Laboratório 3 - Visão Computacional.
Aluno: Eduardo Mafra Pereira.
email: emafra0@gmail.com.
Questão 1.
Neste problema foi preciso realizar a equalização de imagem, processo
muito útil para suavizar imagens muito "densas". Para solucionar o
problema foi ultilizado o comando "histeq", este comando atua
diretamente no histograma ("espectro") da imagem disperçando suas
componentes. Ou seja, equalizando ela.

```
clear all
clc
Im1 = imread('HistEq.jpg'); % Chamada da imagem.
Im1 = rgb2gray(Im1); % Transforma a imagem para a escala Gray.
figure(1),imhist(Im1); % Plota o histograma da figura Im1.
title('Histograma da Imagem 1')
J = histeq(Im1); % Atribui a imagem equalizada a variável J.
figure(2),imshowpair(Im1,J,'montage');
title('Imagem 1 "Sem" e "Com" Aplicação do Histograma');
figure(3),imhist(J); % Histograma da imagem equalizada.
title('Histograma da Imagem 1 Equalizada');
%
%  Processo se repete para as outras duas imagens.
%  Imagem 2:
Im2 = imread('HistEq2.jpg'); % Chamada da imagem.
Im2 = rgb2gray(Im2); % Transforma a imagem para a escala Gray.
figure(4),imhist(Im2); % Plota o histograma da figura Im2.
title('Histograma da Imagem 2')
J = histeq(Im2); % Atribui a imagem equalizada a variável J.
figure(5),imshowpair(Im2,J,'montage');
title('Imagem 2 "Sem" e "Com" Aplicação do Histograma');
figure(6),imhist(J); % Histograma da imagem equalizada.
title('Histograma da Imagem 2 Equalizada');
%  Imagem 3:
Im3 = imread('HistEq3.jpg'); % Chamada da imagem.
Im3 = rgb2gray(Im3); % Transforma a imagem para a escala Gray.
figure(7),imhist(Im3); % Plota o histograma da figura Im3.
title('Histograma da Imagem 3')
J = histeq(Im3); % Atribui a imagem equalizada a variável J.
figure(8),imshowpair(Im3,J,'montage');
title('Imagem 3 "Sem" e "Com" Aplicação do Histograma');
figure(9),imhist(J); % Histograma da imagem equalizada.
title('Histograma da Imagem 3 Equalizada');
```

```
Warning: Image is too big to fit on screen; displaying at 67%
```

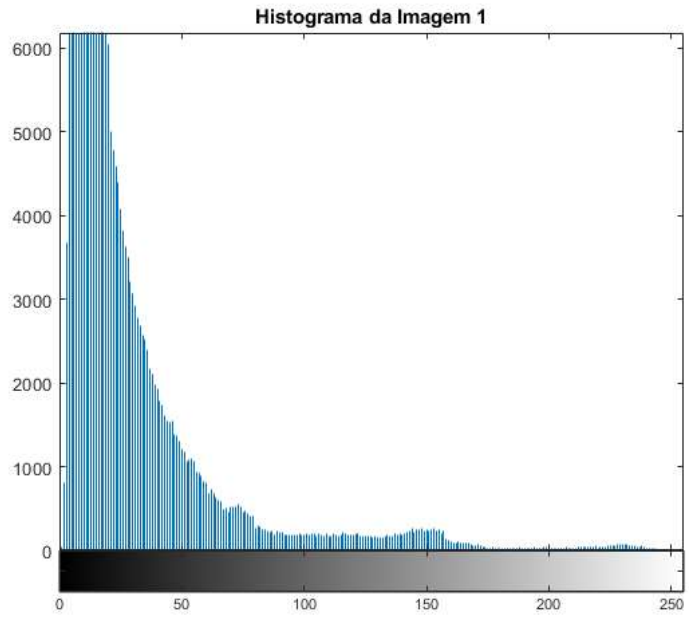
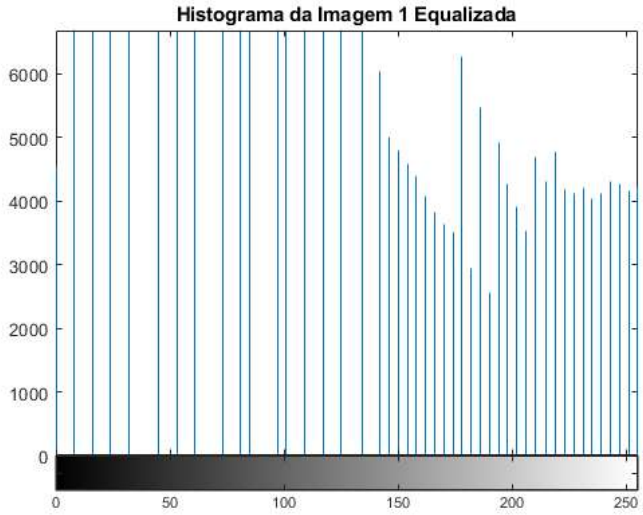


