminar ruídos.
        % Condição para detectar quadrados:
        if prop(i).BoundingBox(:,3)*prop(i).BoundingBox(:,4) <= prop(i).Area*1.2 && ...
            prop(i).BoundingBox(:,3)*prop(i).BoundingBox(:,4) >= prop(i).Area*0.8
        % Mostra o Objeto detectado na imagem:
        plot(prop(i).Centroid(:,1),prop(i).Centroid(:,2),'b*');
        q=q+1; % Relaciona o objeto encontrado a um índice.
        quadrado(q)=i % Acrescenta o objeto em uma lista.
    % Condição para detectar Círculos:
        elseif prop(i).Perimeter <= (pi*(prop(i).EquivDiameter))*1.03 && prop(i).Area >= ...
            pi*(prop(i).EquivDiameter)*0.97 &&  prop(i).Area < ...
            prop(i).BoundingBox(:,3)*prop(i).BoundingBox(:,4)
            q1=q1+1; % Relaciona o objeto encontrado a um índice.
            % Mostra o objeto na imagem
            plot(prop(i).Centroid(:,1),prop(i).Centroid(:,2),'g*');
            circulos(q1)=i; % Acrescenta o objeto em uma lista.
```

```
        elseif    ((prop(i).EquivDiameter/2)^2)*pi ~ prop(i).Area
            q2=q2+1; % Relaciona o objeto encontrado a um índice.
            plot(prop(i).Centroid(:,1),prop(i).Centroid(:,2),'r*');
            stars(q2)=i; % Acrescenta o objeto em uma lista.
        end
    end
end

%  Define as variáves "q" e "ql" como tamanho da lista de objetos para as
% classes quadrado e círculos:
[c,q] = size (quadrado);
[c,ql] = size (circulos);
```

Obs: Para o encontrar os objetos foi necessário criar relações para identificá-los e diferenciá-los. Sendo assim, os objetos só serão quadrados se sua área for igual a da Bounding Box gerada neste objeto, só serão círculos se sua área for inferior a área gerada por sua Bounding Box e seu perímetro for igual ao seu diâmetro multiplicado por pi. Se Caso o objeto não for círculo nem quadrado ele será considerado estrela (vale ressaltar que os objetos com um quantidade inferior a 300 pixels e superior a 3000 serão ignorados).

Continuação...:

```
if stars(1)>0 % Se houver a destecção de estrela (precaução para possíveis
% erros).
%quadrados:
for k = 1:q-1

        %  Calculo do ponto central dos objetos interligados:
        xcentral=(prop(quadrado(k)).Centroid(:,1)+prop(quadrado(k+1)).Centroid(:,1))/2
        ycentral=(prop(quadrado(k)).Centroid(:,2)+prop(quadrado(k+1)).Centroid(:,2))/2

        %  Distância entre o ponto central dos pontos inteligados e o
        % centroid da estrela:
        distx = xcentral-prop(stars(1)).Centroid(:,1)
        disty = ycentral-prop(stars(1)).Centroid(:,2)


        % Ponto central de Bezier:

        % Se o objeto estiver na vertical:
        if prop(quadrado(k)).Centroid(:,1) <= ...
           prop(quadrado(k+1)).Centroid(:,1)*1.2 && ...
           prop(quadrado(k)).Centroid(:,1) >= ...
           prop(quadrado(k+1)).Centroid(:,1)*0.8
        x2 = xcentral + (500/(distx)) % Pontos centrais de bezier
        y2 = ycentral + (1/(disty))
        % Se estiver na horizontal ou diagonal:
        else prop(quadrado(k)).Centroid(:,2) <= ...
            prop(quadrado(k+1)).Centroid(:,2)*1.2 && ...
            prop(quadrado(k)).Centroid(:,2) >= ...
            prop(quadrado(k+1)).Centroid(:,2)*0.8
        x2 = xcentral + (1/(distx)) % Pontos centrais de bezier
        y2 = ycentral + (500/(disty))
        end