Imagem 1 "Sem" e "Com" Aplicação do Histograma



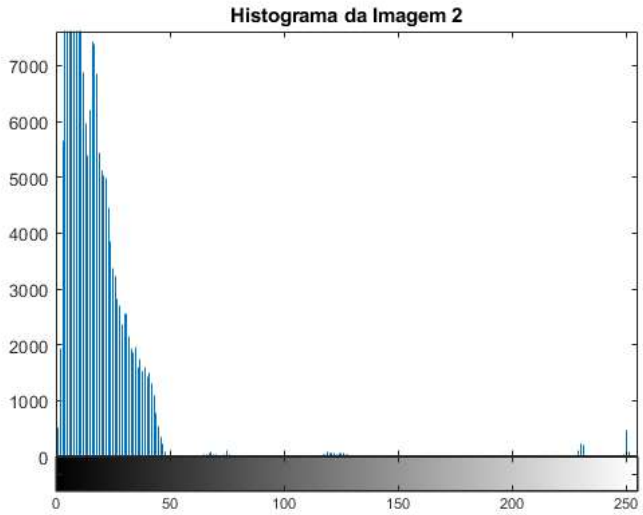
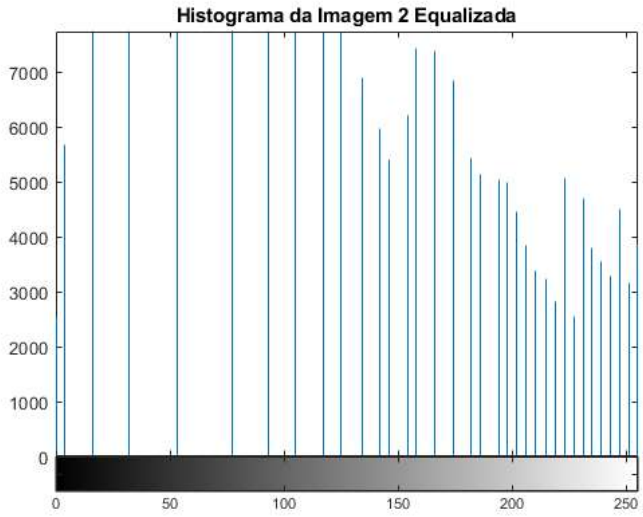
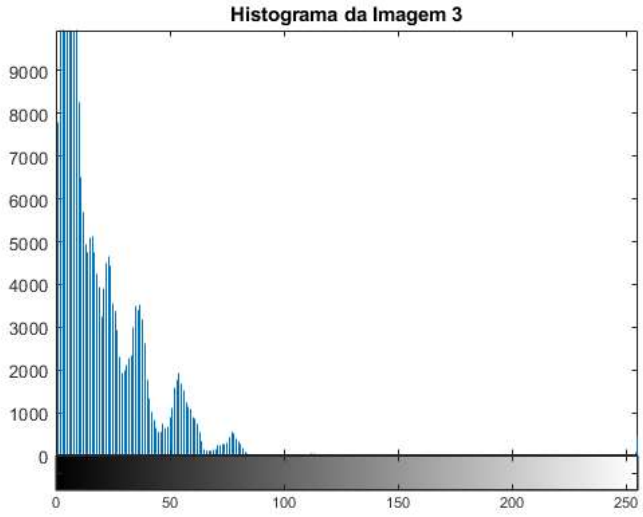
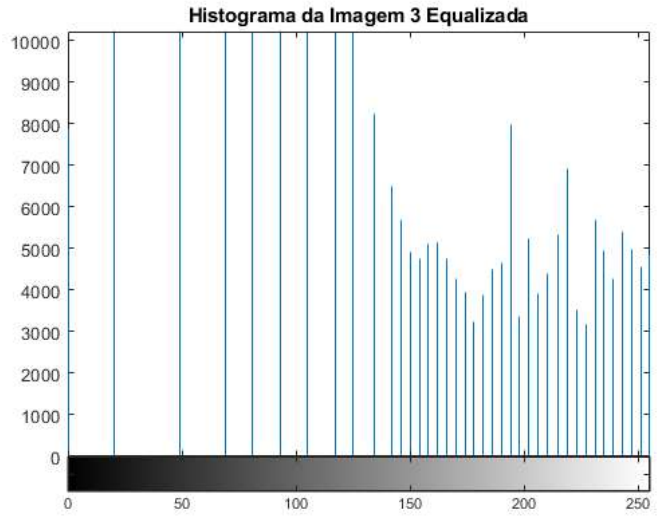


Imagem 2 "Sem" e "Com" Aplicação do Histograma







Questão 2.

O objetivo desta etapa era desenvolver um algoritmo que conseguisse contar dinheiro em uma imagem. Para a solução do problema foi proposto o algoritmo a seguir. Este transforma a imagem original em uma imagem preto e branco e homogeneizam a imagem gerada para deixa-lá mais uniforme.

A imagem gerada é passada por um laço removedor de ruídos, e na sequência transformada para uma imagem indexada. O objetivo deste processo é que imagens indexadas proporcionam uma diferenciação de objetos.

Com a imagem já indexada são atribuídos as quantidades de pixels de cada objeto a uma lista "tam[]". Com as quantidades já atribuidas a lista, esta é aplicada em um laço de repetição que compara os valores de cada elemento da lista afim de contar a quantidade de elementos de mesmo tamanho que a lista contém. E finalmente são atribuidos os pesos das moedas a cada quantidade de objetos equivalentes, proporcionando assim o cálculo do dinheiro total presente na imagem.

```
clear all
clc
Im1 = imread('Moedas.jpg'); % Chamada da imagem.
R= double(Im1(:,:,1));
G= double(Im1(:,:,2));
B= double(Im1(:,:,3));
Im1 = rgb2gray(Im1); % Transformação a imagem de RGB para GRAY.
Im1 = Im1-110; % Equaliza a imagem.
ImHBW = im2bw(Im1); % Transforma da escala Gray para preto e Branco.
imshow(ImHBW); % Mostra a imagem em escala preto e branco homogeneizada
title("Imagem em escala preto e branco homogeneizada")
```

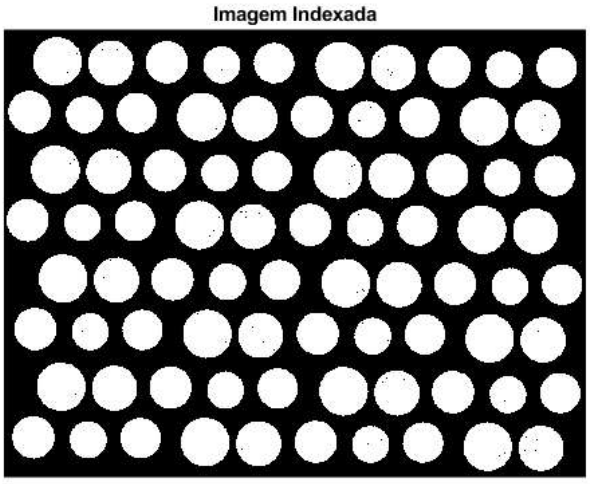
```
Im_Bin = not(ImHBW); % Inversão das cores da imagem.
flag=0;
for i=2:1:length(Im_Bin(:,1))-1 % Laço para eliminar ruído
    for j=2:1:length(Im_Bin(1,:))-1
        flag=0;
        for k=-1:1
            for c=-1:1
                if Im_Bin(i+k,j+c)==1
                   flag=flag+1;
                end
            end
        end
        if flag < 3 % Se o pixel não tiver ao menos 3 pixles brancos ao seu
                    %redor, este pixel é "eliminado".
           Im_Bin(i,j)=0; % "Elimina" pixel.
        end
    end
end
%   Transforma a imagem preto e branco em uma imagem double indexada.
%   A variável num apresenta a quantidades de índices atribuidos a imagem
% ou seja, a quantidade de "objetos" presentes na imagem.
[Im_Label,num] = bwlabel(Im_Bin,8);
Im_Lab=Im_Label;
imshow(Im_Lab);
title('Imagem Indexada');
%N = max(max(Im_Lab));
tam=[]; % Lista para atribuir a quantidade de pixels para cada índice
%(tamanho do objeto).
for i=1:1:num
    [rows,cols,vals] = find(Im_Lab==i);
    [t,c]=size(rows); % atribui a quantidade de índices a variável t.
    tam(i)= t;
end
tam_n=tam;
valores=[]; % Lista para conter a quantidade de "objetos" com mesmo
% "tamanho".
list=[];
for i = 1:1:num
    count=0;
    a = tam_n(i);
    for q = 1:1:num
        %   Se os objetos possuem uma quantidade de pixels similar ao
        %  índice proposto (margem de 5%), a quantidade de objetos é
        %  calculada.
        if tam_n(q) >= (a*0.95) & tam(q) <= (a*1.05)
            count = count +1; % Contador de número de objetos.
            tam_n(q)=0; % Zera os índices que já foram contados.
        end
        if q == num
            valores(i) = count; % Atribui a quantidade de objetos similares
                                %  lista valores[].
        end
    end

end
elementos=[]; % Lista que relaciona os índices aos objetos encontrados
p=1;
for i = 1:1:num
    b=valores(i);
    if b > 0 % Se o índice não foi zerado
       elementos(p)=i; % Atribui-se a posição deste índice na lista
                       %  "valores" para a lista "elementos".
       p=p+1; % Tamanho da lista elementos -1.
    end
end
for i = 1:1:p-1
    a = elementos(i);
    flag=0;
    for q= 1:1:p-1 % Compara quais objetos tem mais pixels ("maior
                   %  tamanho").
        b = elementos(q);
        if tam(a)>tam(b)
            flag=flag+1;
        end
    end
    if flag == 4 % Condições para atribuir os pesos das moedas conforme o
                 %  "tamanho".
        RS1= tam(a);
        QRS1= valores(a);
    end
    if flag == 3
        RS0_50= tam(a);
        QRS0_50= valores(a);
    end
    if flag == 2
        RS0_25= tam(a);
        QRS0_25= valores(a);
    end
    if flag == 1
        RS0_10= tam(a);
        QRS0_10= valores(a);
    end
    if flag == 0
        RS0_05= tam(a);
        QRS0_05= valores(a);
    end
end
```

```
% Valor total calculado.
Total=QRS1+(0.5*QRS0_50)+(0.25*QRS0_25)+(0.1*QRS0_10)+(0.05*QRS0_05)
```

```
Total =

   30.4000
```



Questão 3.
Na etapa 3 foi proposto apagar uma tatuagem utilizando máscaras de suavização. Sendo assim, foi criada uma máscara de suavização e esta será aplicada apenas quando os valores presentes nas matrizes RGB forem maiores que 180. Ou seja, filtro aplicado apenas nos pixels mais escuros.

```
clear all
clc
Im1 = imread('Tatoo1.jpg');
%Img = rgb2gray(Im1);

figure(1),imshow(Im1); % captura dois pontos na imagem para processar a
% reduzida. Só a parte da tatuagem.
title('Imagem original');
[x y] = ginput(2);

%Redução da imagem
Im(:,:,1)=Im1(y(1):y(2),x(1):x(2),1);
Im(:,:,2)=Im1(y(1):y(2),x(1):x(2),2);
Im(:,:,3)=Im1(y(1):y(2),x(1):x(2),3);
%figure(2),imshow(Im);
[n m c] = size(Im); % Escolha de dois pontos que irão ditar o tamanho da
% nova imagem.
figure(2),imshow(Im);
title('Imagem recortada');

Mask=ones(3,3); %criação da máscara de suavização.
div=9;
z=2;

Imagem=double(Im);
Ir(:,:,1)=Imagem(:,:,1);
Ig(:,:,2)=Imagem(:,:,2);
Ib(:,:,3)=Imagem(:,:,3);

 % laço para repetir o procedimento 3 vezes. Um laço para cada
    % profunfidade da matriz (RGB).

for f=1:1:40 % repete a aplicação do filtro 40 vezes para cada matriz RGB
    % da imagem processada.
    for i=z:1:n-z
       for j=z:1:m-z
            val=0;
          % Condição if para aplicar o filtro apenas nos pixels mais
          % escuros, ou seja, na tatuagem.
          if Im(i,j,1)<180 || Im(i,j,2)<180 || Im(i,j,3)<180

            % Realiza o somatório das convoluções realizadas em cada
            % profundidade RGB.
            Ir(i,j)= sum(sum(double(Imagem(i-1:i+1,j-1:j+1,1).*Mask(:,:))));
            Ig(i,j)= sum(sum(double(Imagem(i-1:i+1,j-1:j+1,2).*Mask(:,:))));
            Ib(i,j)= sum(sum(double(Imagem(i-1:i+1,j-1:j+1,3).*Mask(:,:))));


            Imagem(i,j,1) = uint8(Ir(i,j)/div); %Atribui a média dos pixels
            % a aos a imagem gerada.
```

```
            Imagem(i,j,2) = uint8(Ig(i,j)/div);


            Imagem(i,j,3) = uint8(Ib(i,j)/div);

          end

    end
end

end
result(:,:,1)=Imagem(:,:,1);
result(:,:,2)=Imagem(:,:,2);
result(:,:,3)=Imagem(:,:,3);
[z,h]=size(result);
Im2=Im1;
% Faz a junção da imagem recortada com a imagem original.
Im2(y(1)+7:y(2)-7,x(1)+7:x(2)-7,1)=result(8:z-7,8:(h/3)-7,1);
Im2(y(1)+7:y(2)-7,x(1)+7:x(2)-7,2)=result(8:z-7,8:(h/3)-7,2);
Im2(y(1)+7:y(2)-7,x(1)+7:x(2)-7,3)=result(8:z-7,8:(h/3)-7,3);

figure(3),imshow(Im2);
title('Resultado Final');
```

```
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
```

Resultado Final

Questão 4
Para a etapa 4 foi solicitado realizar um realce de imagem através de aplicação de máscaras de realce. Sendo assim, foram propostas 3 máscaras para a solução do problema. Estas serão convolucionadas com a matriz original, e por fim, são somadas todas as multiplicações destes pixels e atribuidas aos elementos centrais do processo.

```
clear all
clc
Im1 = imread('Burrinhos.jpg');
figure(1),imshow(Im1);
title('Imagem Original');
Mask = 0.1*[1 -2 1; -2 4 -2; 1 -2 1]; % máscara teste 1
beta= 1;
%Mask = [0 -1 0; -1 4 -1; 0 -1 0]; % máscara teste 2
%Mask = 0.1*[1 -1 1; -1 4 -1; 1 -1 1]; % máscara teste 3

result=Im1;
result1=result;
for i =2:length(result(:,1))-1
    for j =2:length(result(1,:))-1
        for e=-1:1
            for r=-1:1
                M(e+2,r+2) = result(e+i,r+j); % pontos da matriz original
                % atribuidos a uma matriz 3x3.
            end
        end

        result1(i,j) = result1(i,j)+ beta*(sum(sum(double(M).*Mask)));
        % Somatório dos pontos das matrizes convolucionadas atribuido ao
        % elemento central (i,j) de uma secção da matriz "result1".
    end
end
   figure(2),imshow(result1);
   title('Imagem Realçada');
```

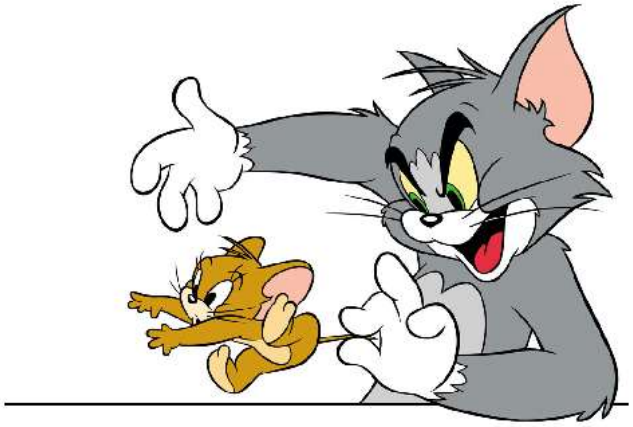
```
Warning: Image is too big to fit on screen; displaying at 67%
```

Questão 5.
Para a quinta etapa foi implementado um algoritmo de detecção de bordas.
neste foram feitas alguns teste. O primeiro como uma máscara de detecção
qualquer, um segundo teste aplicando uma máscara de Sobel vertical e
uma horizontal e por fim o último teste utilizou 2 máscaras de Prewitt (
uma vertical e outra horizontal).
A implementação do código é apresentada da seguinte maneira:

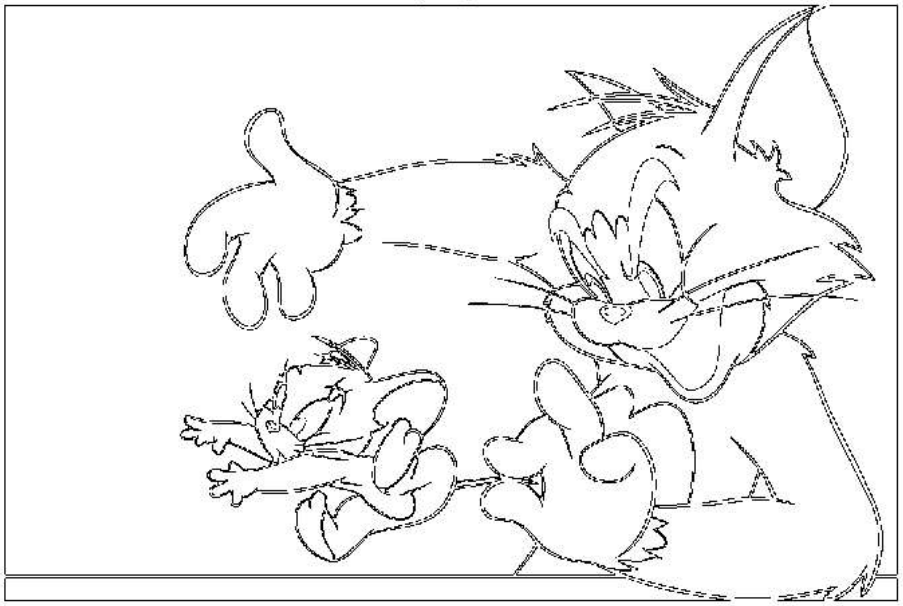
```
clear all
clc
Im1 = imread('tom_e_jerry.jpg');
Img = rgb2gray(Im1);
ImHBW = im2bw(Img); % Trnasformação da imagem para escala preto e branco.
%Im_Bin = not(ImHBW);
figure(1),imshow(Im1);
Mask = ones(3,3);
Mask(2,2) = -8; % Máscara 1.
prewittv = [1 0 -1; 1 0 -1; 1 0 -1];
prewitth = [-1 -1 -1; 0 0 0; 1 1 1]; % Máscaras de Prewitt.
sobelv = [-1 0 1; -2 0 2; -1 0 1];
sobelh = [-1 -2 -1; 0 0 0; 1 2 1]; % Máscaras de Sobel.
val=0;
result=ImHBW;
[n,m]=size(result);
result1=result;
result2=result;
result3=result;
result4=result;
result5=result;
```

```
for i =2:length(result(:,1))-1
    for j =2:length(result(1,:))-1
        for I=-1:1
            for J=-1:1
                M(I+2,J+2) = result(I+i,J+j);
            end
        end
        result1(i,j) = uint8(sum(sum(double(M).*Mask)));
        result2(i,j) = uint8(sum(sum(double(M).*sobelv)));
        result3(i,j) = uint8(sum(sum(double(M).*sobelh)));
        result4(i,j) = uint8(sum(sum(double(M).*prewittv)));
        result5(i,j) = uint8(sum(sum(double(M).*prewitth)));
    end
end
   resultf1=not(result1);%resultado com a máscara 1
   figure(2),imshow(resultf1);
   title('Resultado da Aplicação da Máscara 1')
   resultf2=not(result2+result3);%resultado soma de máscaras de
   % sobel na horizontal e vertical
   figure(3),imshow(resultf2);
   title('Resultado da Aplicação das Máscaras de Sobel')
   resultf3=not(result4+result5);%resultado soma de máscaras de
   % Prewitt na horizontal e vertical
   figure(4),imshow(resultf3);
   title('Resultado da Aplicação das Máscaras de Prewitt')
```

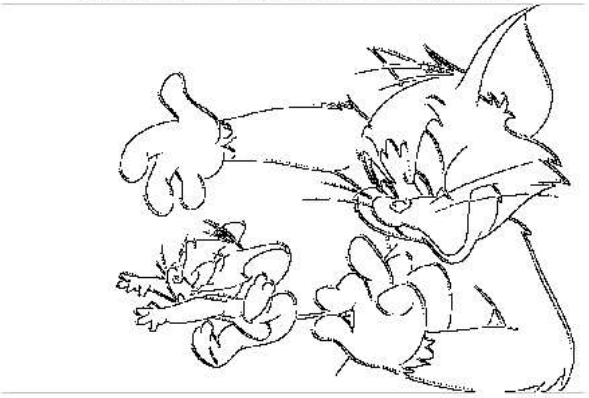
```
Warning: Image is too big to fit on screen; displaying at 67%
```



Resultado da Aplicação da Máscara 1



Resultado da Aplicação das Máscaras de Sobel



Resultado da Aplicação das Máscaras de Prewitt

Published with MATLAB® R2018a