        % Chama a função de Bezier
        bezier(prop(quadrado(k)).Centroid(:,1),...
        prop(quadrado(k)).Centroid(:,2),x2,y2,...
        prop(quadrado(k+1)).Centroid(:,1),prop(quadrado(k+1)).Centroid(:,2));
end
```

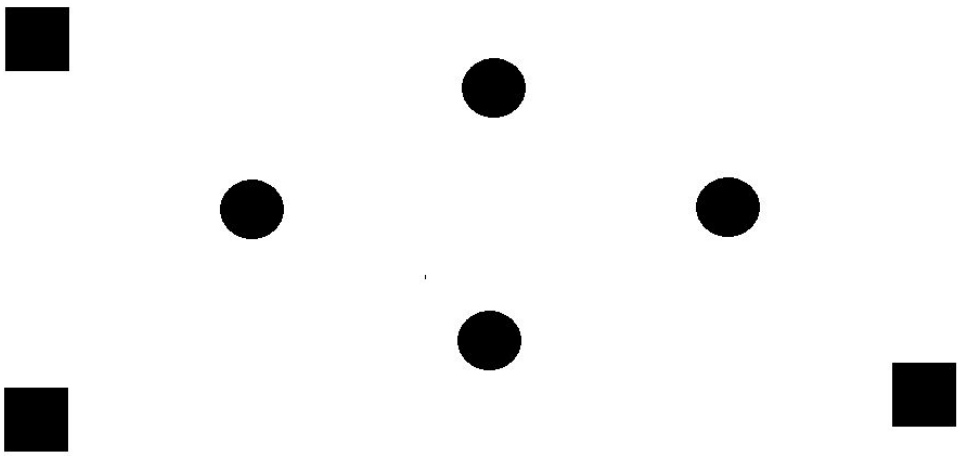
```
% Circulos(Mesmo processo aplicado para a classe dos quadrados):
for k = 1:ql-1
        xcentral=(prop(circulos(k)).Centroid(:,1)+prop(circulos(k+1)).Centroid(:,1))/2
        ycentral=(prop(circulos(k)).Centroid(:,2)+prop(circulos(k+1)).Centroid(:,2))/2

        distx = xcentral-prop(stars(1)).Centroid(:,1)
        disty = ycentral-prop(stars(1)).Centroid(:,2)

        x2 = xcentral + (1/(distx)) %arrumar esse x2 e y2
        y2 = ycentral + (500/(disty))

        bezier(prop(circulos(k)).Centroid(:,1),...
        prop(circulos(k)).Centroid(:,2),x2,y2,...
        prop(circulos(k+1)).Centroid(:,1),prop(circulos(k+1)).Centroid(:,2));
end
end
end
```

Obs 2: Para poder realizar as curvas entre os objetos caso a estrela esteja bloqueando o caminho foi preciso utilizar o conceito de linhas de Bézier, estas linhas são constituídas por 3 pontos e o ponto central dependendo de sua posição pode gerar uma linha curva. Para o algoritmo apresentado foi criado um sistema dinâmico que aplica Curvas de Bézier em todas as linhas entre os objetos, porém seus pontos centrais dependem da sua aproximação com o ponto central da estrela. Portanto quanto mais próximo o ponto central estiver do centróide da estrela, maior influência será gerada neste ponto central. Sendo assim, os objetos que tiverem sua distância central mais afastada do centróide da estrela irão ter no ponto central da função de Bézier um valor muito próximo da posição central entre estes dois objetos, caso a estrela esteja perto deste ponto central o valor aplicado será alterado com maior intensidade proporcionando portanto curvas entres os objetos.

Função de Bézier:

```
function a=bezier(x1,y1,x2,y2,x3,y3)
plot(x1,y1,'s')
hold on
plot(x2,y2,'s')
hold on
plot(x3,y3,'s')
hold on
i=0;
for t=0:0.05:1
    i=i+1;
% Vetores de posição (e aplicação da fórmula de Bézier):
    x(i)=(((t)^2)*x1)+(2*t*(1-t)*x2)+((1-t)^2*x3);
    y(i)=(((t)^2)*y1)+(2*t*(1-t)*y2)+((1-t)^2*y3);
end
% Plot da linhas de Bézier:
plot (x,y,'Color','black')
```

Imagem utilizada para as simulações:



Recomendação para um melhor resultado: desenhar a estrela maior que os objetos apresentados na figura anterior.

Resultados: